



# **TIBCO® Data Science - Team Studio**

## **User Guide**

Version 7.1.0 | September 2023

# Contents

---

<b>Contents</b>	<b>2</b>
<b>The TIBCO Data Science - Team Studio Environment</b>	<b>6</b>
Workspaces	6
Creating a Workspace	7
Workspace Quick Start Guide	8
Other Workspace Activities	8
Workspace Tabs	11
Workflows	62
Supported Workflows	63
Workflow Editor	64
Workflow Actions	65
Workflow Menu	78
Data Explorer	79
Operator Explorer	85
Operator Help	86
Creating a New Workflow	87
Creating a New Legacy Workflow	91
Explore Visual Results	96
Spark Optimization for Data Scientists	98
Deploying Models and Workflows	109
Preview and Visualize Data	114
Running a Workflow	134
Viewing Workflow Results	140
Workflow Variables	146
Touchpoints	153
Creating a Touchpoint	154
Touchpoint Parameters	164

Search .....	172
Search Page Options .....	172
Tags .....	175
Viewing Tags .....	175
Navigating with Tags .....	176
Adding and Editing Tags .....	177
Deleting a Tag .....	178
Renaming a Tag .....	178
Jupyter Notebooks .....	179
Creating a Jupyter Notebook .....	179
Installing Python Packages .....	180
Initializing PySpark for Spark Cluster .....	182
Initializing PySpark for Hadoop Distribution .....	184
Creating a Custom Environment for Running Jupyter Notebooks .....	187
Uploading and Running the Conda Environment Example .....	188
Adding Your Data to a Notebook .....	189
Reading Your Data Using Direct Connection in a Notebook .....	192
Incorporating Notebooks in a Workflow .....	196
<b>Workflow Operators .....</b>	<b>197</b>
Operator Actions .....	197
Connecting Operators .....	197
Selecting Multiple Operators .....	199
Editing Operator Properties .....	200
Deleting Operators .....	201
Moving Connections .....	202
Deleting Connections .....	202
Data Management .....	203
Selecting Groups of HDFS files .....	204
Data Exploration .....	205
Visualizing data with charts and graphs .....	206
Explore Visual Results .....	206

Correlation and Covariance .....	209
Information Value and Weight of Evidence Analysis .....	210
Data Transformation .....	211
Aggregation Methods for Batch Aggregation .....	211
Outliers in Numerical Data .....	213
Creating a Join condition for a database join .....	215
Key-Value Pairs Parsing Example using the Variable Operator .....	216
datetime Format Conversion Examples .....	217
Spark SQL Syntax and Expressions .....	219
Data Modeling and Model Validation .....	222
Cluster Analysis Using K-Means .....	222
Patterns in Data Sets .....	232
Alpine Forest Operators .....	234
Model Export Formats .....	237
Fitting a Trend Line for Linearly Dependent Data Values .....	240
Probability Calculation Using Logistic Regression .....	263
Classification Modeling with Decision Tree .....	285
Classification Modeling with Naive Bayes .....	300
Computed Metrics and Use Case for the Regression Evaluator .....	308
Collaborative Filtering .....	310
Prediction Threshold .....	311
Principal Component Analysis .....	313
Support Vector Machine Classification .....	315
T-Tests .....	320
Testing Models for Performance Decay .....	323
Prediction .....	327
Prediction and Modeling Operator Pairings .....	328
Pearson's Chi Square Operations .....	329
Spark Node Fusion .....	331
Viewing Results for Individual Operators .....	332
Specialized Tools .....	332
Natural Language Processing Tools .....	333



Using Pig User-Defined Functions .....	348
DateTime Input Values .....	348
Setting Up Notebooks for Python Execute .....	350
R Execute .....	352
<b>Workflow Operator Reference .....</b>	<b>360</b>
Legacy Operators .....	360
Apache Spark Specific Operators .....	1016
Operator dialogs .....	1244
Operator Compatibility .....	1282
Processing Tools for Hadoop-Enabled Operators .....	1290
<b>Glossary .....</b>	<b>1295</b>
<b>TIBCO Documentation and Support Services .....</b>	<b>1302</b>
<b>Legal and Third-Party Notices .....</b>	<b>1304</b>

# The TIBCO Data Science - Team Studio Environment

---

The TIBCO Data Science - Team Studio environment was built to facilitate the analytics process at all levels of the business, from defining a problem to driving business action. The collaborative user interface in TIBCO Data Science - Team Studio allows cross-functional teams to work together on data science projects and build machine-learning workflows using a web interface and a minimum of code, while leveraging big-data platforms.

Each component of the application focuses on a different task.

## Workspaces

Workspaces are self-contained projects within TIBCO Data Science - Team Studio. You can add data sets and files, invite others to collaborate, and share progress. You can also build and run analytic workflows and notebooks within the workspace.

Within a workspace, you can perform the following tasks.

- Add and visualize data.
- Create workflows.
- Create notebooks.
- Create Touchpoints.
- Run SQL files.
- Manage scheduled and on-demand jobs.
- Add a sandbox.
- Add and edit members as well as their roles.
- Change the stage of your workspace.
- Create milestones.

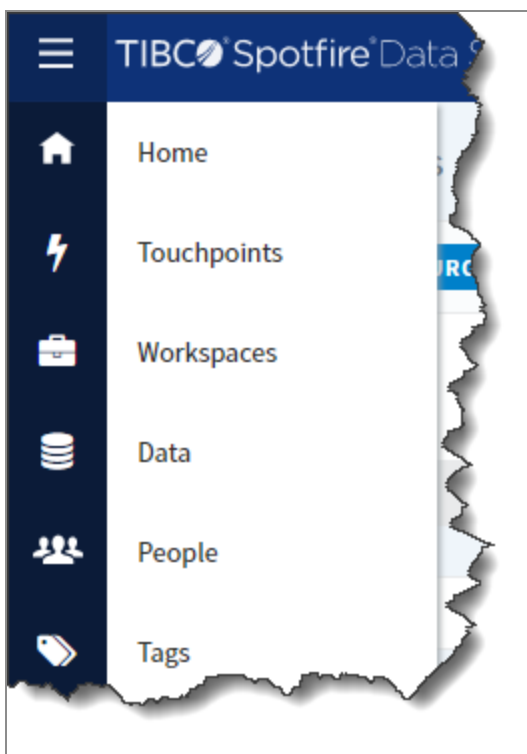
## Creating a Workspace

Workspaces are the virtual areas where you collaborate with your team to manage data.

- You can create your own workspace and add others as a team member.
- Your workspace keeps you current on what's going on, links to relevant datasets, and stores various work files your project depends on.

### Procedure

1. Click **Workspaces** on the sidebar, and then click **Create New Workspace**.



2. Enter a name for the new workspace, and then specify whether it should be public or private.
  - Public - All authenticated users can view the workspace. Members can edit the workspace.
  - Private - Only members can view or edit the workspace.

**CREATE A NEW WORKSPACE**

Workspace Name <sup>\*</sup>  
sample\_workspace

Workspace Privacy

☐ Public  
All authenticated users can view the workspace. Members can edit the workspace.

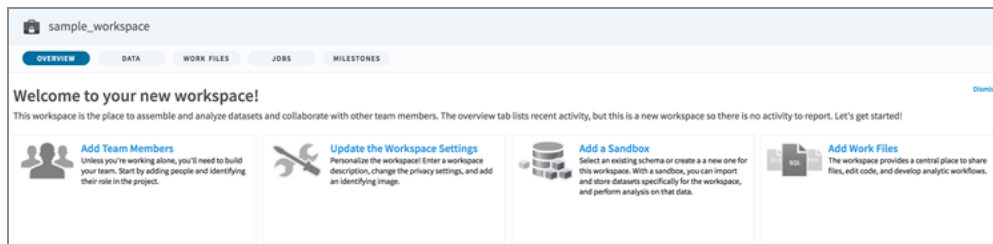
☒ Private  
Only members can view or edit the workspace.

<sup>\*</sup> Required

Cancel Create Workspace

## Workspace Quick Start Guide

After you create your new workspace, the overview page appears. This shows a quick start guide that suggests tasks to get started. You can dismiss this guide at any time.



## Other Workspace Activities

You can perform the following activities for each opened workspace.

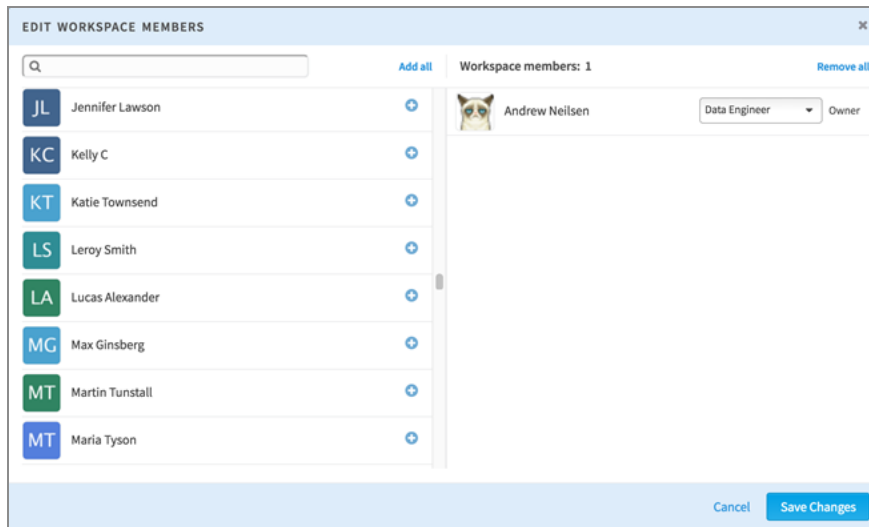
- Add a note to a dataset for others to understand the contents or usage of the data, for example. The note can be viewed from the table's activity section.
- Add an insight to make a comment about business analysis or insight gained.

- Change the workspace settings. Click **Workspace Settings** to open a window for changing specific details about the workspace.

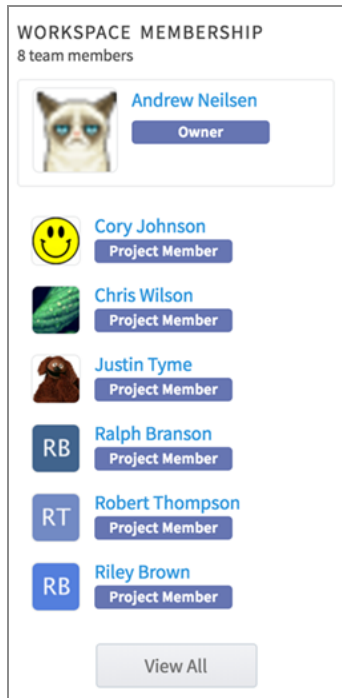
- **Add or Edit Workspace Members** allows for managing users with access to the workspace. On the left side is a list of all users in the application. Click + to add them to the workspace. They receive a notification saying that you have added them. After you add them, you can select a workspace role for them. This helps with team communication and messaging of what each person works on in a project. You can

have only one workspace role at a time, but it can be changed at any time. The available roles are:

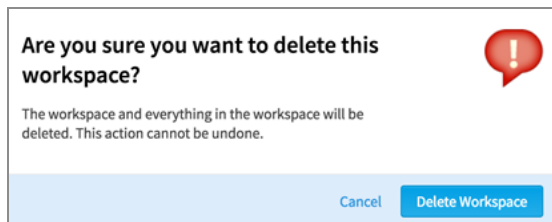
- Business Owner
- Business Analyst
- Data Science Manager
- Data Scientist
- Data Engineer
- Application Engineer
- Project Member



After you add members to the workspace, they are displayed on the sidebar of the **Overview** tab, along with their workspace roles.



- Remove a workspace and the files in the workspace by clicking **Delete**.



## Workspace Tabs

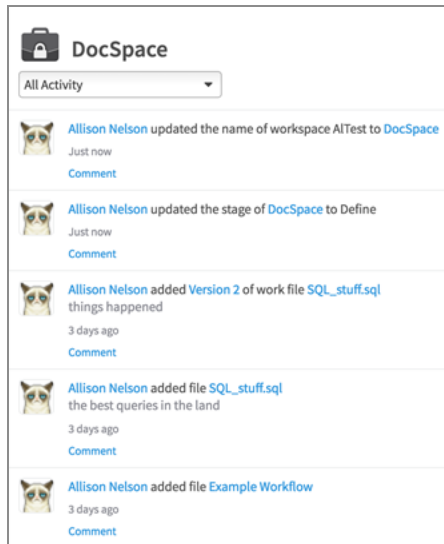
The workspace screen is a centralized location to manage your workflows, data sources, and analytic results. Workspaces consist of the following seven tabs that contain the project information.

### Overview Tab

The **Overview** tab provides a listing of activity on this workspace, as well as information on workspace membership and the workspace stage.

In a workspace, activity is automatically logged in the activity list in the main panel of the **Overview** tab. This includes association of datasets, addition of code, new queries, or any

other activity on the part of any team member. You can see at a glance what has been done recently. The default workspace view is **Show All Activity**. This view displays all of the activity that has occurred in the workspace, such as names of files that have been added, scheduled imports, and any associated comments and insights. You can switch to **Insights** and filter the activity stream to show only insights made on the workspace or objects that belong to the workspace.



Status information is displayed based on milestones managed on the **Milestones** tab. This shows the workspace owner, if the project is on track (that is, if milestones due have been completed), and the number of days until the project is expected to complete. A progress chart shows the milestones completed. In addition, you can see the stage of the workspace. This helps you track your progress through an analytics project.

A TIBCO Data Science - Team Studio project has five stages.

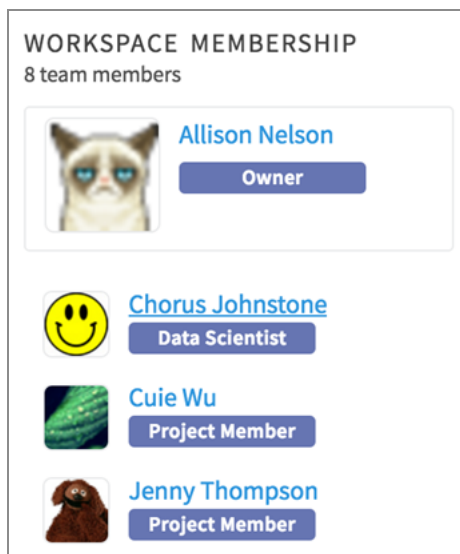
- Define
- Transform
- Model
- Deploy
- Act

You can set the stage of the workspace by selecting **Update Stage** on the sidebar.





A list of members of the workspace is displayed below the sidebar options. You can see the people who are members of the workspace and their workspace roles.

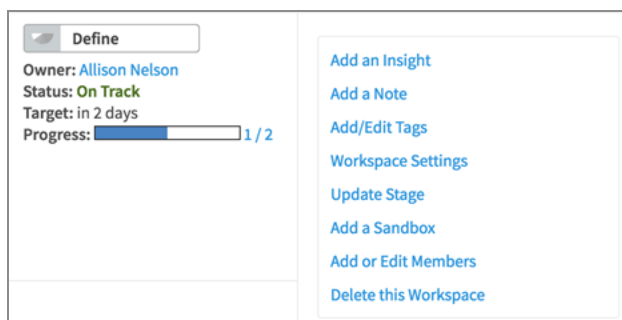


## Workspace Stages

Workspace stages allow you to understand and track your progress through an analytics project.

The process starts with the definition of a business problem, and continues all the way to implementation of a business action based on the results of your project.

All new workspaces start in the **Define** stage. To change your workspace stage, select **Update Stage** from the **Overview** tab.



Use the dropdown list to choose your workspace stage and enter an optional comment.

**UPDATE THE WORKSPACE STAGE** ✕

Stage  
Define ▼

Define the two essential components of a machine learning project. What business problem are we solving? How should the solution change the way our business operates? Locate or acquire all necessary data.

Comments

Cancel Update Stage

The available stages are:

<b>Define</b>	Define the two essential components of a machine learning project. What business problem are we solving? How should the solution change the way our business operates? Locate or acquire all necessary data.
<b>Transform</b>	Transform and cleanse data relevant to the problem into a form amenable to modeling. Perform initial feature selection and engineering.
<b>Model</b>	Create, evaluate, and optimize machine learning models.
<b>Deploy</b>	Deploy models in batch scoring jobs, real-time scoring APIs, and Touchpoints. Integrate the deployed models into business processes.
<b>Act</b>	Monitor and maintain the operational model. Perform minor optimizations. Measure business impact based on new behavior.

This stage information is displayed on your workspace overview as well as in the workspace list and in search results.

You can change your workspace stage at any time. Each change appears as an item in your workspace's activity feed.

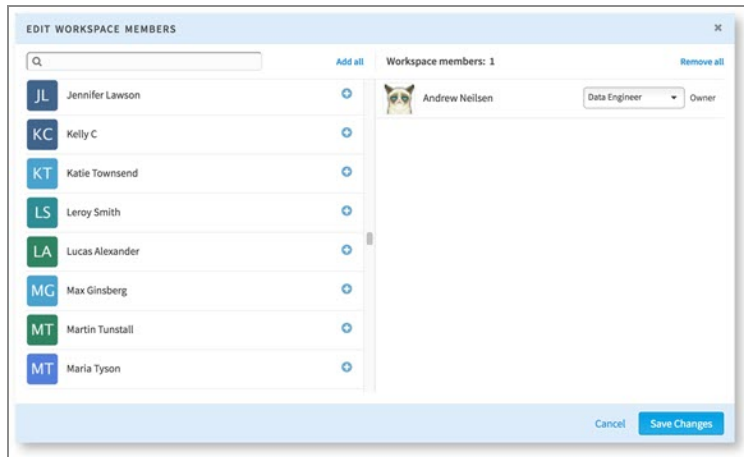
For information about workspace roles, see [Workspace Roles](#).

## Workspace Roles

Workspace roles describe a person's expertise and role in the project. This helps with team communication and messaging of what each person works on. You can have only one workspace role at a time, but it can be changed at any time. The available roles are:

Item	Description
<b>Business Owner</b>	Business Owners deeply understand the problem or opportunity a project addresses, and how it should impact behavior. They evaluate model output, and direct Application Engineers to integrate that output into relevant business processes.
<b>Business Analyst</b>	Business Analysts are the interface between the data science and business layers of the project. They help translate business requirements into requirements for the analytics team. They also use transformed data, and data captured from operationalized models, to generate visualizations and reports that summarize the business impact of the project.
<b>Data Science Manager</b>	Data Science Managers guide the data transformation and modeling process, ensuring that it aligns with the defined business problem or opportunity. They are the contact point between the business and the data science team. They might also build and evaluate transformation flows and machine learning models, and promote best practices.
<b>Data Scientist</b>	Data Scientists use the methods of statistics and machine learning to create meaning from data. They work closely with Data Engineers to cleanse and transform data, generate insights, and build predictive models.
<b>Data Engineer</b>	Data Engineers are responsible for connecting TIBCO Data Science - Team Studio to data sources and building transformation flows that represent relevant data in an appropriate form for feature engineering and modeling.
<b>Application Engineer</b>	Application Engineers build the API integrations and/or user interfaces that bring the output of operationalized models to end users or customers.
<b>Owner</b>	The Owner is the point person for the project - often a Business Owner, Data Science Manager, or Project Manager.
<b>Project Member</b>	Project Members are team members whose roles do not fall into any of the other labeled roles.
<b>Project Manager</b>	Project Managers plan the schedule, identify risks, and coordinate the interdisciplinary groups involved in the project.

You can add or edit members from the workspace overview tab. Click **Add or Edit Members**. Use the + next to a user's name to add the user to the workspace. Then you can use the dropdown list to select a workspace role. Click **Save Changes** to update the workspace members list.



## Members and Non-Members

All workspace members can perform the following tasks:

- **Add an Insight:** Insights appear in the workspace activity stream and your home page. If the insight is published (accomplished by clicking **Publish** on the insight entry in the activity stream), it appears on all TIBCO Data Science - Team Studio users' home page activity streams. This is a good way to promote information to the whole team.
- **Add a Note:** You can comment on the workspace here. Comments can have file attachments and text formatting. You can also notify selected members on your project team about your activity.
- **Add and edit Tags:** Add or edit tags for this workspace. Tags help you search for and categorize projects and files.
- **Edit Workspace Settings:** Members of the workspace who are not the workspace owner or a TIBCO Data Science - Team Studio administrator can edit the workspace name and summary.
- **Update the Stage:** You can update the stage of your workspace here. This allows you to keep track of your project through the analytics process. There are five roles: Define, Transform, Model, Deploy, and Act. Selecting a role displays information

about that role. You can also add a comment when you change a workspace role. See [Workspace Stages](#) for details.

- **Add a Sandbox:** When you add a sandbox, you can easily import files or datasets from your data source. These are your workspace's copies, and you are free to manipulate and transform them. A workspace can only have one sandbox, and once you have added a sandbox, you cannot remove it.

Non-members can view the workspace in read-only mode, but only if the workspace is public. Non-members can also:

- Add an Insight
- Add a Note
- Add and edit Tags

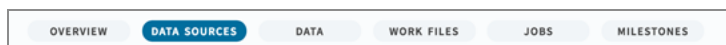
## Owners or Team Studio Administrators

The following is a list of additional tasks that you can perform if you own the workspace or have an Administrator role:

- **Edit Workspace settings:** You can edit all the fields in the Edit Workspace Settings dialog, such as changing the workspace privacy, owner, and status.
- **Add and edit members:** A list of currently existing members is displayed on the left side of the dialog, while the current workspace members are on a list on its right side. Click + next to a person's name to add them to a workspace. You can also edit their workspace role from this dialog.
- **Delete this workspace:** Use this command to delete the current workspace and everything in it. All associated imports are unscheduled. Performing this action cannot be undone.

## Data Sources Tab

The **Data Sources** tab provides a listing of the currently associated data sources in your workspace.

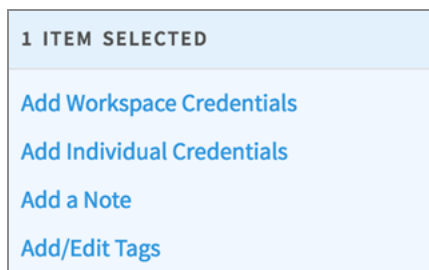


Similar to the **Data** section, this section displays a list of the data sources that are associated or available to your workspace. Keep in mind that Public data sources are

available to all workspaces, while Limited data sources must be associated to your workspace specifically. A Data Administrator can associate these data sources.

Data Sources	
79 Data Sources	Visibility
 1_Public_individual	Public
 gpdb_10.10.3.151	Public
 gpdb_151	Public
 GPDB-S8 Greenplum 4.3.8.0	Public
 GPDB_gws	Public

Databases and Hadoop clusters have their own sections in the list, and each row describes the data source's visibility. When you select a data source, you can see information about the credential settings and add your own, if the data source is configured to accept separate credentials. For example, the data source below has Public visibility and is configured to accept Individual credentials. When you select it, you can choose to add Workspace credentials or Individual credentials.



If you add Workspace credentials, every member of the workspace can access the data source using those credentials. If you select Individual credentials, only you (the user) can access the data source with the credentials you provide. Other members of your workspace must provide their own credentials.

**ADD WORKSPACE CREDENTIALS** [X]

1\_Public\_individual

These credentials will override any existing credentials when accessing this data source in this workspace. Press cancel to use the existing shared or individual credentials in this workspace.

Username

Password

Cancel Submit

**ADD DATA SOURCE CREDENTIALS** [X]

Enter your account credentials for 1\_Public\_individual.

Username

Password

Cancel Save Credentials

When a data source is selected, you can see recent activity for that data source, as well as add notes or tags.

**Shared Credentials**

1 ITEM SELECTED

[Add a Note](#)

[Add/Edit Tags](#)

**ACTIVITY** DETAILS

**NK** Nate Kapinos added a new data source `gpdb_151`

8 days ago

[Comment](#)

To edit a data source, you must navigate to the main **Data** section of the application. You can only do this if you are a Data Administrator or higher.

## Associating a Data Source with a Workspace from Your Workspace

Associating a data source to a workspace allows you to see and use data from that source in your workspace. This is especially useful if you have a data source whose visibility is

specified as Limited. A Data Administrator or Application Administrator role is required to complete this task.

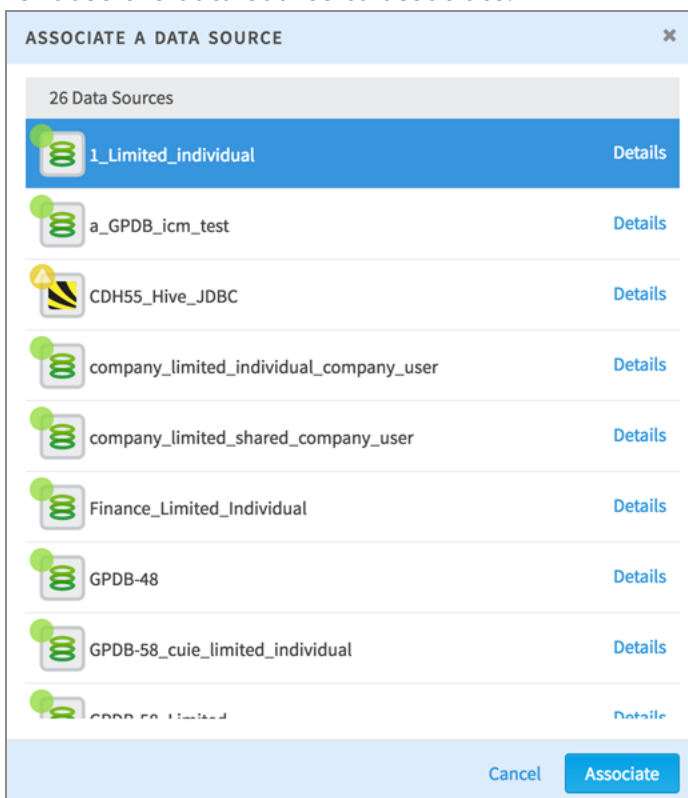
Start this task in an open workspace.

## Procedure

1. Navigate to the **Data Sources** section of your chosen workspace.
2. Select **Associate a Data Source**.



3. Choose the data source to associate.



## Associating a Data Source with a Workspace from the Data Section

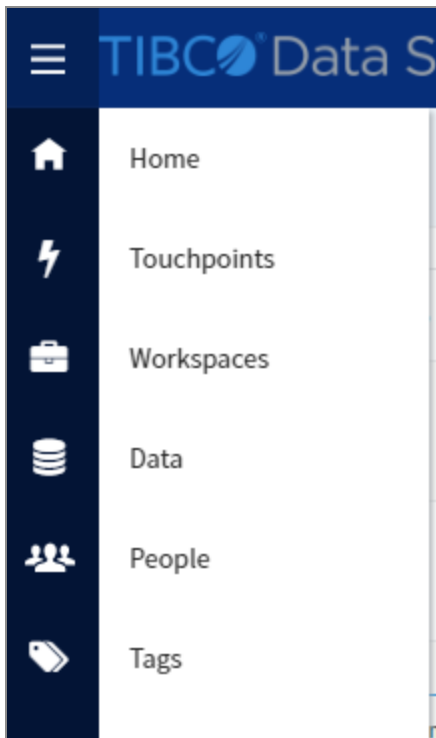
Associating a data source to a workspace allows you to see and use data from that source in your workspace. This is especially useful if you have a data source that is Limited visibility. A Data Administrator or Application Administrator role is required to complete this task.



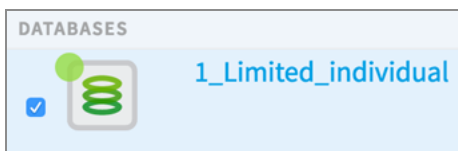
An open workspace.

## Procedure

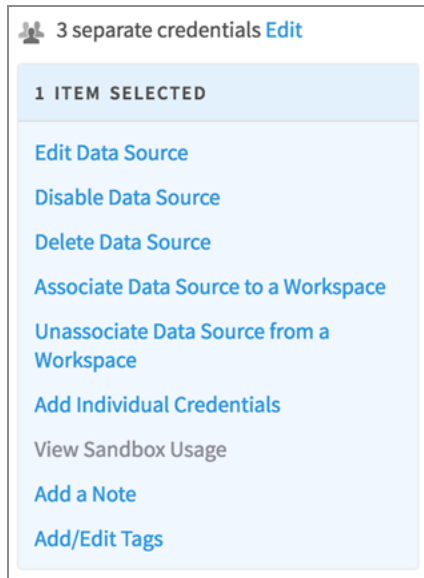
1. Navigate to the main **Data** section of the application.



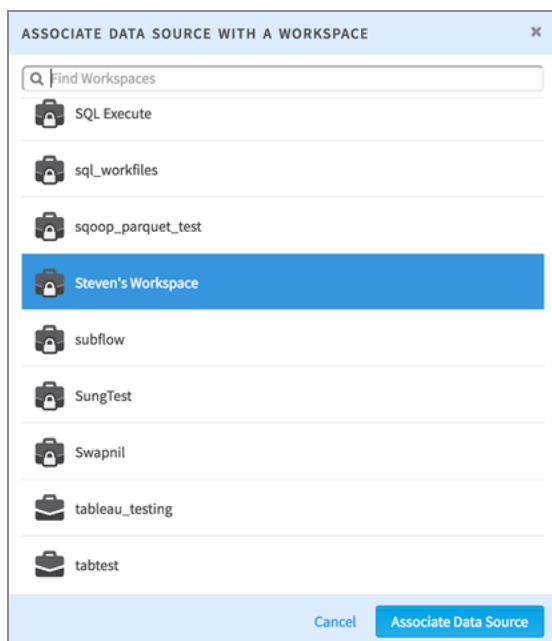
2. Select the data source you want to associate.



3. Choose **Associate Data Source to a Workspace**.



4. Select the workspace to which you want to associate the data.

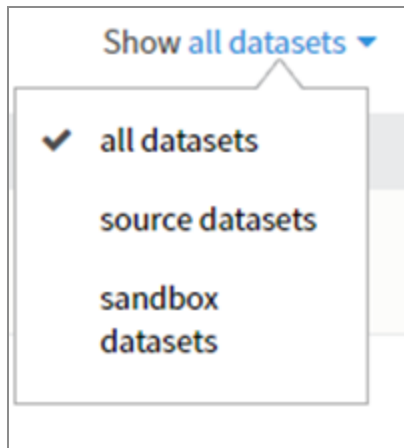


5. Click **Associate Data Source**.

## Data Tab

The **Data** tab displays a list of datasets that are in the workspace sandbox or are associated with the workspace.

By default, the **Data** tab shows all datasets, but you can also filter by source datasets or sandbox datasets.



- Source datasets are datasets that are associated with the workspace but are not in the workspace sandbox. You can import source datasets into the workspace sandbox using the **Import Now** command, after which they become sandbox datasets.
- Sandbox datasets are datasets that are in the workspace's sandbox schema.

If you do not have a sandbox, you can still associate datasets with the workspace.

## Exploring a Data Source

To get details on a data source, you choose it from a list of data sources that are associated with or available to your workspace.

Start with an open workspace.

### Procedure

1. On the **Data Sources** tab, select the dataset to explore by clicking its name.

The details page of the dataset is displayed.

**i Note:** The data view depends on the data source (DB, Hadoop, Hive, or TDV). For example, if you are working with Hadoop, the details page is displayed as rows of comma-separated values. If you are working with PostgreSQL, you see a view similar to the one shown in the following image.

Here is a sample dataset that has been associated with the workspace:

id	times90dayslate	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslate_2years	srslagny

Any TIBCO Data Science - Team Studio user with permission to view this dataset can use these commands:

- [Previewing Data in a Workspace Table](#)
- [Visualizing Data in a Workspace Table](#)
- [Analyzing Data in a Workspace Table](#)
- Adding a Note: Select this option to add a note to a dataset in a schema. If you are working from the workspace, you can support your note with attachments from the desktop, datasets, or work files. If you are working from the data source browser, you can support your note only with attachments from the desktop.
- [Associating Datasets with a Workspace](#)
- [Adding and Editing Tags](#)
- [Creating a New Workflow](#) (Data Analysts and Analytic Developers only)
- Downloading (A specific number of rows or the entire dataset)

**Important:** The **Explore**, **Data Preview**, **Visualize**, and **Analyze** actions can also be run on datasets that you browse to from the main data sources page.

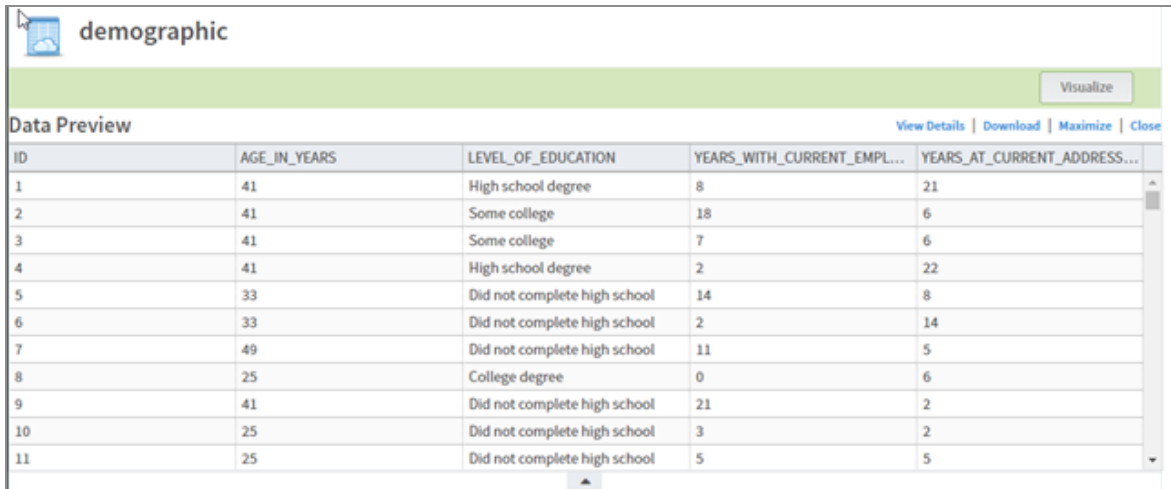
## Previewing Data in a Workspace Table

When you preview data in a table, the table is queried and the first 100 rows are returned. You can also download a specific number of rows or all rows to your desktop.

## Procedure

1. To preview a table's content, click **Data Preview**.

The following is an example of previewed data.



ID	AGE_IN_YEARS	LEVEL_OF_EDUCATION	YEARS_WITH_CURRENT_EMPL...	YEARS_AT_CURRENT_ADDRESS...
1	41	High school degree	8	21
2	41	Some college	18	6
3	41	Some college	7	6
4	41	High school degree	2	22
5	33	Did not complete high school	14	8
6	33	Did not complete high school	2	14
7	49	Did not complete high school	11	5
8	25	College degree	0	6
9	41	Did not complete high school	21	2
10	25	Did not complete high school	3	2
11	25	Did not complete high school	5	5

## Visualizing Data in a Workspace Table

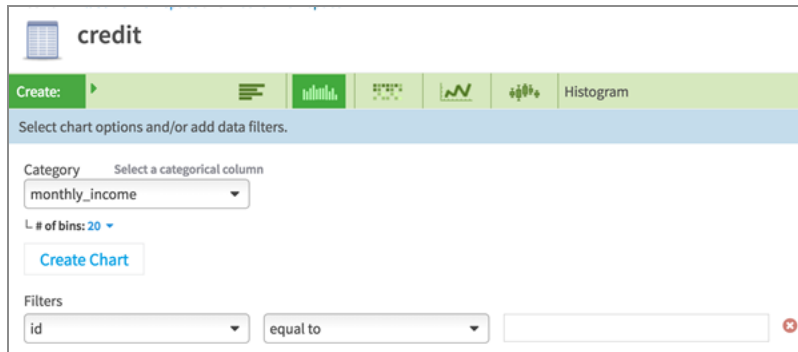
You can create a visual representation of your data, and dynamically add data filters, and switch axes and chart types. You can visualize your data with following types of charts:

- Frequency chart
- Histogram (currently only supports numerical data)
- Heatmap (currently only supports numerical data)
- Time Series
- Boxplot

## Procedure

1. Select a table, and then click **Visualize**.
2. From the **Create** toolbar, select the type of visualization.

### Example



**credit**

Create: Bar Chart Scatter Plot Line Chart Area Chart Histogram

Select chart options and/or add data filters.

Category Select a categorical column  
monthly\_income

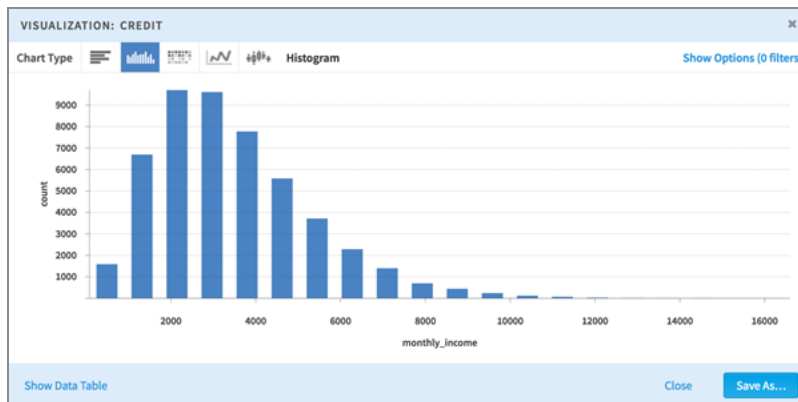
L # of bins: 20

Create Chart

Filters  
id equal to

3. If prompted, select options or filters for the type of visualization.
4. Click **Create Chart**.

### Example



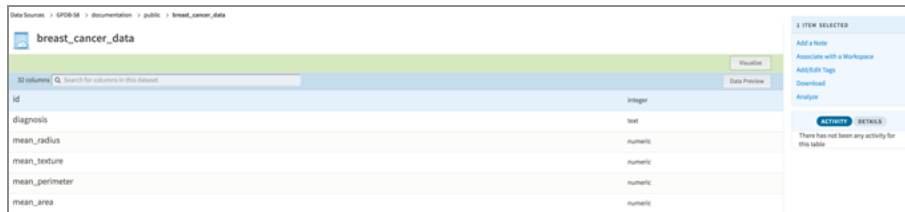
## Associating Datasets with a Workspace

When you are browsing data sources and you find datasets that you would like to work with, you can associate those datasets with your workspace.

### Procedure

1. Select the dataset(s) to associate.

It should show a preview of the dataset and some options on the sidebar.



## 2. Select **Associate with a Workspace**.

### Result

Associating a dataset means that it appears on the [Data Tab](#) on your workspace. The link to the dataset remains there for easy reference, and you can use associated datasets in [Jupyter Notebooks](#).

For other data, see [Importing Data into a Workspace's Sandbox](#).

## Importing Data into a Workspace's Sandbox

When you create a sandbox, all of the datasets or tables in the named schema are automatically added to the sandbox. They appear in the dataset list under the workspace data tab as sandbox datasets. You can also import data manually.

### Manually Importing External Files

Perform the following steps for importing the external files:

Choose **Import File**.



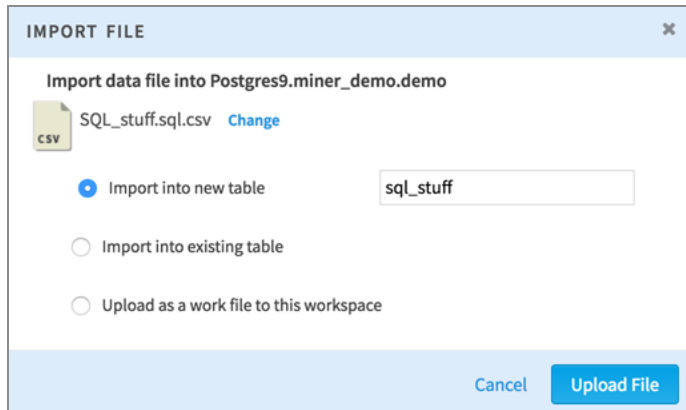
A dialog for adding a file from your system is displayed.



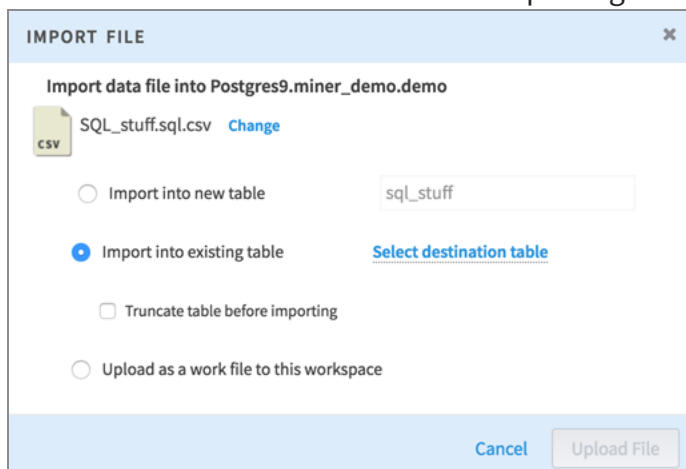
After selecting a file, you can choose where to place it.

- **Import into new table:** Choose a name for the new table. This table is created in

your sandbox schema.



- **Import into existing table:** Choose **Select destination table** to choose a location for the new dataset.
  - Use **Truncate table before importing** if you have an auto-increment column that needs to be reset before importing.



- Use **Upload as a work file to this workspace** to upload this file as a regular work file to the workspace. It is not associated with the sandbox schema.

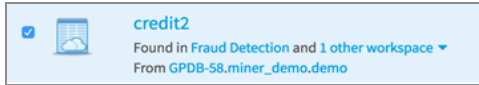
## Importing Associated Data

Follow this procedure to import associated data into a workspace's sandbox.

### Procedure

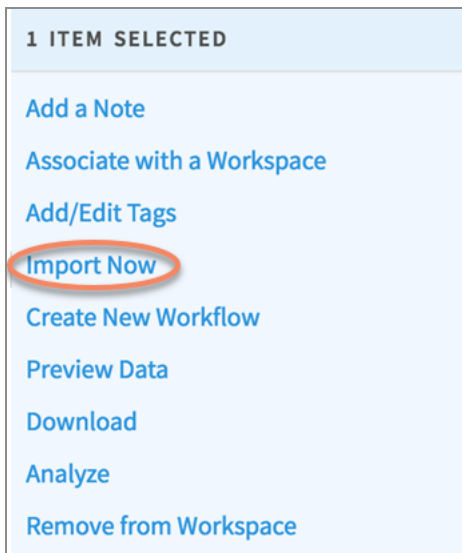
1. Select the dataset that interests you.





2. Choose **Import Now** from the contextual sidebar of the data tab to import the dataset into your sandbox.

**Note:** You can create imports only from source datasets or TIBCO Data Science - Team Studio views in a workspace.

A screenshot of the 'IMPORT NOW' dialog box. It has a title bar with a close button. The 'Import Destination' is 'Postgres9.miner\_demo.demo'. There are two radio button options: 'Import into new table:' (selected) with an empty text input field below it, and 'Import into an existing table:' with a 'Select destination table' button. Under the 'Options' section, there are two checkboxes: 'Truncate table before importing' (unchecked) and 'Limit the number of rows to import' (unchecked) with a text input field containing '500'. At the bottom, there are 'Cancel' and 'Begin Import' buttons.

If you are importing data into a new table, choose a valid name for the table. This

table is created in your sandbox schema.

**i Note:** Valid table names must start with a letter followed by alphanumeric and/or underscore characters with a maximum length of 64 characters.

- If the **Limit the number of rows to import** option is selected, enter the number of rows to import. If this option is left unselected, the entire dataset is imported.

If you are importing data into an existing table, use **Select destination table** to choose a location for the new dataset.

- Use **Truncate table before importing** if you have an auto-increment column that must be reset before importing.
  - If the **Limit the number of rows to import** option is selected, enter the number of rows to import. If this option is left unselected, the entire dataset is imported.
3. Click **Begin Import** to start the import process.

## Importing Oracle Datasets

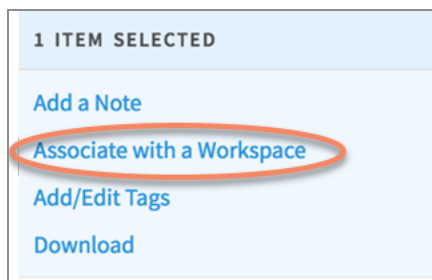
Follow this procedure to import an Oracle dataset into a workspace's sandbox.

### Before you begin

Start from an open workspace.

### Procedure

1. Browse to the Oracle dataset to import.
2. Select the **Associate with a Workspace** command.



3. Follow the instructions in [Importing Associated Data](#) to add the table to the

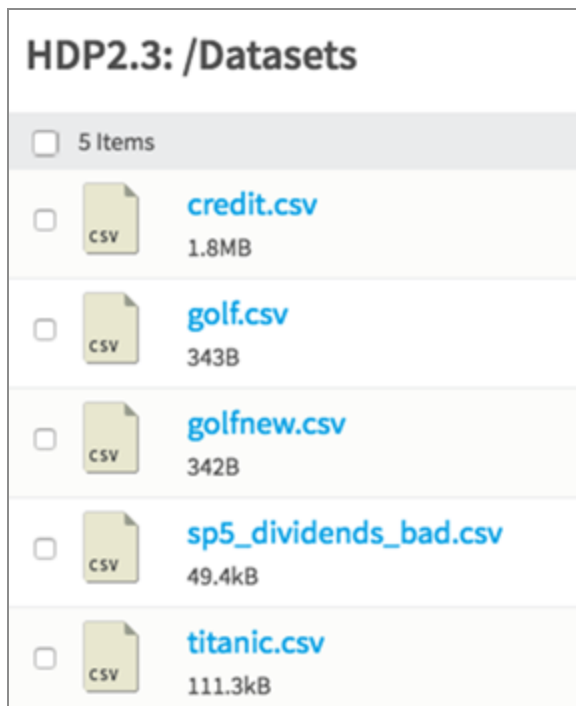
workspace.

## Importing Hadoop Datasets

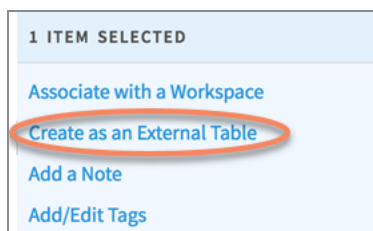
Follow this procedure to import a Hadoop dataset into a workspace's sandbox.

### Procedure

1. Browse a Hadoop data source to obtain a list of directories and files.



2. Select the CSV file, and from the right pane, click **Create as an External Table**.



**CREATE EXTERNAL TABLE**

Select workspace: DocSpace Table name: credit\_csv

Delimiter: ☐ Tab ☒ Comma ☐ Semicolon ☐ Space ☐ Pipe ☐ Other

column_1	column_2	column_3	column_4	column_5	column_6	column_7	column_8
text	text	text	text	text	text	text	text
id	times90dayslate	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslate...	srsdlqncy
2	0	0.26	0.324	4	1956.592205	0	0
6	0	0.25	0.927	6	3983.456356	0	0
52	0	0.46	0.372	7	3595.370678	0	0
76	2	0.88	0.154	3	1729.964479	2	0
84	0	0.14	0.69	7	2507.165171	0	0
86	0	0.11	0.131	5	2126.599617	0	0
100	0	0.32	0.188	7	5158.108717	0	0
112	0	0.28	0.251	4	2752.624174	0	0
132	0	0.17	0.343	4	1108.773033	0	0

Cancel Create External Table

- From the **Select workspace** dropdown menu, browse a list of workspaces for which you are a member.  
Only workspaces with sandboxes are displayed.

- In the **Table name** box, choose a table name for your import.

Be sure that you use a table name that is valid for your database provider.

TIBCO Data Science - Team Studio attempts to determine which delimiter your CSV file uses. If you use a non-standard delimiter or this determination is wrong, use the **Delimiter** command to choose a new delimiter.

A preview of the data in tabular format is displayed. Verify that this format is correct, and then click **Create External Table**.

## Result

The new external table is created in the sandbox schema of the workspace you choose.

## Scheduling Recurring Imports

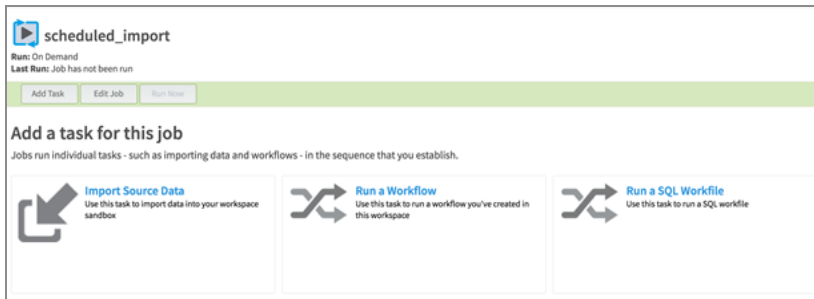
You can schedule an import of a data source dataset so that it runs automatically at an interval you specify. This is useful if you have a table that is often updated and you want to add new data as it comes in. TIBCO Data Science - Team Studio starts the process at the time you specify and imports the data into the destination table.

If you choose to notify people upon the success or failure of the job, they receive email notifications of the job's status.

## Procedure

1. Navigate to your workspace's **Jobs** tab.
2. Click **Create a Job**.

3. Choose a name and an optional description for the job.
4. Specify whether to run the job on demand or on a schedule.
  - a. If you choose **On Demand**, the job runs when a user navigates to the job in the workspace and clicks **Run Now**.
  - b. If you choose **On a Schedule**, the job runs automatically based on a schedule you provide. To make a recurring data import, choose this option.
5. Specify your preferences for job notifications. This controls who is notified about the job status. You can choose settings for job success, job failure, or both. The people you choose receive email notifications and TIBCO Data Science - Team Studio notifications.
6. Click **Create**. The job detail page is displayed.

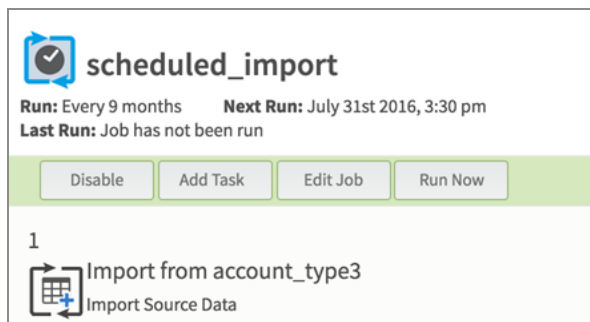


7. Select a task to add to the job. To set up a recurring data import, click **Import Source Data**.

8. Select the source table to import. All eligible tables for import from this workspace are shown. In this example, you can see a few TIBCO Data Science - Team Studio views that were generated by a SQL query.

9. Specify the destination table. This can either be an existing table or a new table. If you choose a new table, you must input a valid table name.

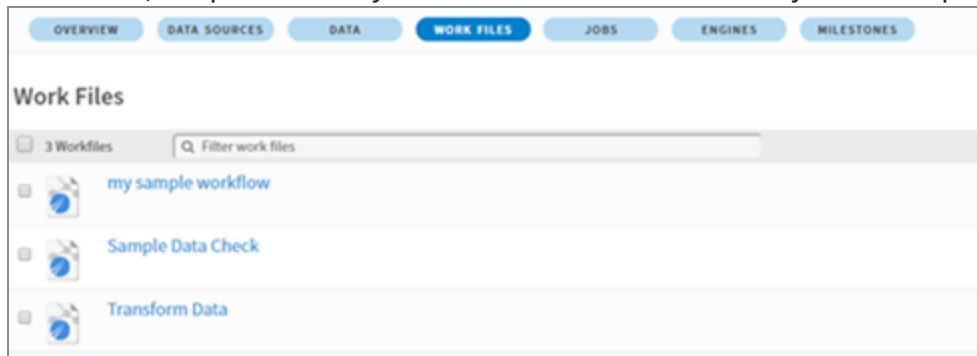
10. Specify any additional options. For example, you can choose to truncate the existing table before importing, which can be useful if you have auto-increment columns to reset. You can also specify how many rows of the data set to import.
11. Click **Add** to add the task to this job. It should now appear as a task on the job page. It shows the run frequency, the next run time, and the last run.
12. Click **Enable** to start the job scheduler. TIBCO Data Science - Team Studio automatically runs this job at the time listed as **Next Run**. To run the import sooner, click **Run Now**.



13. To add extra steps to the job, select **Add Task** and add as many as you want. You can drag them around to specify the order in which they run.

## Work Files Tab

The **Work Files** tab is the holding place for text files, images, notebooks, uploaded code, workflows, or queries that you have written and saved in your workspace.



You can create work files on the **Work Files** tab, as well as import them.

## Types of Work Files

The **Work Files** tab contains a list of the work files associated with this workspace. The default view for this section is **Show All Files**. The other dropdown options you can select are as follows:

- SQL files
- Workflows
- Notebooks
- Code files
- Text files
- Image files
- Other files

The work files are sorted alphabetically by default. You can also sort them by date.

## Creating a Work File

Work files are the files on which you perform analytics, extract data, and store results for your projects. You can find a list of work files as well as options to create new ones from the **Work Files** tab of a workspace.

The types of work files are:

- Workflows
- Notebooks
- TIBCO Data Science - Team Studio Models
- SQL files
- Code files
- Text files
- Image files
- Links
- Other

You can create an SQL file, a notebook, or a workflow directly from the workspace interface. The other types of files can be uploaded to the workspace using the **Upload a**



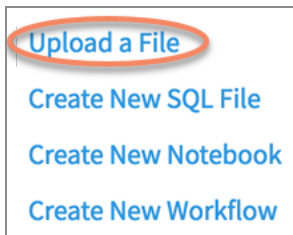
**File** command.

**i Note:** Valid names for work files include numbers, letters, spaces, and any of the characters ( ) . - \_

Work files with other characters in the name cannot be created.

## Procedure

1. To create a new work file, choose **Upload a File**.



2. From the Upload File dialog, click **Select a File**.
3. Browse and select the file to upload, and then click **Open**.
4. Provide a description, and then click **Upload File**.

## Result

The file is uploaded to Team Studio as a work file and is ready for use.

## Editing a Work File

When you select a work file from the list, a list of options is displayed on the right sidebar.

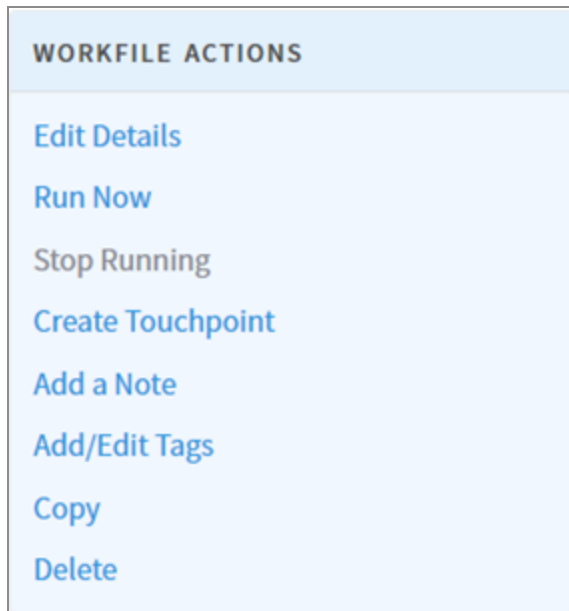
All work files have the following options.

- **Edit details:** Use to change the file name and description of your file.
- **Add a Note:** Use to add a note that describes updates or progress on this file. This note is published to the **Activity Feed** on your workspace.
- **Add/Edit Tags:** Use to add, edit, or remove tags for your work file here. You can also add tags for multiple work files at once by checking more than one work file. To learn more about tags, see [Tags](#).

- **Copy:** Use to copy this work file to another workspace.
- **Delete:** Use to delete a work file. You can only delete a work file if you own that file or if you are the administrator.

Special types of work files might have different options. For example, workflows include the option to **Run Now**, **Stop Running**, and **Create Touchpoint**. For more information about workflows, see [Workflow Editor](#).

To edit a work file, open it and choose a command from the menu on the right side of the screen.

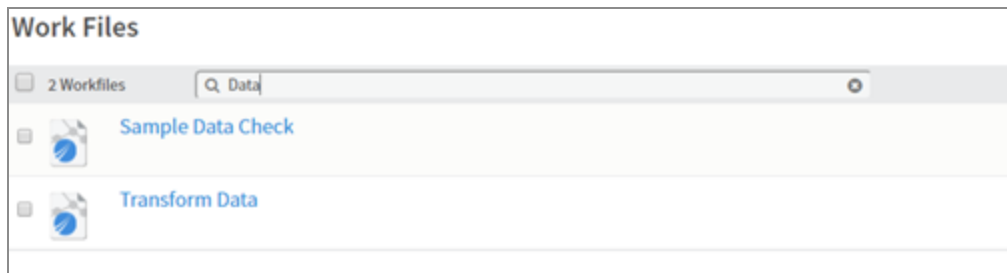


## Searching for a Work File

Work files can be easily searched based on their name or any tags that have been associated with them.

### Procedure

To search for a work file, enter its name in the search box.



The list of work files is automatically filtered down as you type in the search box.



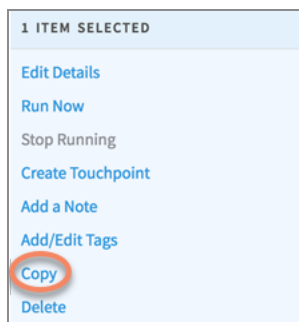
**Note:** The search capability for work files is case-sensitive, which means that searching for "data" is not the same as searching for "Data."

## Copying a Work File to a Workspace

You can create a copy of a work file in another workspace by using the **Copy** option from Workfile Actions. This can be used when you want to move the file to another one of your workspaces, or if you have a colleague's work file you want to make a copy of as a starting point.

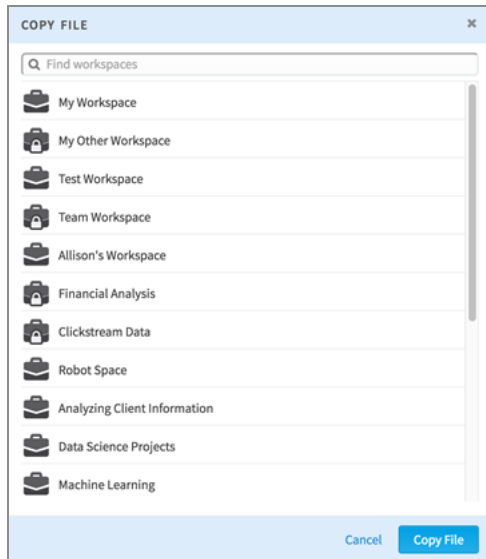
### Procedure

1. To add someone else's work file to a workspace, navigate to the work file to copy, and then select **Copy**.



The Copy File dialog is displayed.

2. Select which workspace to associate the work file with, and then select **Copy File**.



**Note:** When you add or import new workflows, TIBCO Data Science - Team Studio determines whether the flow depends on a data source to which the user does not currently have access. If the data source is not available for the current user, TIBCO Data Science - Team Studio prompts the user to supply an alternative. Otherwise, a confirmation message shows that the file is successfully copied to the workspace.

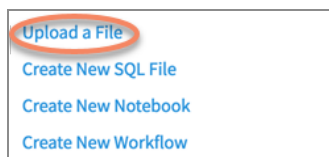
## Importing a Work File

Any type of work file can be loaded and stored together as part of the analysis in a workspace.

Start in the workspace list of work files, from the menu in the right pane.

### Procedure

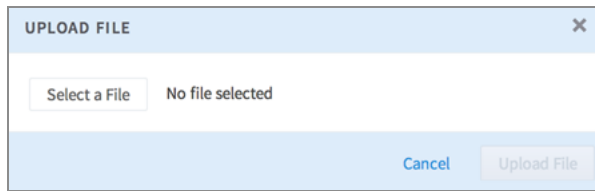
1. Select **Upload File**.



The **Upload File** dialog is displayed.

2. Select **Upload File**.

The file is imported into the workspace.



You can choose locally-stored workflow files (.afm), SQL files, CSV files, PDF files, results files, or any other related type of file used in the analysis.

**Note:** When you add or import new workflows, TIBCO Data Science - Team Studio determines whether the flow depends on a data source to which the user does not currently have access. If the data source is not available for the current user, TIBCO Data Science - Team Studio prompts the user to supply an alternative.

After you have uploaded the file, you can see it in the work file list for your workspace.

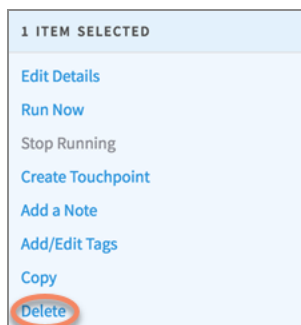
## Deleting a Work File

Work files can be deleted from a workspace.

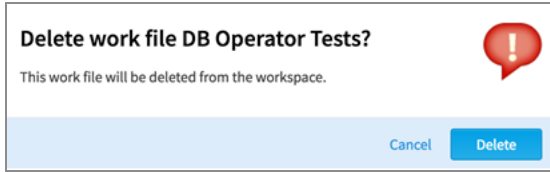
Start in the work files section of the work space.

### Procedure

1. Select the work file to delete.
2. From the Workfile Actions pane, click **Delete**.



An alert confirmation is displayed, prompting you to confirm removing the file from the workspace.



3. Click **Delete**.

## Creating a SQL Work File

You can write and run code within the data tab of the workspace page to manipulate a dataset in your sandbox. You can author your queries directly in our integrated SQL editor. Code can be written in SQL, PL/pgSQL or any other language installed in the target database.

Any SQL code developed in the SQL editor is automatically created in the schema of the workspace sandbox.

For information about the SQL Editor, see [SQL Editor](#).

**Note:** Valid names for SQL files include numbers, letters, spaces, and any of these characters: ( ) . - \_

Work files with other characters in the name cannot be created.

Start in the workspace list of work files, from the menu in the right pane.

### Procedure

1. Click **Create New SQL File**.



2. Provide a name and, optionally, a description for the file, and then click **Add SQL File**.

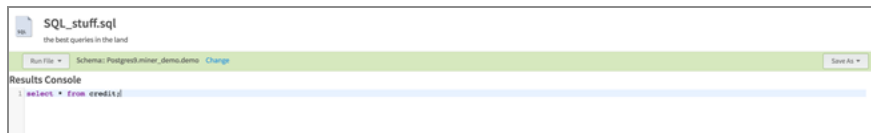
A query editor is displayed.

**i Note:** You can upload a SQL file (or any other type of file) with the **Upload a File** command on the **Work Files** tab in your workspace. For more information, see [Importing a Work File](#).

## SQL Editor

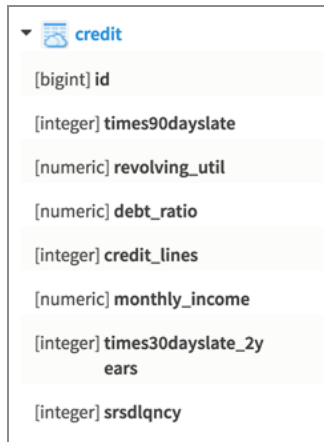
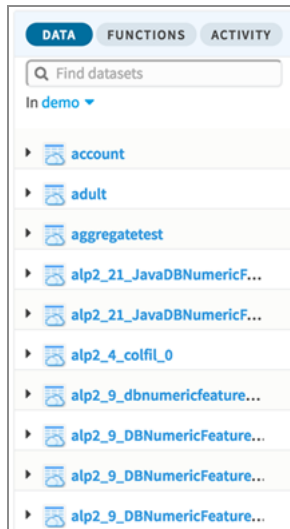
You can edit and run your queries using the SQL editor.

## SQL Editor



Begin typing your SQL statement, or click the **Data** or **Functions** menu on the sidebar to build a more complex query. Drag a table or a function to the SQL editor to use it in your query, and the syntax is copied into the box for you. Additionally, the SQL editor supports autocomplete, which you can access by typing control-space.

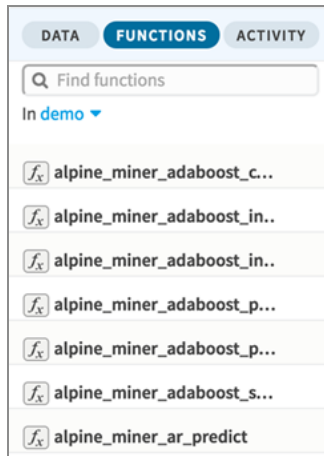
## Data



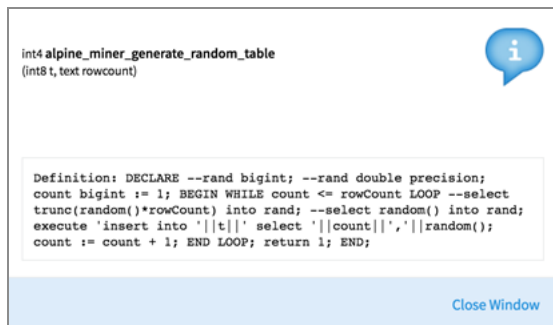
The tables from your sandbox schema are populated here. You can drag an entire table to the SQL editor to show the syntax for that table, or use the arrow to expand your selection and see the columns and their data types. You can also drag a single column to the SQL editor for use in a select statement, for instance.



## Functions



You can find user-defined functions (UDFs) and stored procedures here. Hover over a function to find more information about that function. To see a more detailed description, click **show more**.



TIBCO Data Science - Team Studio has a set of stored procedures that enable many machine learning algorithms on databases. For more information on stored procedures, see the topic "Database Stored Procedures" in the *TIBCO® Data Science - Team Studio System Requirements*.

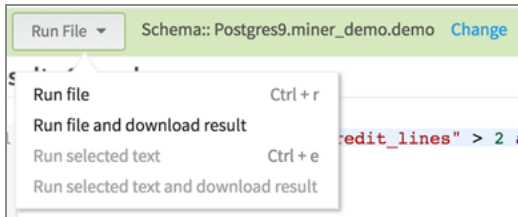
## Running a SQL Work File

Use this procedure to run your SQL work file.

Start in an open SQL workfile.

## Procedure

1. Click **Run File**.

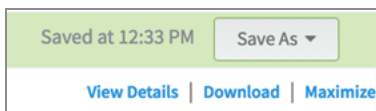


2. To see the output, pull down the **Results Console** with the small tab in its center. The **Results Console** is displayed just above the query editor.

id	times2daydate	revolving_util	debt_ratio	credit_lines	monthly_income	times2daydate_years	credit_lines
214	0	0.28	0.017	4	534.0020253	0	0
719	0	0.39	0.391	4	763.3642099	0	0
389	0	0.51	0.415	4	555.7949071	0	0
1574	0	0.27	0.331	4	755.2650896	0	0
306	0	0.35	0.38	5	555.1473509	0	0
958	0	0.38	0.085	4	918.2116896	0	0
1318	0	0.36	0.303	5	971.6774077	0	0
1409	0	0.18	0.605	4	761.4375958	0	0
1605	0	0.29	0.122	6	971.9020551	0	0
2331	0	0.15	0.207	4	886.4810947	0	0
2564	0	0.56	0.354	5	822.1033883	0	0

View Details | Download | Maximize

3. You can view details about the code's execution, download the results, or maximize the result pane. To do any of these, click the arrow below the result console to collapse it.



## Version Control in SQL Work Files

In a SQL work file, your code is automatically saved as you work on it. You can also save new versions so that you and your team can iterate quickly and build upon prior work.

Perform this task in the SQL work file.

## Procedure

1. Click **Save As**.

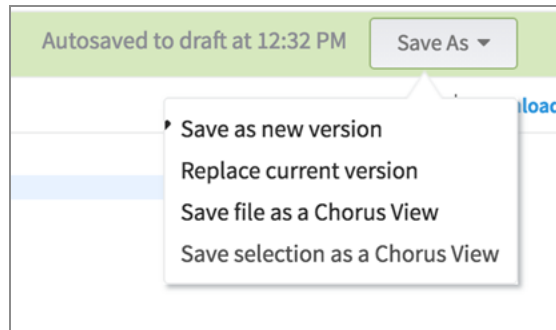
This saves a new version manually.

The following commands are available.

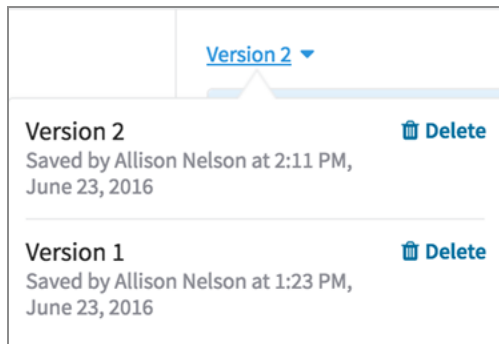
- **Save as new version:** Saves a new version of the file in the versioning list. This

does not affect previous changes/versions.

- **Replace current version:** Replaces the current version you are working on.
- **Save file as a Team Studio View:** Saves the query/results as a TIBCO Data Science - Team Studio view. TIBCO Data Science - Team Studio views can be used in analytic workflows just like database tables.
- **Save selection as a Team Studio View:** Saves a selection of text as a TIBCO Data Science - Team Studio view.



2. To review the previous versions of an SQL work file, click the **Version** link, and then choose the revision that interests you. You can then edit and run that version of the file without disrupting the most recent version.



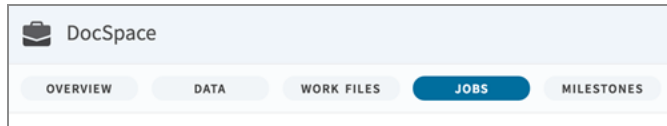
You can also delete previous versions of the work file. When code is generally useful, your team members can make copies of it in their workspaces to use as their own starting points.

## Procedure

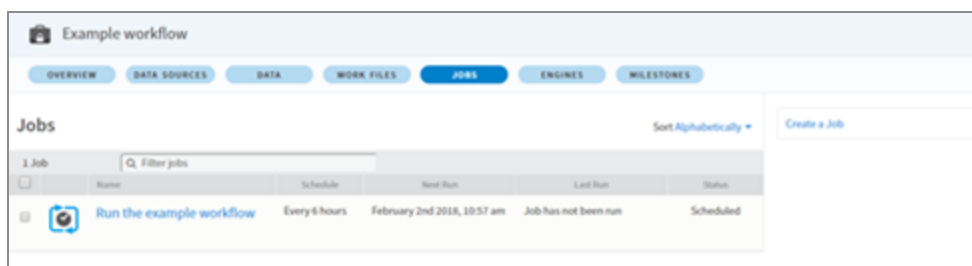
From the work file actions menu, click **Copy**. For more information, see [Copying a Work File to a Workspace](#).

## Jobs Tab

The **Jobs** tab displays a list of jobs in the workspace. A job is an executable process that contains a number of tasks, such as importing data or executing a workflow. Using the **Jobs** tab, you can create, edit, and run jobs.



## The Jobs List



Each entry in the jobs list shows the job's schedule, when it is scheduled to run next, and when it ran last. By default, the job list is sorted in alphabetical order, but you can change the sort order to show the jobs that runs next.

You can schedule jobs to run automatically on a periodic schedule, or on demand. A job that is set up to run on a schedule can be disabled, in which case it can only be run on demand until it is enabled again. The status of the job is displayed to the right of the entry. Jobs can be **On Demand**, **Scheduled**, **Running**, **Stopping**, or **Disabled**.

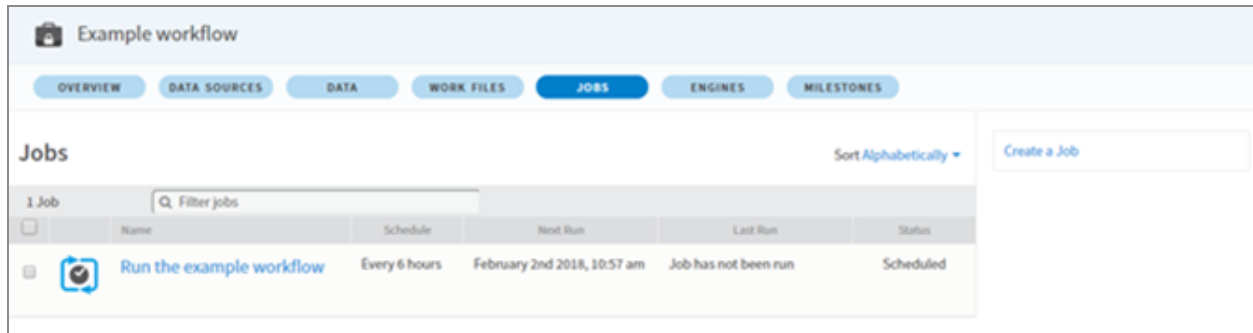
Using commands in the right panel, you can perform the following tasks:

- **Edit Job:** Update the job details, including schedule and notifications. To change the tasks in the job, navigate to the job's detail page by clicking on the job's name then select the task and 'Edit'.
- **Run Now:** Begin running the job immediately.
- **Stop:** Stop running a job that has started.
- **Enable or Disable:** Toggle the enabled status for the job between enabled and disabled. (This is only available for scheduled jobs).
- **Delete:** Delete the job from the workspace.

## Viewing the Jobs List

The **Jobs** tab displays a list of jobs in the workspace.

Click the **Jobs** tab in an open workspace to view the Jobs list.



Each entry in the jobs list shows the job's schedule, when it is scheduled to run next, and when it ran last. By default, the jobs list is sorted in alphabetical order, but you can change the sort order to place the jobs that runs next at the top.

You can schedule jobs to run automatically on a periodic schedule, or on demand. A job that is set up to run on a schedule can be disabled, in which case it can only be run on demand until it is enabled again. The status of the job is displayed to the right of the entry. Jobs can be On Demand, Scheduled, Running, Stopping, or Disabled.

Using the commands in the right panel, you can perform the following tasks:

- **Edit Job** - Update the job details, including schedule and notifications. To change the tasks in the job, navigate to the job's detail page by clicking on the job's name, selecting the task, and then selecting **Edit**.
- **Run Now** - Begin running the job immediately.
- **Stop** - Stop running a job that has started.
- **Enable** or **Disable** - Toggle the enabled status for the job between enabled and disabled. (This is available only for scheduled jobs).
- **Delete** - Delete the job from the workspace.

## Creating a Job

This topic describes the process of adding a job to a workspace.

Start in the main Jobs window.

## Procedure

### 1. Click **Create a Job**.

**CREATE A NEW JOB**

Name \*  
Run the example workflow

Description

**Schedule**  
Jobs can run on a recurring schedule or on demand

☐ On Demand  
☒ On a Schedule

Run Every \* 6 Hours

Next run date/time:  
07 05 2016 12 20 pm  
(GMT-08:00) Pacific Time (US & Canada)

☒ Disable this job on 07 06 2016

**Job Notifications**

Notify on success:  
☐ Nobody  
☐ Entire Workspace  
☒ Select People...

Notify on failure:  
☒ Nobody  
☐ Entire Workspace  
☐ Selected People

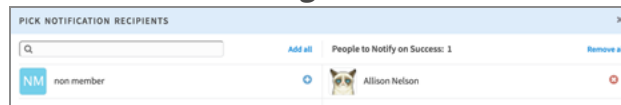
\* Required

Cancel Create

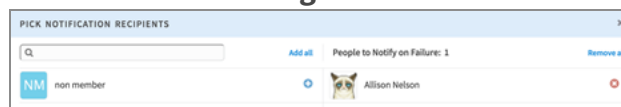
- **Name:** Choose a descriptive name for the job.
- **Description:** Optionally, add a description about the job.
- **Schedule:** Specify whether to run this job on demand or on a schedule. If you choose **On a Schedule**, this panel expands to include scheduling information.
  - **Run Every:** You can choose to run the job every  $n$  hours, days, weeks, or months, where  $n$  is an integer.
    - If **Weeks** is selected, checkboxes are available to choose the days of the week the job should run. You can select one or many days (for example, every Monday, Wednesday, and Friday).
    - If **Months** is selected, options are available to run the job on the defined day of the month (for example, the 5th of each month), day of the week (for example, the 2nd Monday of each month), or the last day of the month.
  - **Next run date or time:** Select when the job next runs. This is when the

schedule starts running. You can select days, times, and the time zone.

- **Disable this job on:** If you want the job to run only for a selected amount of time, choose when the schedule ends. You can select date and time.
- **Summary information:** Describes the schedule as entered above.
- **Job Notifications:** Specify whether to send notifications to people regarding the job's status.
  - **Notify on success:** If the job succeeds, the specified people are notified of the job's status and results.
    - **Nobody:** No one receives notifications. You must return to the job tab in this workspace to see the job's status and results.
    - **Entire Workspace:** Everyone in the workspace receives a TIBCO Data Science - Team Studio notification and an email notification.
    - **Selected People:** If you choose this option, the **Select People** link is displayed. Click this link to display the **Pick Notification Recipients** dialog, in which you can choose people to notify. Click + next to a person's name to add that person to the recipient list, and then click **Save Changes**.



- **Notify on failure:** If the job fails, the specified people are notified of the job's status and results.
  - **Nobody:** No one receives notifications. You must return to the job tab in this workspace to see the job's status.
  - **Entire Workspace:** Everyone in this workspace receives a TIBCO Data Science - Team Studio notification and an email notification.
  - **Selected People:** If you choose this option, the **Select People** link is displayed. Click this link to display the **Pick Notification Recipients** dialog, in which you can choose people to notify. Click + next to a person's name to add that person to the recipient list, and then click **Save Changes**.



2. When you are finished configuring the job details, click **Create**.

The job detail screen is displayed. Here you can add or edit tasks for your job. If you choose a scheduled job, TIBCO Data Science - Team Studio creates the job in a disabled state so you can make changes to the job and not worry about it starting without your knowledge. When you are ready to start the job schedule, you can enable the job.



## Adding Tasks to a Job

A job can run four different types of tasks.

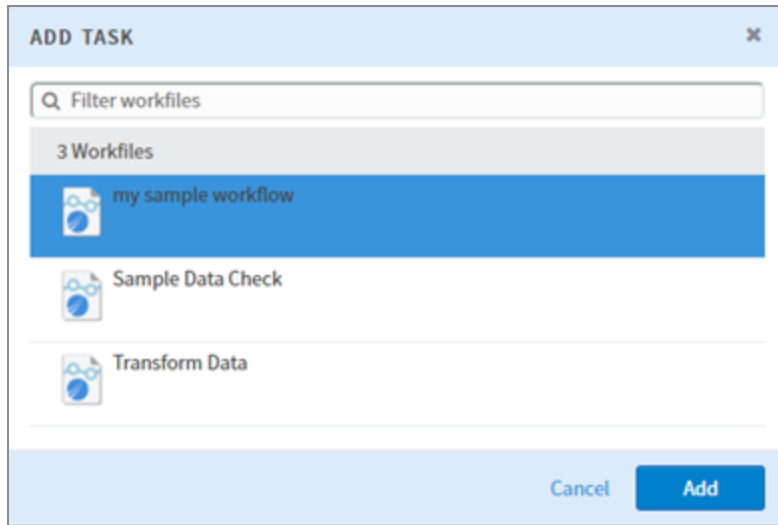
- **Import Source Data:** Use this task to import data into your workspace sandbox. This is useful if you want to import data into your sandbox on a recurring basis.
- **Run a Workflow:** Use this task to run a [workflow](#) in the workspace.
- **Run a SQL Work File:** Use this task to run a [SQL workflow](#).
- **Run a Python Notebook:** Use this task to run a [Python Notebook](#).

### Procedure

1. Click **Add Task** or select one of the listed tasks to get started.
2. Choose a task type.

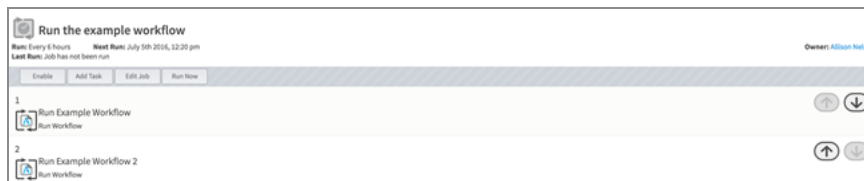
For example, for a workflow, a dialog is displayed that shows the workflows you have access to in this workspace. Select the workflow to run. If you want to run more than one workflow in succession, create another task for it within the job.





You can add several tasks to a job. They are run synchronously.

3. To change the sequence of the tasks, use the up and down arrows to the right of the task list.



## Running a Job

You can run a job from the job detail page or from the job list.

From the job detail page or the job list, click **Run Now**.

The status of the job changes to **Running**. If you choose to receive notifications, a TIBCO Data Science - Team Studio notification is displayed when the job finishes. You also get an email, based on your notification preferences. Otherwise, the **Last Run** column of the job list displays a link to the job's results when it is complete. Additionally, a new entry appears in the **Activity** section of the workspace. Click the last run time to see a summary and the job results.

RUN DETAILS: WORKFLOW				
Task Name	Started	Finished	Duration (hh:mm:ss)	Status
Run Example Workflow 2	July 5th 2016, 11:55 am	July 5th 2016, 11:56 am	00:00:45	<a href="#">Workflow Result</a>
Job Summary	July 5th 2016, 11:55 am	July 5th 2016, 11:56 am	00:00:45	✔ Job ran successfully

Close Window

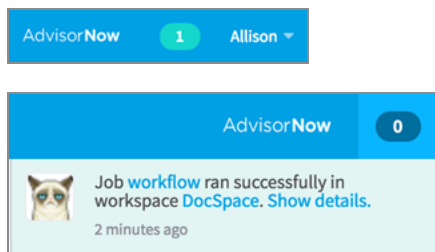
If the job fails, error information is displayed in this dialog.

**i Note:** Any currently running jobs are aborted if the TIBCO Data Science - Team Studio server is restarted.

## Viewing Job Results

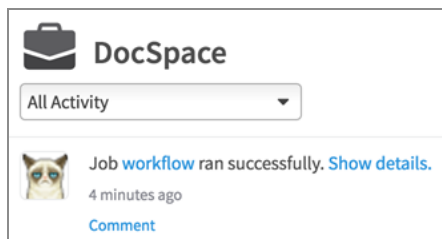
After the job is completed, use one of the following methods to view job results.

- If you have notifications turned on, you see a TIBCO Data Science - Team Studio notification when a job finishes, based on your settings.

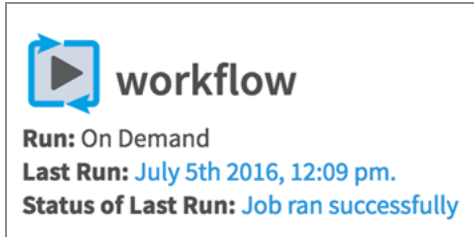


To show the run details dialog, click **Show details**, where you can view the workflow result.

- The job status also shows in the activity section of your workspace. This can be found on the [Overview Tab](#).



- You can view your job results from the job detail page itself by clicking the **Last Run** time.



## Milestones Tab

The **Milestones** tab is where you create milestones - targeted tasks that must be completed within a workspace by a certain date - to track progress on your analytics projects.

When you select the **Milestones** tab, a list of milestones is displayed, sorted by target date.

Milestones		
2 Milestones		
Name	Target Date	Status
 Do the thing!	2016-06-30	Completed
 Do the other thing	2016-07-16	Planned

Each milestone has the following attributes:

- **Name**
- **Target date**
- **Status: Planned or Completed**

For more information, see [Creating a Milestone](#).

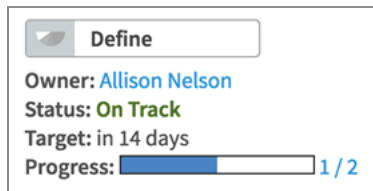
The following actions are available for each milestone:

- **Edit** - Choose a new name or target date for the milestone.
- **Delete** - Removes the milestone from the workspace.
- **Complete** - Marks the milestone as completed. The milestone icon changes to confirm completion.

A completed milestone has the following action:

- **Restart** - Toggles the completion status of the milestone.

You can see the progress on your team's milestones from the workspace [Overview Tab](#).



The **Target** shows when the next milestone's target date is. The **Progress** indicator shows how many milestones have been completed. In this example, one out of two milestones are completed, and the next target is in 14 days. Click the number beside the progress bar to navigate to the milestone list.

## Viewing a Milestone

When you select the **Milestones** section, a list of milestones is displayed, sorted by target date.

Click the **Milestones** tab. to view the details.

Milestones		
2 Milestones		
Name	Target Date	Status
Do the thing!	2016-06-30	Completed
Do the other thing	2016-07-16	Planned

Each milestone has the following attributes:

- Name
- Target date
- Status: Planned or Completed

## Creating a Milestone

Use this procedure to configure a new milestone.

CREATE A NEW MILESTONE

Name \*

Prepare Report

Target date of the milestone \*

07 01 2016

\* Required

Cancel

Create

## Procedure

1. Click **Create New Milestone**.
2. Provide a name for the milestone.
3. Specify a target date for the milestone.

This date represents when you want to complete this unit of work.

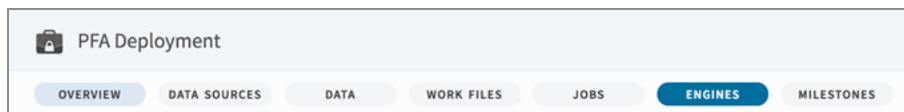
- a. Click the calendar icon to access the date picker widget.

4. Click **Create**.



The milestone is added to the list and sorted by target date.

## Engines Tab

The **Engines** tab provides a place to create, edit, govern, and test engines - real-time services that perform single-sample scoring using a model description in a PFA document.



Each entry in the engines list shows the name of the engine, its source PFA document, its governance stage, and whether it is deployed. The engine list is sorted in alphabetical order, and you can filter it using the search bar at the top.

Engines					
2 Engines		Filter engines by name, owner, approver, or stage			
<input type="checkbox"/>	Name	Source Document	Stage	Status	Tags
<input type="checkbox"/>	 Credit Default Risk	<a href="#">Credit Default Risk.pfa</a>	Proposed	Not deployed	
<input type="checkbox"/>	 Credit Default Risk (Champion)	<a href="#">Credit Default Risk.pfa</a> Deployed version: 1	Approved	Deployed	

Engines have three governance stages that influence the system administrator's ability to deploy them to production deployment targets. Any engine can be deployed to development deployment targets. For more information, see the topic "Deployment Targets" in *TIBCO® Data Science - Team Studio System Requirements*.

Engines start in the draft stage, where they can be proposed. The system administrator can approve the engine or reject the engine, moving it to the approved stage or back to the draft stage, respectively. Only approved engines can be deployed to production deployment targets.

Using commands in the right-hand panel, you can perform the following tasks.

- **Deploy Engine:** Push the current version of the engine to its deployment target.
- **Undeploy Engine:** Undeploy the engine from its deployment target.
- **Test Engine:** Open a dialog to send the engine test data and view its responses.
- **Propose Engine:** Move the engine from the draft stage to proposed.
- **Approve Engine:** Move the engine from the proposed stage to approved.
- **Reject Engine:** Move the engine from the proposed stage to draft.
- **Edit Engine:** Change the name, description, PFA document, deployment target, or the name of the administrator who can approve the engine.
- **Delete Engine:** Delete the engine. This also undeploys the engine if it is deployed.

These actions might be unavailable if the engine is not in the appropriate state. For example, you cannot undeploy an engine that is not already deployed, and you cannot approve an engine that is in the draft stage.

## Viewing the Engines List

The **Engines** tab displays a list of engines in the workspace.

Click the **Engines** tab.

Engines					
2 Engines		Filter engines by name, owner, approver, or stage			
	Name	Source Document	Stage	Status	Tags
<input type="checkbox"/>	Credit Default Risk	<a href="#">Credit Default Risk.pfa</a>	Proposed	Not deployed	
<input type="checkbox"/>	Credit Default Risk (Champion)	<a href="#">Credit Default Risk.pfa</a> Deployed version: 1	Approved	Deployed	

Each entry in the engines list shows the name of the engine, its source PFA document, its governance stage, and whether it is deployed. The engines list is sorted in alphabetical order, and you can filter it using the search bar at the top.

Engines have three governance stages that influence your ability to deploy them to production (see "Deployment Targets" in *TIBCO® Data Science - Team Studio Installation and Administration*).

(Any engine can be deployed to development deployment targets.) Engines start in the draft stage, where they can be proposed. The engine approver either approves or rejects

the engine, moving it to the approved stage or back to the draft stage, respectively. Only approved engines can be deployed to production deployment targets.

Using commands in the right panel, you can perform the following tasks:

- **Deploy Engine** - Push the current version of the engine to its deployment target.
- **Undeploy Engine** - Undeploy the engine from its deployment target.
- **Test Engine** - Open a dialog to send the engine test data and view its responses.
- **Propose Engine** - Move the engine from the draft stage to proposed stage.
- **Approve Engine** - Move the engine from the proposed stage to approved.
- **Reject Engine** - Move the engine from the proposed stage to draft.
- **Edit Engine** - Change the engine's name, description, PFA document, deployment target, or approver.
- **Delete Engine** - Delete the engine. This also undeploys the engine if it is deployed.



**Note:** These actions might be disabled or unavailable if the engine is not in the appropriate state. For example, you cannot undeploy an engine that is not already deployed, and you cannot approve an engine that is in the draft stage.

## Creating an Engine

Follow this procedure to create an engine.

### Procedure

1. To start the process of adding an engine to the workspace, in the side panel, click **Create an Engine**.

2. Fill out the dialog as follows:

- **Name:** Specify a descriptive name for the engine.
- **Description:** Optionally, add a description about the engine.
- **PFA Document:** Specify the PFA document from the workspace that you want the engine to use for real-time scoring.
- **Deployment Target:** Specify the server to deploy the engine to.
- **Approver:** Specify which TIBCO Data Science - Team Studio user to notify to approve or reject the engine when it is proposed.

3. Click **Create Engine**.

The engine starts undeployed and in the draft governance stage.

## Deploying and Governing an Engine

You can deploy an engine if it has a development deployment target, or if it is approved for a production deployment target.

### Procedure

1. To deploy an engine, select **Deploy Engine** from the side panel.

Once the engine is deployed, its **Details** tab in the side panel shows which version of the PFA document is deployed, the deployment target name and environment, and the URL of the deployed engine.



To undeploy the engine, select **Undeploy Engine** from the side panel.

ACTIVITY	DETAILS
<b>Source Document</b> Credit Default Risk.pfa Deployed version: 1	
<b>Deployment Target</b> Dev Development	
<b>Deployed Engine URL</b> http://localhost:8981/engine/266f3fa4-4af5-42e5-8e29-89375f6c71e1	
<b>Owner</b> Chorus Admin	
<b>Approver</b> Chorus Admin	
<b>Last Modified</b> March 17th 2017, 11:29 am	

2. To govern an engine, use the **Propose**, **Approve**, and **Reject** actions in the side panel. When you propose an engine, it notifies the assigned approver using an in-app notification as well as email (if the approver has email notifications enabled).

## Testing an Engine

Use this procedure to test an engine.

### Procedure

1. To submit arbitrary data to the scoring engine, click **Test Engine**, and then provide the data in the resulting dialog.

This data should be formatted as JSON (one JSON object per line) or CSV. If you use CSV, enclose headers and surround string fields with double-quotation marks. If you want white space in the engine input, include spaces only around the commas.

TEST ENGINE

Place a CSV or newline-separated JSON object representation of the samples you'd like to score in the input box, then click Test. If using CSV, please include headers and surround string fields with double-quotes. Engine output will appear in the output box below.

```

revolving_util,credit_lines,monthly_income,times30dayslate_2years
0.5,10,5000,2
0.5,10,5000,0
0.5,10,5000,1
0.5,10,3000,0

```

Test

POSTed input to <http://localhost:8981/engine/266f3fa4-4af5-42e5-8e29-89375f6c71e1/instance/fcdb3537-1a3a-4b49-b0ec-51ed2a529038/action>

Engine response:  
 PRED,CONF,INFO  
 "NO",0.9623565162978394,{"YES":0.037643483702160675,"NO":0.9623565162978394}  
 "NO",0.983536315306315,{"YES":0.016463684693684973,"NO":0.983536315306315}  
 "NO",0.9750499123288003,{"YES":0.02495008767119968,"NO":0.9750499123288003}

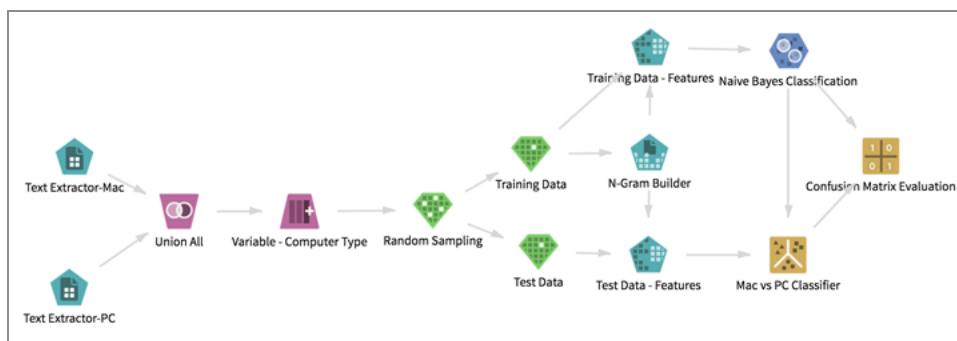
Close Window

- With the input in place, click **Test** to send it to the deployed engine.

The engine response is displayed in the text box below, including the URL where the input was sent.

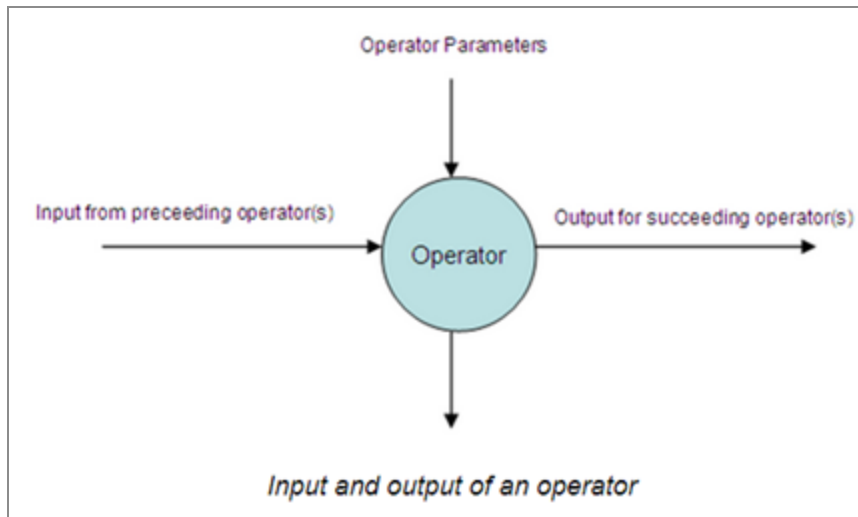
## Workflows

Model your analytics process with a directed graph of operators called a workflow. Workflows combine data sources and operators (algorithms and transformations).



Each operator takes input from the preceding operator(s) or data source(s), performs its task, and produces output for the succeeding operators. Operators have parameters that can be configured by the user. These parameters can affect everything from the columns to analyze to the Spark settings to fine-tune your job's performance. The application performs validation checks when it creates connections between operators and configures their parameters.

At runtime, information produced from an operator is passed to its succeeding operator(s). This forms an information flow through the operators that can be inspected and analyzed at any intermediate point.



When you run a workflow, TIBCO Data Science - Team Studio checks the inputs and any needed dependencies on each operator as execution reaches that point in the graph. If required parameters are missing, the workflow does not start and errors are reported. Additionally, the title of the offending operator is displayed in red.

When an operator finishes its task, TIBCO Data Science - Team Studio gets any output and displays it as a section in the workflow result screen.

## Supported Workflows

The TIBCO® Data Science - Team Studio version 7.1.0 supports the legacy workflows. In addition to the legacy workflows, TIBCO Data Science - Team Studio also introduces a new type of workflow - **New Workflow**. Some of the features of a New Workflow are:

- Supports only the TIBCO® Data Virtualization data sources.
- Runs or executes only in supported Apache Spark 3.2 or later cluster.
- Have a subset of available operators.
- Compatible with TIBCO® Data Virtualization data source and Apache Spark operators. For more information on supported operators, see [Apache Spark specific operators](#).
- By default, the workflows are **Fused**, such that, the entire workflow is executed as a

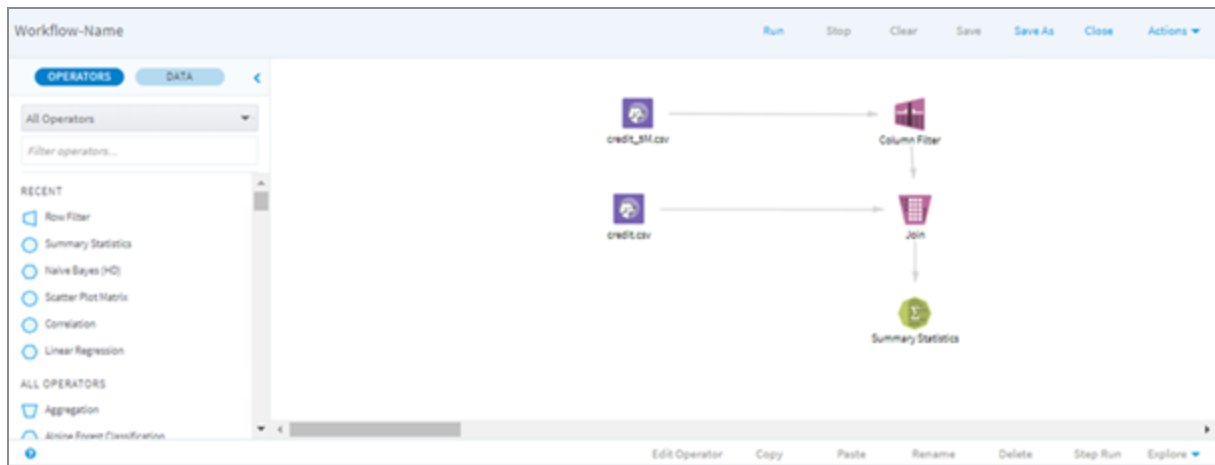
single Spark application which increases the efficiency.

- The **Convert to Spark** menu option is not available for these workflows, since they are executed in a single Spark application.
- The execution of notebooks within a workflow is currently not supported.
- Currently, the workflow does not support the Custom Operators (MODs) that were previously developed for the legacy workflows.

## Workflow Editor

Workflows - collections of operators and data - are created and edited in the TIBCO Data Science - Team Studio workflow editor.

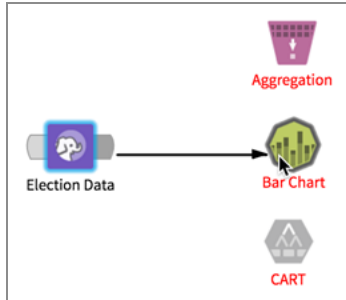
Create workflows within the editor by dragging data and operators onto the canvas. You can move elements around and manipulate them with the mouse.



Operators and data are connected with arrows. To connect a data source to an operator, hover your mouse over the data source. Notice that two tabs appear on either side of the data icon.



To create an arrow, click a tab and drag your mouse. Release the arrow over the operator you want to connect to.



Operators that you can connect to remain in color; operators that you cannot connect to directly are grayed out.

For a more detailed description of operators and how to use them, see [Operator Actions](#).

## Workflow Actions

The following workflow actions are available from the **Actions** dropdown menu.

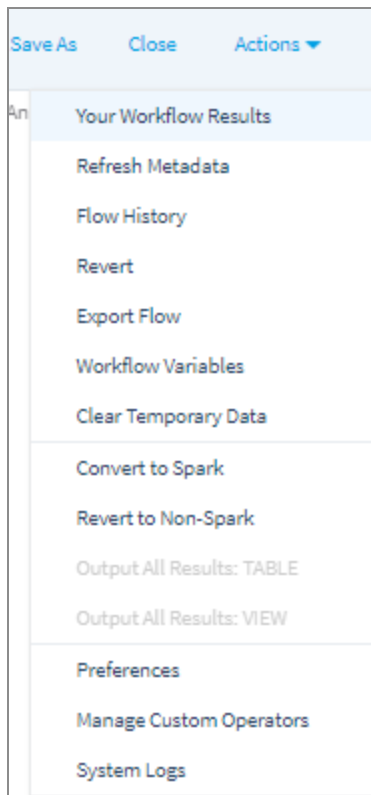
## Your Workflow Results

To view results of previously run flows, use the **Your Workflow Results** command.

### Before you begin

To access the **Actions** menu, you must have a workflow open on the workflow canvas.

From the **Actions** dropdown menu, click **Your Workflow Results**.



## Result

Workflows run by the current user are displayed along with the workflow version, start time, and end time.

Your Workflow Results				
Flow Name Filter: * ▼				
Flow Name	Run Type	Version	Start Time ▼	End Time
Example_Association_Rules	Manual	2	2017-05-23 16:54:26	2017-05-23 16:58:25
MADlib Custom Operators	Manual	5	2017-04-11 14:35:37	2017-04-11 14:35:45
MADlib Custom Operators	Manual	2	2017-04-11 14:00:26	2017-04-11 14:00:40
madlib	Manual	3	2017-02-08 16:12:05	2017-02-08 16:12:25
madlib	Manual	3	2017-02-08 16:08:21	2017-02-08 16:08:21
stuff	Manual	3	2017-02-06 10:25:25	2017-02-06 10:25:45
madlib	Manual	3	2017-02-03 15:33:06	2017-02-03 15:33:06
madlib	Manual	2	2017-01-24 16:18:23	2017-01-24 16:18:32
madlib	Manual	2	2017-01-24 16:15:19	2017-01-24 16:15:27

\* indicates current unsaved flow result

Done Delete Show Result

- To view the results of a workflow in a new window, from the Your Workflow Results

dialog, select the workflow, and then click **Show Result**.

- To remove the results from the Your Workflow Results dialog, select a workflow and click **Delete**.

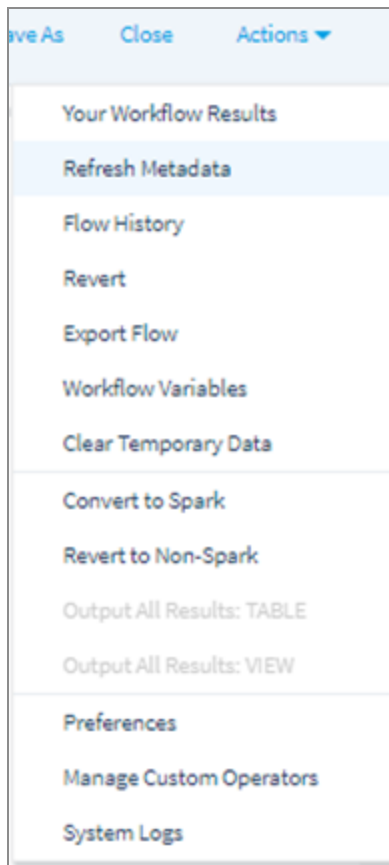
## Refresh Metadata

Use the **Refresh Metadata** command to manually refresh the metadata for a database connection.

### Before you begin

To access the **Actions** menu, you must have a workflow open on the workflow canvas.

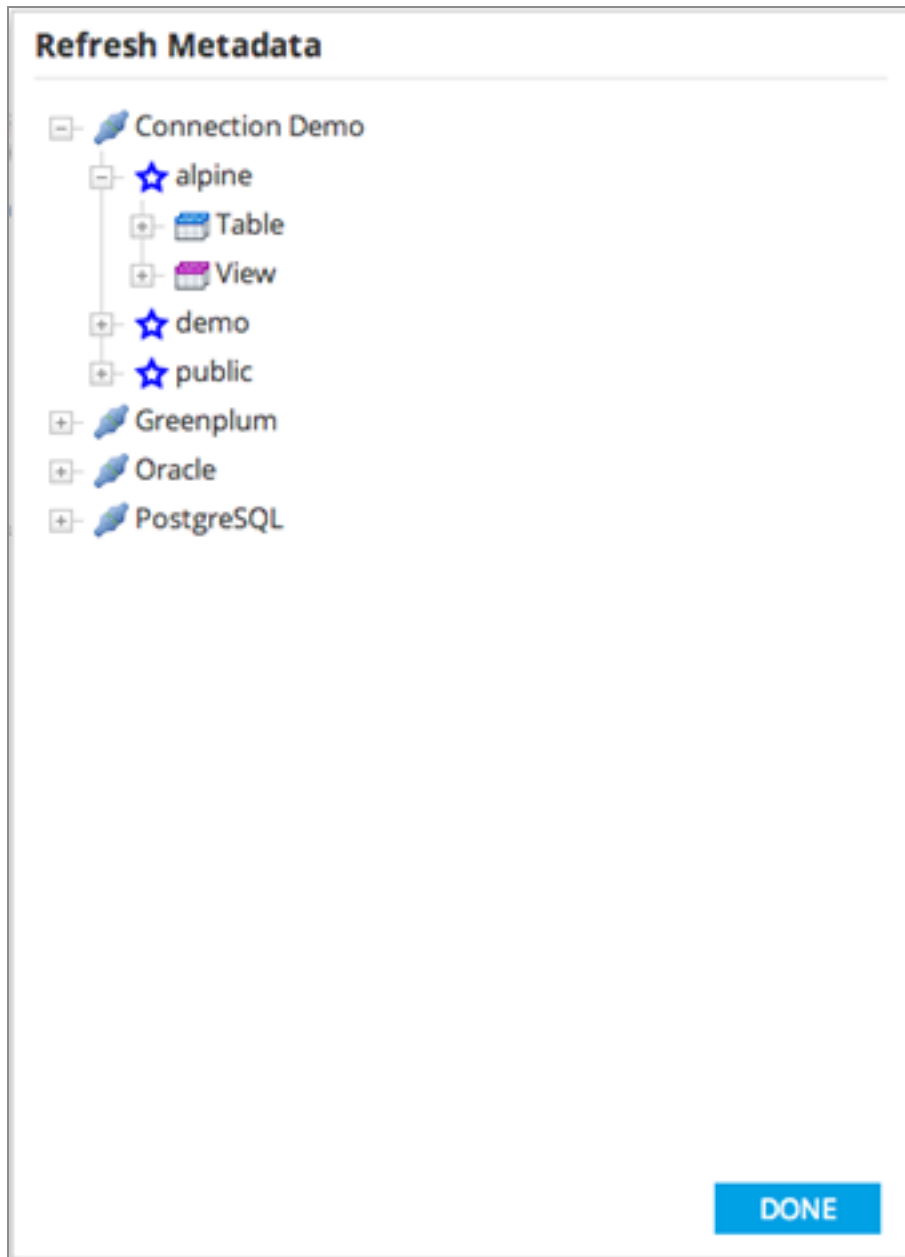
From the **Actions** dropdown menu, click **Refresh Metadata**.



### Result

The Refresh Metadata dialog is displayed.

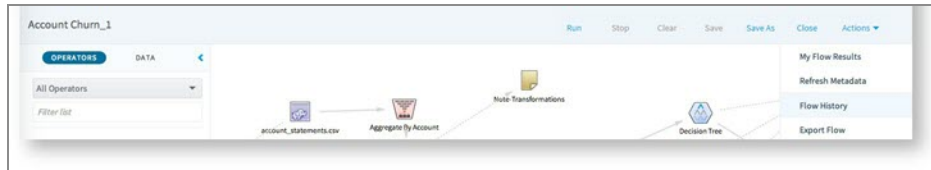
- To reload the metadata to the next level of the tree, right-click a connection and choose **refresh**.
- To view more specific components of the connection, expand the tree.
- To refresh the metadata for specific components (for example, all tables or a specific table), right-click and choose refresh on child nodes





## Flow History

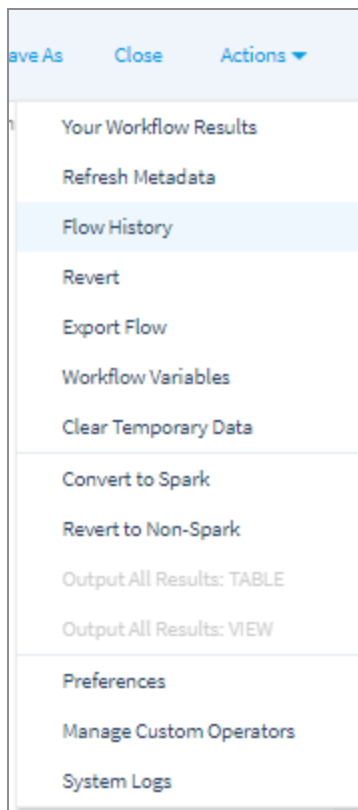
The **Flow History** command provides a chronological summary of the version history of a workflow within a workspace.



### Before you begin

To access the **Actions** menu, you must have a workflow open on the workflow canvas.

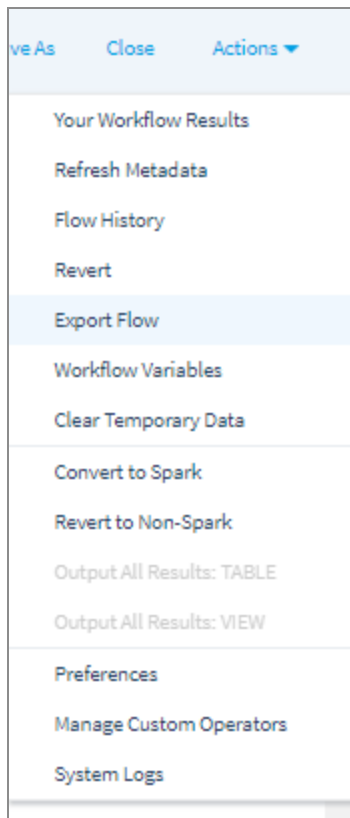
From the **Actions** dropdown menu, click **Flow History**.



### Result

The Flow History dialog is displayed.





## Result

The workflow is downloaded as an .afm file.

# Workflow Variables

The **Workflow Variables** command enables you to create and edit workflow variables.

## Before you begin

To access the **Actions** menu, you must have a workflow open on the workflow canvas.

From the **Actions** dropdown menu, click **Workflow Variables**.

## Result

The Workflow Variable Settings dialog is displayed.

## Clear Temporary Data

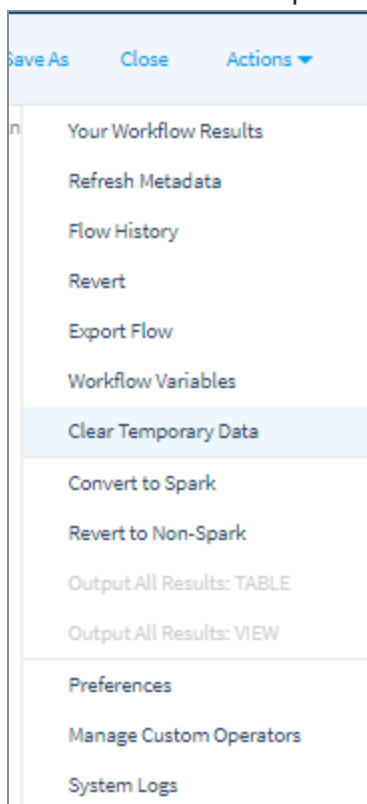
You can remove temporary tables, views, and files (those created by TIBCO Data Science - Team Studio operators) in a workflow.

### Before you begin

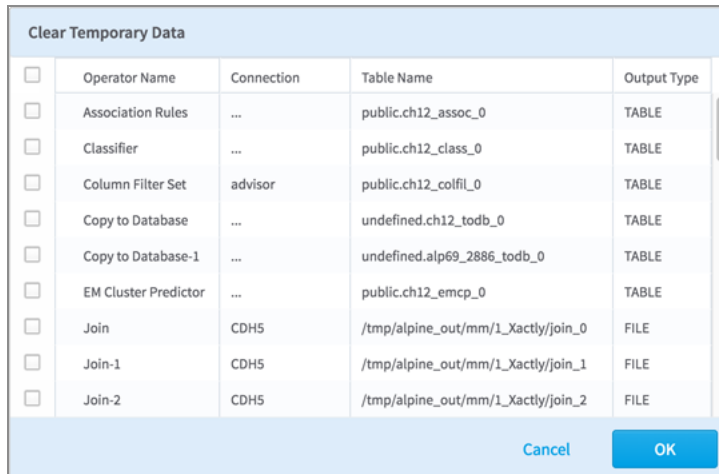
To access the **Actions** menu, you must have a workflow open on the workflow canvas.

### Procedure

1. From the **Actions** dropdown menu, click **Clear Temporary Data**.



The Clear Temporary Data dialog opens. Each table, view, or file currently configured for creation in the open workflow is displayed in the dialog.



- You can specify individual rows from the **Clear Runtime Directories** table, the **Clear Individual Operator Results** table, or both. Select the checkbox next to the item to clear it.
  - You can specify all rows in the **Clear Runtime Directories** table, the **Clear Individual Operator Results** table, or both. Select the checkbox next to the column header to select all rows in the table.
2. Click **OK** to clear all selected items, or click **Cancel** to close the dialog without dropping anything.

Clear Temporary Data

Clear Runtime Directories

	Connection	Folder type	Size
<input checked="" type="checkbox"/>	CDH57-1.1.1.1	HDFS Runtime Directory	151 KB
<input checked="" type="checkbox"/>	CDH57-1.1.1.1	HDFS Output Directory	100 KB

Clear Individual Operator Results

	Operator Name	Connection	Table Name	Output Type
<input checked="" type="checkbox"/>	Summary Statistics	CDH57-1.1.1.1	...	File
<input checked="" type="checkbox"/>	High turnout filter	CDH57-1.1.1.1	...	File
<input checked="" type="checkbox"/>	Column Filter	CDH57-1.1.1.1	/tmp/tsds_out/.../KHR_Test_1_3356/colfil	File
<input checked="" type="checkbox"/>	Correlation	CDH57-1.1.1.1	...	File
<input checked="" type="checkbox"/>	Random Sampling	CDH57-1.1.1.1	...	File
<input checked="" type="checkbox"/>	Predictor - Turnout	CDH57-1.1.1.1	...	File
<input checked="" type="checkbox"/>	Regression Evaluator (HD)	CDH57-1.1.1.1	/tmp/tsds_out/.../KHR_Test_1_3356/Regre	File
<input checked="" type="checkbox"/>	Variable	CDH57-1.1.1.1	/tmp/tsds_out/.../KHR_Test_1_3356/var_C	File

Cancel

OK

## Convert to Spark/Revert to Non-Spark

If you have a workflow that you created before version 6.4, and if operators in the workflow now can use Spark (node fusion operators), you can set the operators all to use Spark, without setting the option individually for each operator. Likewise, if you want to revert a workflow to its pre-version 6.4 status, where no operator uses Spark, you can do that without setting the option individually for each operator.

By default, existing workflow operators have the option to use Spark set to **no**, which ensures no upgrade impact. New operators in existing or new workflows have the default for **Use Spark** set to **yes**.

## Before you begin

To access the **Actions** menu, you must have a workflow open on the workflow canvas.

You must have updated your workflows to 6.4 or later.

- To update all of the available node fusion operators to **Use Spark = yes**:  
From the **Actions** dropdown menu, click **Convert to Spark**. When you set this option, you are ready to run your existing workflow using Spark Node Fusion.

**Note:** For the [Row Filter \(DB\)](#) and [Variable \(DB\)](#) operators, advanced syntax uses Spark SQL, rather than Pig, if **Use Spark** is set to **yes**.

- To revert to non-Spark versions of all operators:  
From the **Actions** dropdown menu, click **Revert to Non-Spark**. Doing so reverts the workflow to its pre-6.4 state.

## Output All Results: Table or View

TIBCO Data Science - Team Studio can use a lot of disk space. During development, it can be helpful to store intermediate tables for easy review. But when you move to a workflow production, changing operators to output type **View** can be the most efficient for big data storage.

But you do not need to open each operator and set this option; instead, you can use an action to make a bulk change, switching tables to views, or if needed, back to tables.



## Before you begin

To access the **Actions** menu, you must have a workflow open on the workflow canvas.

- To output the results as a View:  
From the **Actions** dropdown menu, click **Output All Results: VIEW**.

- To output the results as Table:  
From the **Actions** dropdown menu, click **Output All Results: TABLE**.

## Preferences

The **Preferences** command enables you to edit a variety of workflow editor preferences.

### Before you begin

To access the **Actions** menu, you must have a workflow open on the workflow canvas.

From the **Actions** dropdown menu, click **Preferences**.

### Result

The Edit Preferences dialog is displayed..

For more information, see "Workflow Editor Preferences" in *TIBCO® Data Science - Team Studio Installation and Administration*.

## Manage Custom Operators

The **Manage Custom Operators** command enables you to upload new custom operators and manage existing ones.

### Before you begin

To access the **Actions** menu, you must have a workflow open on the workflow canvas.

You must be an application administrator to upload custom operators.

From the **Actions** dropdown menu, click **Manage Custom Operators**.

### Result

Your operator(s) are displayed in the operator list. Everyone with permissions to edit workflows can use them in their workflows.



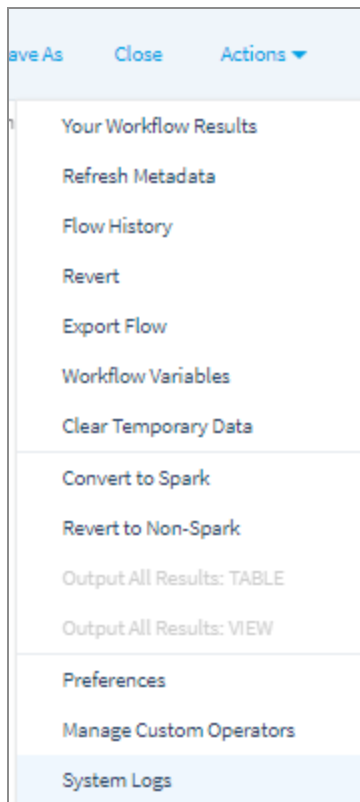
## System Logs

The system logs provide easy UI access to the back-end server logs. This can be helpful for debugging workflows.

### Before you begin

To access the **Actions** menu, you must have a workflow open on the workflow canvas.

From the **Actions** dropdown menu, click **System Logs**.



### Result

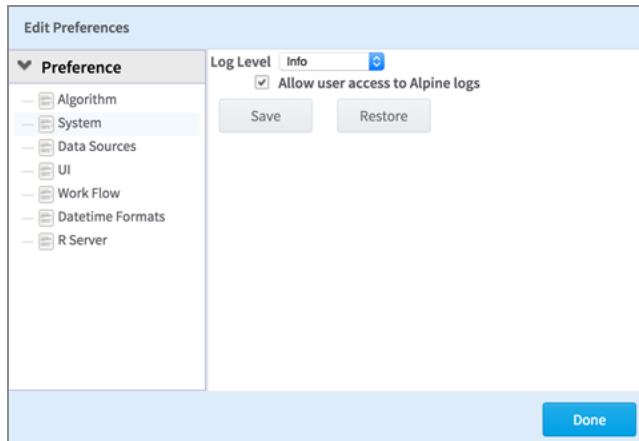
The system Logs dialog is displayed.

**Note:** Alpine.log is also tagged with the user's name to assist with debugging and understanding system usage.

- To retrieve the latest updates to the logs, click **Refresh**. An administrator can always

view the system logs; however, an ordinary user can only view the logs if the application preferences are set to allow viewing by all users.

- To provide access to all users, an administrator enables the option by clicking **Actions > Preferences**. The administrator can also set the **Log Level** at which logging is viewed.



## Workflow Menu

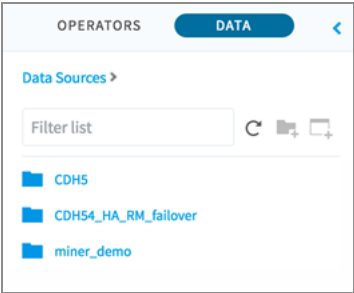
The menu for the Workflow canvas contains the controls to run, stop, clear, save, and revert changes to the workflow, as well as other actions.

Menu option	Description
<b>Run</b>	Runs the entire workflow. As your workflow runs, progress messages are displayed in the Results pane. When the workflow is finished, click on any operator to see its output. For more information, see >>Running a Workflow<<.
<b>Stop</b>	Stops a running workflow. Administrators can also stop running workflows from the Administrator Console. See the >>Administrator Console<< help for more information on this option.
<b>Clear</b>	Clears any cached results about the workflow. Use this option first to remove all previously cached results if you have selected the <b>Step Run</b> option on this workflow previously, and you want to rerun everything from scratch. Clearing the results is also useful for troubleshooting: modified operators do not always

Menu option	Description
	match their step run results.
<b>Save</b>	Displays a dialog where you can provide a comment, and then click <b>Save</b> to save this version of your workflow. This version is displayed in your workflow history and can be restored. Saving the workflow also updates the preview image of the workflow.
<b>Save As</b>	<p>Displays a dialog where you can save a version of the open workflow with a different name, and without overwriting the existing workflow.</p> <p>The default value for <b>Workflow Name</b> is appended with _1. Provide a useful name, and optionally a description. The original workflow is reverted to its previous state and closed. The new workflow is always saved to the workspace with the original.</p>
<b>Revert</b>	<p>Displays a dialog in which you can revert any changes you have made since your last save. In the dialog, click <b>OK</b> to reset the workflow to its last saved state.</p> <p><b>Note:</b> If you want to further revert your workflows, then from the <b>Actions</b> dropdown list, click <b>Flow History</b>.</p>
<b>Actions</b>	Displays a dropdown list of actions appropriate to the specific role for this workflow. That is, the available options can change depending on your application role. You can learn more about the workflow actions menu at >>Workflow Actions<<.




## Data Explorer

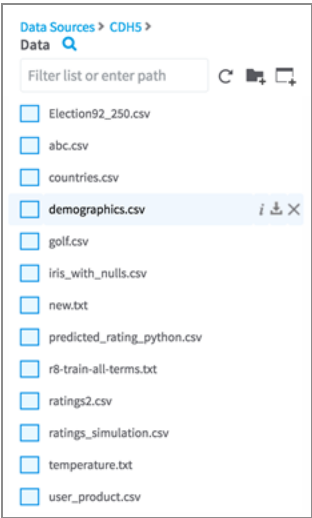
The data explorer shows data sources you have connected to the selected workflow. There is no need to know whether they are Hadoop data sources or databases: they are all browsable from this dialog.



If you have added multiple data sources to your workflow, you see a list of them first. After selecting a data source, you can navigate the file system/tables and select data to use in your workflow. To add data to a workflow, drag it over to the workflow canvas.

To navigate to a specific file, enter a search term or the file path in the search box.

Icon	Description
	The circular arrow refreshes the data source listing.
	The folder icon with a plus sign adds a folder at the current location in the file hierarchy.
	The rectangle with a plus sign imports a dataset from your local file system.



When you select a file, note that several other options become available.

 The "i" icon shows information about that file.

For datasets on HDFS, the information looks like the following.

Hadoop File Properties	
Name	credit.csv
Owner	mapred
Group	supergroup
Last modified time	Oct 26, 2015 1:48:28 PM
Access time	Jul 8, 2016 2:02:16 PM
File size	1.78MB
Block size	128.0MB
Permissions	rw-r--r--
<div>Done</div>	

For database datasets, the information displayed shows the data type and name of each of the columns.

Table Columns	
Column Name	Column Type
id	BIGINT
times90dayslate	INTEGER
revolving_util	NUMERIC
debt_ratio	NUMERIC
credit_lines	INTEGER
monthly_income	NUMERIC
times30dayslate_2years	INTEGER
srsdlqncy	INTEGER
<div>Done</div>	




The download arrow icon downloads the selected file to your local file system.

For datasets on HDFS, you can define which parts of the file or the entire file to download.

Download Hadoop File	
Name	credit.csv
Owner	mapred
Group	supergroup
File size	1.78MB
Download Entire Contents	<input type="checkbox"/>
Start at line	<input type="text" value="1"/>
Number of lines	<input type="text"/>
<div> <span>Close</span> <span>Download</span> </div>	

For datasets on a database, define how many lines to download. If you do not specify a number, the entire table is downloaded.

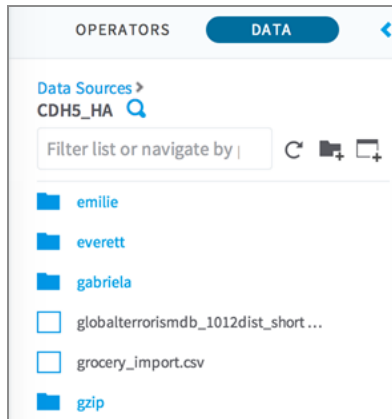
Download Results	
Name	credit
Number of lines	<input type="text"/>
<div> <span>Close</span> <span>Download</span> </div>	

 The x icon deletes the selected dataset. This option is only available on HDFS data sources.

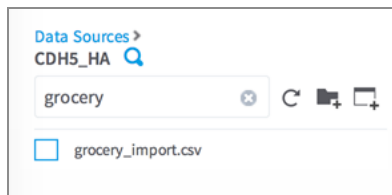
## Browse Hadoop Sources

For Hadoop sources, the data explorer navigates the directory hierarchy of HDFS. At each level, a blue link represents a directory or folder and a gray link represents a Hadoop file.

Select a directory to view the contents or drag a Hadoop file to an open workflow. Enter an expression or wildcards into the **Filter** text field to limit the datasets that are displayed.



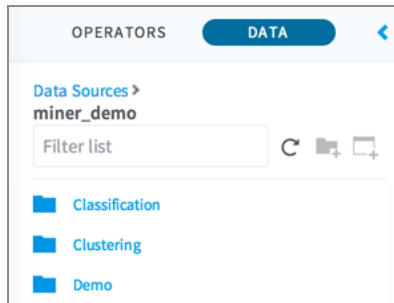
Within Hadoop data sources, you can also navigate by entering a path directly into the **Filter** field or the dialog that is invoked from the magnifying glass icon.



Paths that begin with a forward slash (/) are treated as absolute paths. Other paths are treated as relative paths. To aid in navigation, the header above the listing displays the current level path name. You can navigate back by using the breadcrumb control above the level name. The breadcrumb collapses as the hierarchy grows. Hover over the breadcrumb to expand the path.

## Browse Database Sources

For database sources, the second file level is the schema level. Each schema appears as a folder icon in the list. Clicking on a schema link navigates to the table/view level. At the table/view level, you can drag and drop data from the explorer to an open workflow.



**Note:** For Greenplum data sources, external tables can be viewed and used like any other Greenplum table.

## Data Sources

Each workspace provides access to data sources that are associated or available to your workspace.



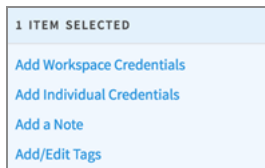
Databases and Hadoop clusters have their own sections in the list. Each row describes the data source's visibility. Data sources have properties that limit availability and can specify credentials. If you select a data source, you can see information about the credential settings and add your own, if the data source has been configured to accept separate credentials.

Availability level and credentials	Description
Public availability	Available to all workspaces.
Limited availability	Must be specifically associated with your workspace. See your data source administrator for assistance.
Workspace credentials	Every member of the workspace can access the data source using the same credentials.



Availability level and credentials	Description
Individual credentials	Only you (the user) can access the data source with credentials you provide. Other members of a workspace with access to this data source must provide their own credentials.

For example, the data source below has Public visibility and has been configured to accept Individual credentials. You can select it, and then provide Individual credentials.

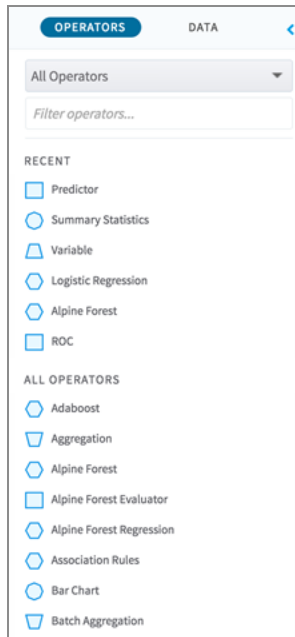


When you select a data source, you can see recent activity for that data source. You can also add notes or tags.

To edit a data source, you must navigate to the main **Data** section of the application. You can edit a data source only if you are a Data Administrator or higher.

## Operator Explorer

From the **Operators** explorer, you can filter and search through the list of available operators.



An operator is a unit of computation, usually an algorithm or a data transformation task. You can connect operators together to make an analytic workflow that showcases all parts of the analytics process, from transforming the data to predicting and scoring a model.

Recently-used operators are listed first in the operator list. You can search and filter for specific operators or types of operators.

- To search for a particular operator, type your search term in the **Filter operators** text box.
- To filter by category, click the dropdown list box. For example, to filter for only those operators that are compatible with Hadoop data sources, select **Hadoop-enabled** in the dropdown list box.

Users with Data Analyst or Analytics Developer roles can add operators to open workflows from the **Operators** tab of the Explorer. Depending on the role, different sets of operators are available. Analytics Developers can use all operators, while Data Analysts can perform more ETL-oriented tasks. For more information on the permissions, see *TIBCO Data Science - Team Studio Version and Licensing*.

## Operator Help

TIBCO Data Science - Team Studio offers four ways of getting help on an operator.

## Operator Tooltips

Each operator has a float-over tooltip that provides a basic overview of the operator. If you hold the cursor over the operator on the left-hand-side of the workflow editor, the tooltip note appears.

## Right-click Help

If you right-click on the operator in the workflow and choose Help from the menu, the Help topic associated with the operator opens.

## dialog Help

If you double-click an operator, the dialog associated with it opens on the right side of the screen. Click ? to open the Help topic associated with the operator.

## Help

If you select an operator on a workflow, you can click ? next to its name in the bottom-left corner of the screen.

# Creating a New Workflow

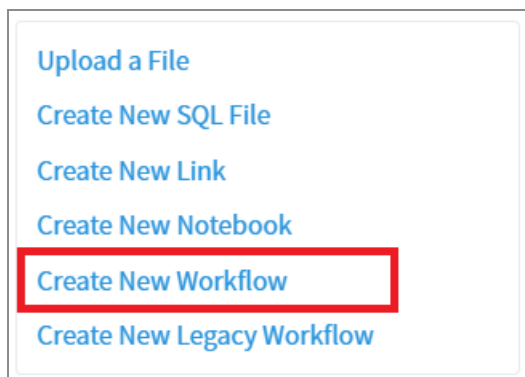
From the [Work Files](#) tab, users can create a new workflow within their workspace.

## Before you begin

You must have a workspace that you can write to.

## Procedure

1. From the **Work Files** tab, click **Create New Workflow**.



2. A CREATE A NEW WORKFLOW dialog appears. Perform the following steps:
  - a. In the **Workflow Name** field, specify a name for the workflow. Names can contain letters, numbers, spaces, and any of the characters ( ) . - \_ . Workflows with other characters in the name cannot be created.
  - b. In the **Description** field, provide a brief description of the workflow. This field is optional.
  - c. From the **Compute Cluster System** dropdown list, select the compute cluster to use within the workflow.
  - d. From the **Data Storage System** dropdown list, select the data storage system to use within the workflow.

**CREATE A NEW WORKFLOW** [X]

Workflow Name \*  
New Workflow

Description  
This is a test workflow

**Workflow Datasources**

*i* A workflow acts on datasets that are located in your data sources. Please select the data sources needed for this workflow. At least one data source is required, and multiple sources can be added with the "Add A Data Source" action.

Compute Cluster System \*  
spark-standalone-few-large-executors

Data Storage System \*  
TDV

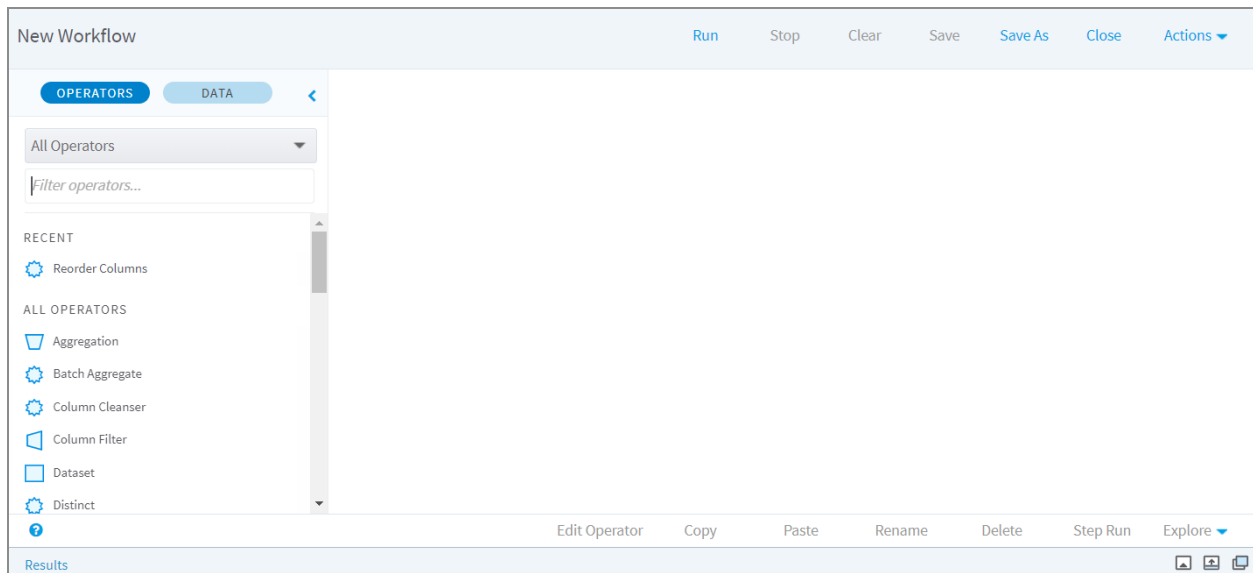
\* Required

Cancel Add Workflow

3. Click **Add Workflow**.

## Result

The workflow canvas is displayed, and you can edit it. The [Operator Explorer](#) and the [Data Explorer](#) are on the left side. Drag operators or data sources onto the canvas to begin building a workflow.



After the new workflow is saved, it is displayed in the [Work Files Tab](#) listing for the workspace.

## Importing TIBCO Data Virtualization Data into a Workflow

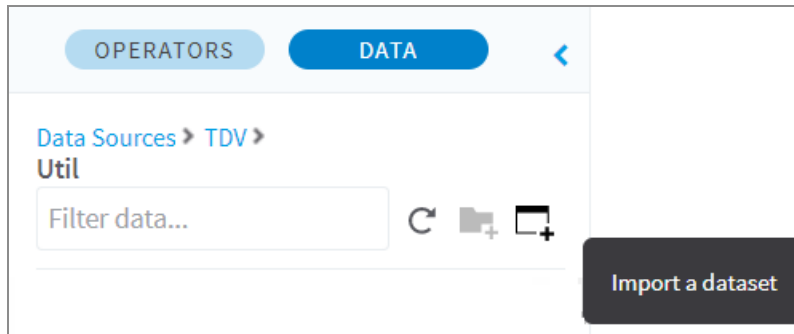
You can import data from a TIBCO® DV directly into a workflow.

### Before you begin

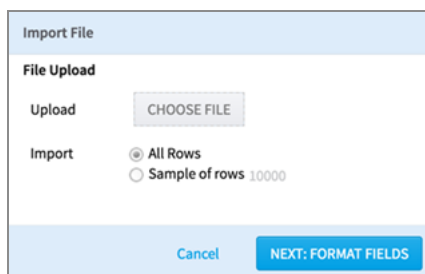
- You must have a workspace that you can write to.
- You must be connected to a TIBCO Data Virtualization data source.

### Procedure

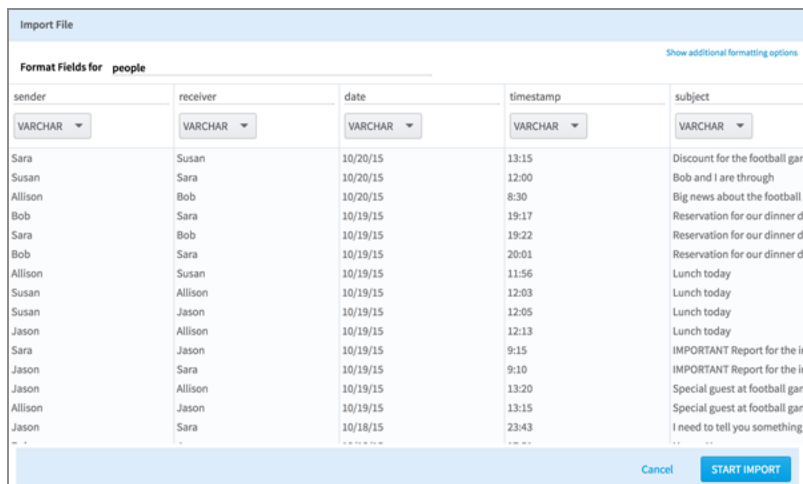
1. From the Data Explorer, navigate to a schema within a TIBCO Data Virtualization connection in the [Data Explorer](#) and select **Import a dataset**.



2. Select a file from the local disk.
3. When the file finishes uploading, select **Next: Format Fields**.



The following dialog displays a preview of the data set to import. You can specify the name of the table to create in the TIBCO Data Virtualization.



TIBCO Data Science - Team Studio determines the delimiter used in the file and the data type of each column. You can override the recommendation if necessary.

Use **Show additional formatting options** to choose the delimiter, escape character, and quote character. Additionally, you can specify whether the uploaded file contains

a header row and filter columns/empty values.

For each column, you can specify the name and data type, as well as options in the additional formatting options.

**Note:** To be imported as date, columns must be of the format "yyyy-mm-dd." To be imported as datetime, columns must be of the format "yyyy-[m]m-[d]d hh:mm:ss[.f...]". (Values in brackets are optional.)

4. To begin the import, click **Start Import**.

## Result

When the import is completed, you are notified with a confirmation message in the application.

# Creating a New Legacy Workflow

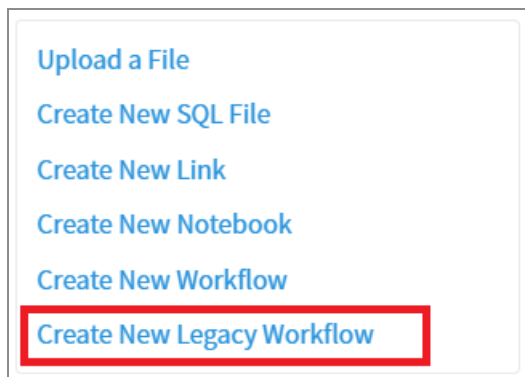
From the Work Files tab, users can create a new legacy workflow within their workspace.

## Before you begin

You must have a workspace that you can write to.

## Procedure

1. From the **Work Files** tab, click **Create New Legacy Workflow**.



2. A CREATE A NEW LEGACY WORKFLOW dialog appears. Perform the following steps:
  - a. In the **Workflow Name** field, specify a name for the workflow. Names can

contain letters, numbers, spaces, and any of the characters ( ) . - \_ . Workflows with other characters in the name cannot be created.

- b. In the **Description** field, provide a brief description of the workflow. This field is optional.
- c. From the **Data Source** dropdown list, select the data source to use within the workflow. If you are choosing a database data source, also select the database to use. Additional data sources can be added to the workflow later if necessary.



**Note:** You can add more than one data source to a workflow by using the **Add a Data Source** options.

CREATE A NEW LEGACY WORKFLOW

Workflow Name \*

Start-workflow

Description

This is a test workflow.

Workflow Datasources

A workflow acts on datasets that are located in your data sources. Please select the data sources needed for this workflow. At least one data source is required, and multiple sources can be added with the "Add A Data Source" action.

Data Source- \*

CDH6

Add A Data Source

\* Required

Cancel

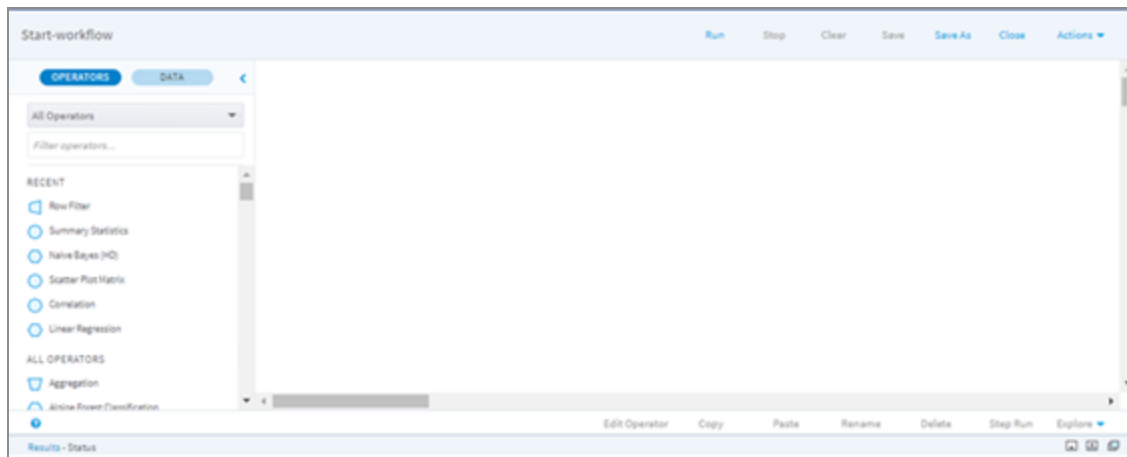
Add Workflow

### 3. Click **Add Workflow**.

## Result

The workflow canvas is displayed, and you can edit it. The [Operator Explorer](#) and the [Data Explorer](#) are on the left side. Drag operators or data sources onto the canvas to begin building a workflow.





After the new workflow is saved, it is displayed in the [Work Files Tab](#) listing for the workspace.

## Importing Database Data into a Workflow

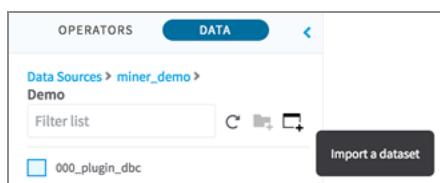
You can import data from a database directly into a workflow.

### Before you begin

You must have a workspace that you can write to.

### Procedure

1. Navigate to a schema within a database connection in the **Data Explorer** and select **Import**.



2. Choose a file from the local disk.
3. When the file finishes uploading, select **Next: Format Fields**.

**Import File**

**File Upload**

Upload

Import ☒ All Rows ☐ Sample of rows 10000

The following dialog displays a preview of the dataset to import. You can specify the name of the table to create in the database.

**Import File**

[Show additional formatting options](#)

**Format Fields for people**

sender	receiver	date	timestamp	subject
Sara	Susan	10/20/15	13:15	Discount for the football gam
Susan	Sara	10/20/15	12:00	Bob and I are through
Allison	Bob	10/20/15	8:30	Big news about the football g
Bob	Sara	10/19/15	19:17	Reservation for our dinner d
Sara	Bob	10/19/15	19:22	Reservation for our dinner d
Bob	Sara	10/19/15	20:01	Reservation for our dinner d
Allison	Susan	10/19/15	11:56	Lunch today
Susan	Allison	10/19/15	12:03	Lunch today
Susan	Jason	10/19/15	12:05	Lunch today
Jason	Allison	10/19/15	12:13	Lunch today
Sara	Jason	10/19/15	9:15	IMPORTANT Report for the in
Jason	Sara	10/19/15	9:10	IMPORTANT Report for the in
Jason	Allison	10/19/15	13:20	Special guest at football gam
Allison	Jason	10/19/15	13:15	Special guest at football gam
Jason	Sara	10/18/15	23:43	I need to tell you something

TIBCO Data Science - Team Studio determines the delimiter used in the file and the data type of each column. You can override the recommendation if necessary.

Use **Selecting Show additional formatting options** to choose the delimiter, escape character, and quote character. Additionally, you can specify whether the uploaded file contains a header row and filter columns/empty values.

For each column, you can specify the name and data type, as well as options in the additional formatting options.

**Note:** To be imported as date, columns must be of the format "yyyy-mm-dd." To be imported as datetime, columns must be of the format "yyyy-[m]m-[d]d hh:mm:ss[.f...]". (Values in brackets are optional.)

- To begin the import, click **Start Import**.

## Result

When the import is completed, you are notified with a confirmation message in the application.

# Importing Hadoop Data into a Workflow

You can import data from HDFS directly into a TIBCO Data Science - Team Studio workflow.

## Before you begin

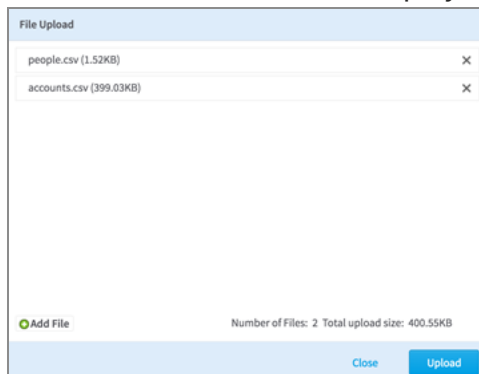
- You must have a workspace that you can write to.
- You must be connected to a Hadoop data source.

## Procedure

1. From the [Data Explorer](#), navigate to a folder within a Hadoop connection and select **Import a dataset**.



2. In the resulting dialog, click **Add File**, and then choose a file from your local disk. The selected file is displayed in the File Upload dialog. You can continue to add or remove files. The files are displayed in the dialog.



3. Click **Upload**

## Result

The selected files are uploaded. When the import is completed, you are notified with a confirmation message in the application

## Explore Visual Results

TIBCO Data Science - Team Studio uses Plotly charts and graphs to enhance how you interact with your data. Plotly is available to the Explore operators, to many Model operators, and for all associated results shown in Touchpoints and HTML reports. The topics in this section discuss tips and tricks to use these visualizations to the fullest extent. Note the location of each of these features in the diagrams.

## Navigating the Results Panel

You can see the results of running a workflow in the Results panel.

Perform these tasks from a workflow that you have open for editing.



### Before you begin

You must have read and write access to the workflow.

### Procedure

1. In the workflow, select an operator.
2. In the lower left corner, click **Results** (indicated in the image by 1).

Optionally, in the lower right corner, you can click the left-most icon displaying the small up arrow (also labeled 1 in the image). When the Results panel is open, this icon is displayed as a small down arrow, which you can use to close the Results panel.

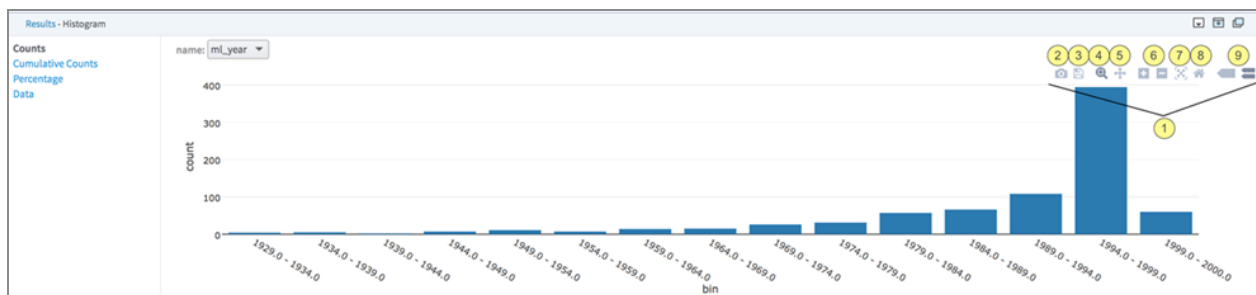
3. To increase the size of the Results panel, click **Expand**.  
(Click again to return to the original panel size.)

Alternatively, you can open your visual results in a new window.





## Plotly Charts and Graphs

Plotly visualizations are available for Exploration operators and several Model operators.

After you access the Results panel, you can use the additional features provided with Plotly charts and graphs to help you explore your data. These additional features, shown in the following screen capture, are highlighted by numbers that correspond to the descriptions provided.



Reference number	Description
1	To find the available actions, hover in the top right corner of your visualization.
2	To download the visualization to your computer as a PNG file, use the Camera icon.
3	To export your visualization to the Plotly online chart editor, use the <b>Save</b> icon. From the Plotly online chart editor, you can modify colors, data sort orders, and other characteristics of your visualization.
4	To zoom for more detail of a visualization area, click and drag that area. This is the default action when you open the Plotly chart. If you change the default action, you can return to this option by clicking the associated icon, labeled in the image
5	To switch from zoom to pan, use the <b>Pan</b> icon.

Reference number	Description
	To zoom manually, use [+] and [-].
	To autoscale your visualization, use the crossed arrows icon.
	To return to the original view, click the <b>Home</b> icon.
	To display hover information, choose one of the two options: One shows values on both axes, and the other shows only the primary axis value.

## Spark Optimization for Data Scientists

Many operators in TIBCO Data Science - Team Studio run using custom Spark algorithms. While many Spark implementations come with default configuration settings, these might not be optimal for every use case.

You can edit your Spark configuration from TIBCO Data Science - Team Studio in three ways:

- Operator settings
- Alpine.conf (overridable at the workflow level with workflow variables)
- Data source configuration

You can edit the settings, such as memory and executors, for each of these options. Each takes effect on a different scope of the application.

To learn more about Spark settings, see the [official documentation](#).

## Spark Autotuning

Tuning your Spark parameters can get confusing. TIBCO Data Science - Team Studio includes automatic optimization for increased performance on Spark jobs.

Based on the size of your cluster, the available resources in your queue, the size of the input data, and what is known about the operator, TIBCO Data Science - Team Studio can dynamically assign Spark parameters at runtime. Spark autotuning is currently available on the following operators.

- Aggregation
- Alpine Forest Classification
- Alpine Forest Regression
- ARIMA Time Series
- Association Rules
- Batch Aggregation
- Classification Threshold Metrics
- Collapse
- Column Filter
- Correlation
- Correlation Filter
- Distinct
- Fuzzy Join
- Gradient Boosting Classification
- Gradient Boosting Regression
- Join
- K-Means
- LDA Trainer
- LDA Predictor
- N-Gram Dictionary Builder
- Naive Bayes
- Neural Network
- Normalization
- Null Value Replacement

- Numeric to Text
- Pivot
- Replace Outliers
- Sort by Multiple Columns
- Linear Regression
- Logistic Regression
- Resampling
- Row Filter
- Set Operations
- Stability Selection
- Summary Statistics
- Text Extractor
- Text Featurizer
- Transpose
- Unpivot
- Variable
- Window Functions - Aggregate
- Window Functions - Lag/Lead
- Window Functions - Rank

To enable Spark autotuning, no action is required. These operators default to **Automatic Optimization** being applied. You can apply a greater degree of control by editing the advanced configuration for each of the Spark settings.

TIBCO Data Science - Team Studio sets the following Spark parameters.

- `spark.executor.memory`
- `spark.driver.memory`
- `spark.executor.cores`
- `spark.default.parallelism` and `spark.sql.shuffle.partitions`



Additionally, TIBCO Data Science - Team Studio can determine if dynamic allocation is enabled on the cluster and, if so, to use that to choose the maximum number of executors (`spark.dynamic.allocation.max.executors` and `spark.dynamic.allocation.enabled`). If dynamic allocation is not enabled on the cluster, TIBCO Data Science - Team Studio sets a value for `spark.executor.instances` based on your cluster size, input data, and current operator.

To override the settings, set **Automatic Optimization** to **no**, and then edit the settings provided in the [Advanced Settings dialog](#) or add your own key/value pairs. TIBCO Data Science - Team Studio always uses a setting provided by the user before attempting to specify one.

## Settings for Spark-Enabled Operators

The easiest and quickest way to change Spark settings is from the operator itself.

- You can enable Automatic Configuration for the Spark parameters. TIBCO Data Science - Team Studio selects default values to run the operator.
- You can edit these parameters in the operator parameter dialog directly by setting **Advanced Settings Automatic Optimization** to **No**, and then clicking **Edit Settings** to display and edit the settings in the [Advanced Settings dialog](#).

Additional parameters can be available, depending on the operator. Additionally, you can add parameters, using any of those mentioned in the [official Spark documentation](#).

As a use case example, imagine that you are parsing a lot of files using the Text Extractor. The Spark job keeps failing or going very slow. Depending on your input data, you can take one of the following actions to correct these problems.

- If you have lots of medium or small sized file (hundreds of thousands of files < 40MB) to parse and the job is failing, you should try to increase the driver memory and the number of executors.
- If you have bigger files to parse ( > 90MB) and the Spark job is failing, increase the executor memory so that the bigger files are parsed by a single executor. You should also increase the driver memory.

## Data Source configuration

Spark settings can be changed on the data source itself. To do this, you must have access to the Hadoop cluster with Spark installed.

## Tips and tricks

For more information about Spark optimization, see the following resources.

- [Performance Tuning for SparkSQL](#)
- [Spark on YARN Parameters](#)
- [Spark Tuning Cheat-Sheet](#) (mentioned in the video lecture)
- [Top 5 Mistakes When Writing Spark Applications](#)

## Advanced Settings dialog

When Spark is enabled for an operator, you can apply the Automatic configuration for the Spark parameters, setting the default values to run the operator. However, you can edit these parameters directly.

To edit these parameters directly in the operator parameter dialog, select **No** for **Advanced Settings Automatic Optimization**, and then click **Edit Settings**. Set your desired configuration in the resulting Advanced Settings dialog.

Advanced Settings

[Spark Documentation](#)
[Add Parameter](#)

Override?	Name	Value
<input type="checkbox"/>	Disable Dynamic Allocation	<input type="checkbox"/>
<input type="checkbox"/>	Number of Spark Executors	<input type="text"/>
<input type="checkbox"/>	Spark Executor Memory (MB)	<input type="text"/>
<input type="checkbox"/>	Spark Driver Memory (MB)	<input type="text"/>

**i Note:** Available options are determined by the type of operator. The following table shows the settings that apply for all operators for which you can enable Spark. For information about additional settings, see the specific operator help.

- If you check a checkbox from the **Override?** column, you can specify a value for the corresponding setting, which supersedes any default value set by your cluster or workflow variables. If you provide no alternative value, the default value is used.
- If you click **Add Parameter**, you can provide custom Spark parameters. This option provides more control and tuning on your Spark jobs. See [Spark Autotuning](#) for more information.

Setting	Description
<b>Disable Dynamic Allocation</b>	<p>Select both checkboxes to indicate that idle CPU cores or execution memory should not be released for other applications.</p> <p>Dynamic allocation allows Spark to increase and decrease the number of executors as it needs them over the course of an application. If you can configure dynamic allocation on your cluster, it is probably most performant to do so.</p> <p>By default, dynamic allocation is disabled. TIBCO Data Science - Team Studio can use dynamic allocation only if the following conditions are true.</p> <ul style="list-style-type: none"> <li>• It is enabled in <code>alpine.conf</code>.</li> <li>• You have not set the number of executors.</li> <li>• Your cluster is correctly configured for dynamic allocation.</li> </ul>
<b>Number of Executors</b>	Specify the number of Spark executors to run this job ( <code>spark.executor.instances</code> ).
<b>Executor Memory in MB</b>	<p>Specify the Spark executor memory in megabytes.</p> <p>This value depends on the size of the data, the resources on the cluster, and the YARN container. TIBCO Data Science - Team Studio prevents you from setting this value higher than the size of the YARN container. Override this behavior by setting the <code>limit.spark.executor.memory</code> value in <code>alpine.conf</code> to <code>false</code>.</p>
<b>Driver Memory in MB</b>	<p>Specify the Spark driver memory, in megabytes.</p> <p>Some operators, such as Alpine Forest and Summary Statistics, pull a lot of information back to the driver, so these operators assign more driver memory.</p>

Setting	Description
	This value depends on the size of the data, the resources on the cluster and the YARN container, and the algorithm. TIBCO Data Science - Team Studio prevents you from setting this value higher than the size of the YARN container, even if this is set by the user. Override this behavior by setting the <code>limit.spark.executor.memory</code> value in <code>alpine.conf</code> to <code>false</code> .
<b>Number of Executor Cores</b>	<p>Specify the number of executor cores to use on each executor for the Spark job (<code>spark.executor.cores</code>).</p> <p>When this value is explicitly set, multiple executors from the same application can be launched on the same worker if the worker has enough cores and memory. Otherwise, each executor grabs all the cores available on the worker by default, in which case only one executor per application can be launched on each worker during one single schedule iteration. See the Spark documentation for more information.</p>

## Additional Information

See the [Spark Documentation](#) for more details on the available properties.

## alpine.conf Spark Settings

The following settings are added to the Spark submission. You can edit Spark settings, TIBCO Data Science - Team Studio-specific Spark settings, and YARN settings. All of these values can be found and edited in the file `alpine.conf`. Any Spark tuning you define at the operator level takes precedence.

### Spark parameters at the datasource level

As of version 6.2, you can specify any Spark parameter at the datasource configuration level in TIBCO Data Science - Team Studio. However, any Spark parameter you define in `alpine.conf` or at the operator level takes precedence.

**CONFIGURE CONNECTION PARAMETERS**

Edit Connection Parameters Individually

Parameter	Value	Action
hadoop.security.authentication	kerberos	✖
yarn.app.mapreduce.am.staging-dir	/user	✖
yarn.resourcemanager.admin.address	cdh5adkerberosa.alpinenow.local:8033	✖
yarn.resourcemanager.resource-tracker.address	cdh5adkerberosa.alpinenow.local:8031	✖
dfs.datanode.kerberos.principal	hdfs/_HOST@ALPINENOW.LOCAL	✖
dfs.namenode.kerberos.principal	hdfs/_HOST@ALPINENOW.LOCAL	✖
mapreduce.jobhistory.principal	mapred/_HOST@ALPINENOW.LOCAL	✖
yarn.resourcemanager.principal	yarn/_HOST@ALPINENOW.LOCAL	✖
yarn.resourcemanager.scheduler.address	cdh5adkerberosa.alpinenow.local:8030	✖
alpine.principal	alpine/alpineqa4.alpinenow.local@ALPINENOW.LOCAL	✖
alpine.keytab	/home/chorus/keytab/alpine.keytab	✖
<b>spark.yarn.queue</b>	<b>root.users.engineering</b>	✖

Load Configuration from Resource Manager

Cancel Save

## Spark Values

The following settings are added to the Spark submission in alpine.conf, where you can edit them if you choose. While these can be used to define some overall settings, any Spark tuning you define at the operator level takes precedence.

Name	Default value	Notes
spark.yarn.api.timeout	5000	
spark.yarn.report.interval	2000	
spark.executor.extraJavaOpts		
spark.rdd.compress	true	
spark.io.compression.codec	"org.apache.spark.io.Snappy	

Name	Default value	Notes
	CompressionCodec"	
spark.eventLog.enabled	false	
spark.dynamicAllocation.enabled	true	Dynamic allocation can be disabled at the operator level. Regardless of this value, we use only dynamic allocation if your cluster is correctly configured to enable dynamic allocation.
spark.driver.maxResultSize	2g	A Spark value that sets the maximum size of data that can be brought back to the driver.
spark.driver.extraJavaOptions	-XX:MaxPermSize=256m	

## Team Studio-Specific Spark Values

TIBCO Data Science - Team Studio uses the following settings to determine the bulk of the Spark settings (if they are not set manually at the operator level).

Name	Default Value	Notes
spark.driver.memory	1024	Minimum driver memory. For large data sets and large clusters, we increase this value.
spark.show.conf	true	
min.spark.executor.memory	1024	

Name	Default Value	Notes
<code>percent.resources.available.to.alpine.job</code>	1.0 = 100%	Percent of available resources that we allocate to a given job. Consider reducing this value if there are many TIBCO Data Science - Team Studio users or if you are worried about them launching very large jobs.
<code>limit.spark.driver.memory.based.on.capacity</code>	true	<p>Limits the Spark driver memory based on the memory capacity of the YARN container. If the memory setting is too high, we use the largest possible driver memory that still fits in the YARN container.</p> <p>The YARN container must be large enough to accommodate the memory set by the Spark driver memory parameter and the overhead.</p>
<code>limit.spark.executor.memory.based.on.capacity</code>	true	<p>Limits the Spark executor memory based on memory capacity of the YARN container. If <code>spark.executor.memory</code> requested is too large, we set <code>spark.executor.memory</code></p>

Name	Default Value	Notes
		to the capacity. This setting also ensures that the total executor memory requested + driver memory is not too large for the total available memory on the cluster.
<code>spark.max.executors.per.machine</code>	5	
<code>alpine.small.cluster.threshold.g</code>	6	We assume that if the total resources on the cluster are less than 6 GB, minimum memory settings are used.

## YARN Configuration Values

To read information about your cluster, we access the YARN API connected to it. If we cannot contact the YARN API (usually due to security or ports being blocked), we use the following default values for your cluster size.

Name	Notes
<code>assumed.yarn.number.nodes = "4"</code>	Number of nodes (and thus how many containers) the cluster has available.
<code>assumed.yarn.scheduler.maximum.allocation.vcores = "15"</code>	Assumed number of virtual cores in the YARN configuration file. If the YARN can be reached, we use <code>yarn.scheduler.maximum.allocation.vcores</code> . Otherwise, we use the "assumed" value.  If we cannot get the total available virtual



Name	Notes
	cores for the cluster, we assume that it is the number of nodes multiplied by the cores per container, and that the container is the size of a node.
<code>assumed.yarn.scheduler.maximum.allocation.mb= "8000"</code>	<p>Assumed memory in megabytes available to each YARN container. We use the <code>yarn.scheduler.maximum.allocation.mb</code> value from the YARN configuration file if we can get it; otherwise, we use the "assumed" value.</p> <p>This is the value used to cap the driver and executor memory. If we cannot determine the resources available on the cluster, we assume that the memory available on the cluster is the number of nodes multiplied by this value.</p>

## Deploying Models and Workflows

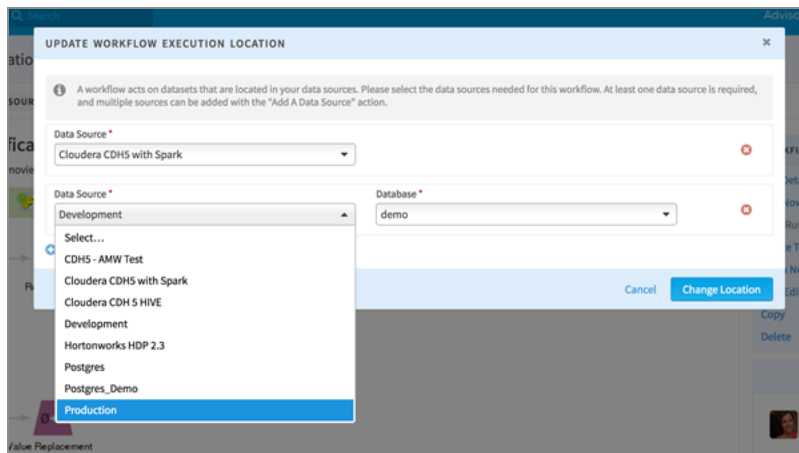
Once a workflow is developed, there are several ways that it can be deployed or operationalized.

Production, deployment, and optimization are explored in the following use cases.

## Moving a Workflow from Development to Production

Typically, a workflow is developed within one environment-perhaps a private folder on Hadoop, or a development schema-and then deployed within another. You can easily switch which data sources a workflow executes in, and then to transition completed flows from development to production.

If the database and table structure (or HDFS file structure) match between the development and production environments, the workflow runs in the new environment with no modifications. If the structure is different, the workflow might require some initial mapping.



Moving a workflow from a development data source to production is as easy as updating its execution location.

If TIBCO Data Science - Team Studio is not connected to the production environment, you have several options for exporting your workflows. For ETL workflows on database, you can download a SQL log that can run directly against another database. For modeling workflows, use the export operator to create PMML, PFA, or TIBCO Data Science - Team Studio Model Format documents. PMML and PFA documents can be embedded in scoring engines for real-time single sample scoring (see below), and TIBCO Data Science - Team Studio Model Format documents can be executed by Java objects.

```
[16:54:59] Logistic Regression started running .....
[16:55:00] Logistic Regression finished
[16:55:00] Export started running .....
[16:55:00] Exported the model file golf.pfa to the local Chorus Workspace.
[16:55:00] Export finished
[16:55:00] Analytic Flow finished
[16:55:00] Click on individual operators to see their output.
[16:55:00] Click here to download SQL log.
```

PFA export and SQL download capabilities in the activity log.

## Preparing Data and Deploying Models

In many cases, data science teams prepare data from a variety of different sources before modeling.

This data might originate from relational databases, flat files, or structured and unstructured data in Hadoop. A single workflow can connect to all of these sources for aggregation and cleansing into a final consolidated representation, which can then be moved using [Data Operators](#) to the desired analytics sandbox, such as a folder in HDFS.

Teams typically operationalize these flows using the job scheduler, periodically updating the analytics sandbox with a cleansed and aggregated version of the latest live data. The

same jobs can contain subsequent modeling flows that update trained models as soon as the new data are available.

## Optimizing Models

Operations teams should simplify and optimize the workflows that are created in the development environment by the analytics team. Over time, the operations team can use the collaborative environment to encourage and enforce best practices, by creating template workflows and custom operators that encapsulate standards and optimizations.

Here are some best practices when optimizing your workflow for production:

- Remove non-essential operators such as charts, summary statistics, accuracy tests, and unapproved models.
- Consolidate multiple operators into one (for example, successive variable operators).
- Perform pre-aggregations and filters as early as possible.
- Perform pre-aggregations and filters at the source of the data.
- Use [Sub-Flow](#) to standardize common transformations.
- Use custom operators to streamline complex operations into a single step, and to standardize common operations. (See *TIBCO® Data Science - Team Studio Development Kit* for more information.)

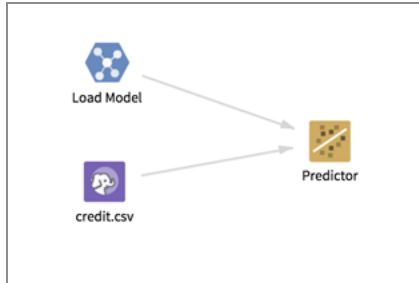
You can score, deploy, and manage your models in one place. You can also automate job runs and integrate them into your development process from the command line using our APIs.

## Batch Model Scoring

To batch score new data in a workflow, first export the desired model in TIBCO Data Science - Team Studio Model Format using the Export operator.

**i Note:** It's possible to score directly off of a modeling operator, but in that case you'll also retrain the model every time you run the workflow. The [Export](#) lets you fix the version of the model used for scoring. Your model can now be loaded into any other workflow in the workspace using the [Load Model](#) operator. Connect the load model operator and the data set you'd like to score to a [Prediction Operators](#), and the entire data set is scored in batch mode.

The Load Model operator works both in database and in Hadoop, so you can train a model on Hadoop, export it, then load it up in, for example, Teradata for scoring. As you accrue more data, you can [Testing Models for Performance Decay](#) for performance regressions, and automatically publish new models when performance improves.



## Real-Time Model Scoring

The Export operator can also export in PMML and PFA formats.

Projects like [jPMML](#) provide scoring engines for PMML, and TIBCO Data Science - Team Studio provides a RESTful PFA scoring engine management service, ScORE, based on the [Hadrian engine](#) for PFA. These formats are appropriate for real-time, single sample, or small-batch scoring.

## Code Generation

ETL workflows on databases generate SQL logs that can be ported to other database environments.

In addition to the [Load Model](#) operator, we supply a Java object that can hydrate TIBCO Data Science - Team Studio Model Format for scoring in custom Java or Scala applications, or from the command line.

SQL generated from an ETL workflow:

```

-----
-- SQL Log for workflow: Credit ETL (version: 4);
-- Run by user: mthyen;
-- Date: 06-09-2017 13:50:05;
-----
-- Operator: Table Join;
drop table "demo"."djc";
CREATE TABLE "demo"."djc" AS (SELECT "credit_a"."id" as "id","credit_a"."times90dayslate" as "times90dayslate","credit_a"."revolving_util" as "revolving_util","credit_a"."debt_ratio" as "
-----
-- Operator: Replace nulls with medians;
drop table "demo"."alp1399_1573_repl_0";
CREATE TABLE "demo"."alp1399_1573_repl_0" AS (SELECT "id","times90dayslate","revolving_util","debt_ratio","credit_lines","monthly_income","times30dayslate_2years","srslqncy", (case when
-----
-- Operator: Normalization;
select avg(CAST ("credit_lines" AS DOUBLE PRECISION)),var_pop(CAST ("credit_lines" AS DOUBLE PRECISION)) from "demo"."alp1399_1573_repl_0";
drop table "demo"."djc_norm";
CREATE TABLE "demo"."djc_norm" AS (SELECT "id","times90dayslate","revolving_util","debt_ratio",["credit_lines"-(CAST (4.95064 AS DOUBLE PRECISION))]/sqrt(CAST (1.7264435904 AS DOUBLE PREC

```

## Model Management

Starting in version 6.3, TIBCO Data Science - Team Studio provides advanced features for model governance and deployment to real-time scoring services. In the **Administration** section, you can modify the deployment targets to which TIBCO Data Science - Team Studio has access.

The following image shows the **Deployment Targets** page in the **Administration** section.

Administration

Administration > Deployment Targets

Deploy target created

### Deployment Targets

URL	Name	Environment	Server Type
http://test2.com	test2	Development	ScORE
http://test.com	test	Production	ScORE

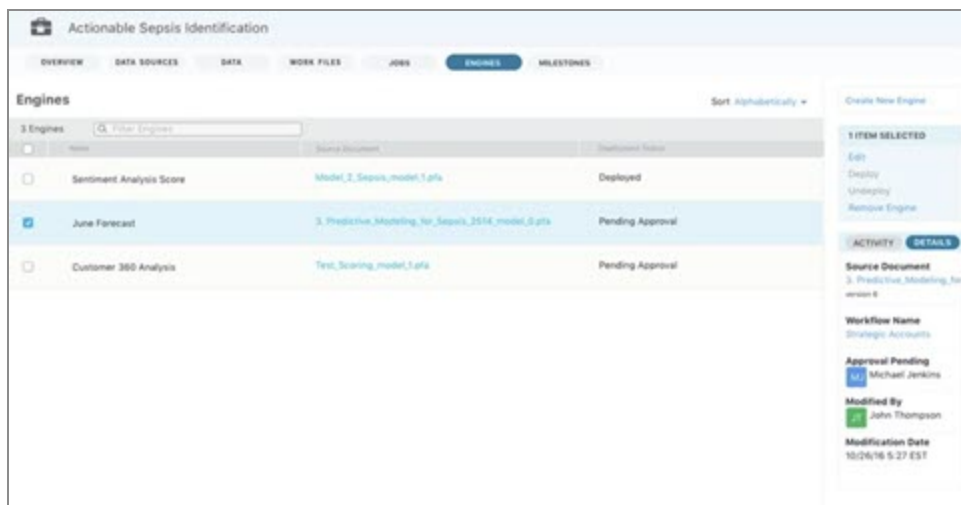
Create New Target

1 ITEM SELECTED

Delete

Development
ScORE
Cancel
Save

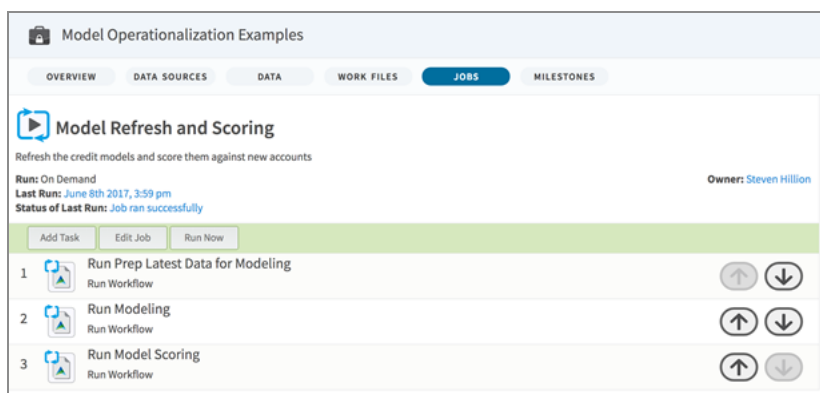
With deployment targets defined, you can create engines with PFA documents in your workspace. **Engines** is a new tab in the workspace navigation for managing deployed real-time models, including an explicit approval step before they can be deployed to production scoring environments.



## Workflow Scheduling

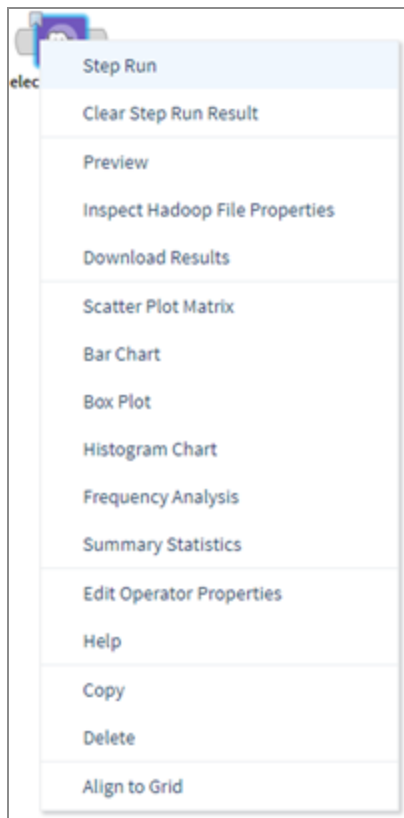
Your workflows and models can also be scheduled in a job. Jobs can run a variety of tasks, ranging from simply executing a workflow to refreshing, scoring, and deploying models.

For more information, see [Jobs Tab](#).



## Preview and Visualize Data

We provide several data visualization and exploration options in the contextual menu of any data source. Right-click the data operator to see the options.



In addition to on-the-fly data visualization, TIBCO Data Science - Team Studio provides additional data exploration and visualization functionality through the [Exploration Operators](#) operators.

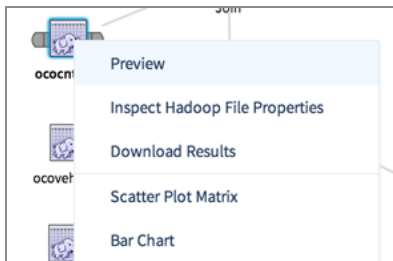
## Preview

**Preview** provides a way for you to quickly see a subset of the data contained in a dataset.

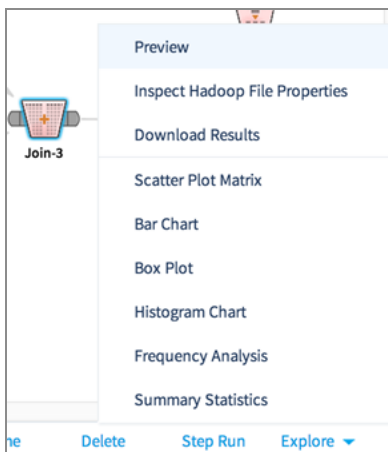
/emilie/ococnt.csv				
idsousc	anccot	dtdech	cpdaaucn	dtderent
446,433	644.05	10/22/2013	0	2/26/2012
5.26e+6	217.94	6/2/2014	1	12/15/2013
7.52e+6	387.06	3/8/2013	1	6/29/2014
2.02e+6	483.95	9/1/2012	0	5/4/2013
2.73e+6	912.99	3/20/2014	1	1/6/2014
9.23e+6	132.71	11/24/2013	0	5/4/2014
2.27e+6	204.65	1/6/2013	0	5/5/2014
8.74e+6	699.22	3/12/2013	0	5/6/2014
9.48e+6	893.06	9/19/2012	0	5/7/2014
166,136	39.04	2/23/2012	1	5/8/2014

Use one of the following two methods to access **Preview**.

- Right-click on the data operator.



- At the bottom of the workflow canvas, from the **Operator** toolbar, click **Explore > Preview**.



## Notes

- The default number of rows displayed in **Preview** are configurable. See your administrator for more information.
- For **Preview** to be enabled, your system must be configured for the following.
  - Database operators configured to output a table or a view.
  - Hadoop operators configured to store results.

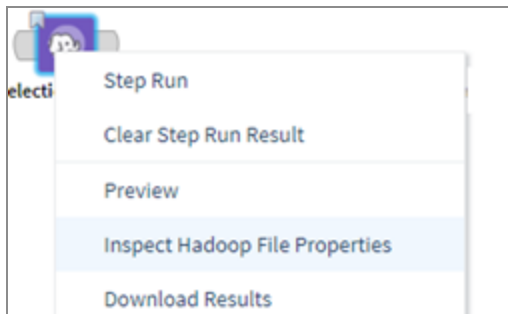
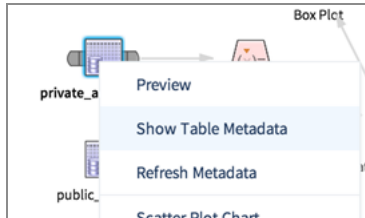
- **Note:** Disable any pop-up blockers to view the preview window.



## Show Table Metadata or Inspect Hadoop File Properties

The **Show Table Metadata** and **Inspect Hadoop File Properties** commands provide information about a data source.

If you are reading from a database, click **Show Table Metadata**:



If you are reading from a Hadoop data source, click **Inspect Hadoop File Properties**:

Table Columns	
Column Name	Column Type
rank	DOUBLE PRECISION
name	TEXT
state	TEXT
admitrate	DOUBLE PRECISION
graduate_rate_4yr	DOUBLE PRECISION
total_cost_per_year	DOUBLE PRECISION
avg_needbased_aid	DOUBLE PRECISION
avg_nonneedbased_aid	TEXT
perc_of_nonneed_based_aid	TEXT
ave debt at eraduation	TEXT

[Done](#)

Hadoop File Properties	
Name	election92.csv
Owner	hdfs
Group	supergroup
Last modified time	Mar 10, 2018 5:00:09 PM
Access time	Aug 28, 2018 2:48:55 PM
File size	465.17KB
Block size	128.0MB
Permissions	rw-r--r--

## Scatter Plot Chart

Use Scatter Plot Chart to create a scatter plot of the corresponding dataset.

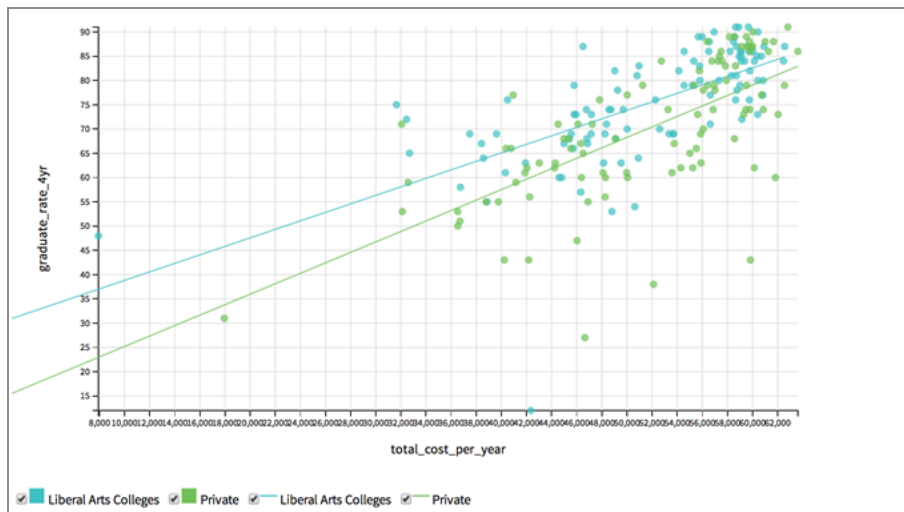
In order for Scatter Plot Chart to be enabled, database operators must output a table/view.

Scatter Plot Chart is accessible from the **Explore** menu of the operator toolbar or by right-clicking an appropriate operator.

### Description of parameters

- **Reference Column (Y-axis)** corresponds to the X-axis.
- **Analysis Column (X-axis)** corresponds to the Y-axis.
- **Category Column** is a grouping of points denoted by color.

Scatter Plot Parameters	
Analysis Column (Y-axis)	graduate_rate_4yr
Reference Column (X-axis)	total_cost_per_year
Category Column	type
<div> <div>Cancel</div> <div>OK</div> </div>	



## Bar Chart

Visualizes the attributes of a data set in bar chart format.



## Information at a Glance

Parameter	Description
Category	Explore
Data source type	DB, HD
Send output to other operators	No
Data processing tool	Pig

Use this operator to choose the following three attributes (columns) in the data table as the dimensions in the bar chart:

- The X dimension (**Category**) takes a category-type attribute to construct the X-axis.
- The Y dimension (**Value**) takes a numerical attribute to construct the Y-axis.
- The grouped dimension (**Series**) accepts a category-type attribute to group the bars in the chart.

## Input

A data set from the preceding operator.

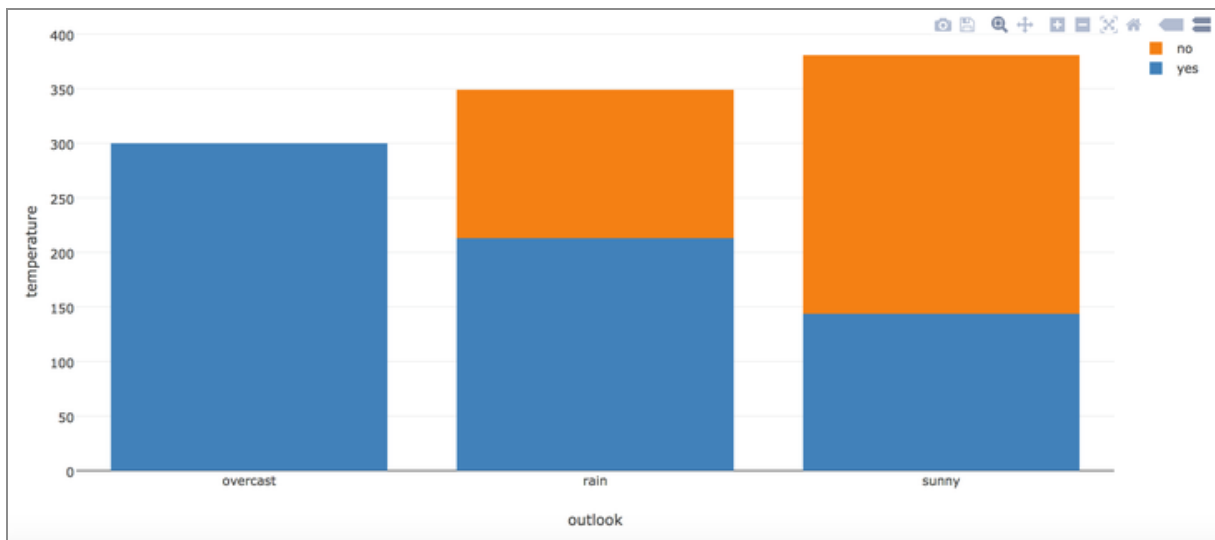
## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Category</b>	The categorical column for the X-axis.
<b>Series</b>	The categorical column for the groups.
<b>Value</b>	The numerical column for the Y-axis.

## Output

### Visual Output

A bar chart such as the following example.

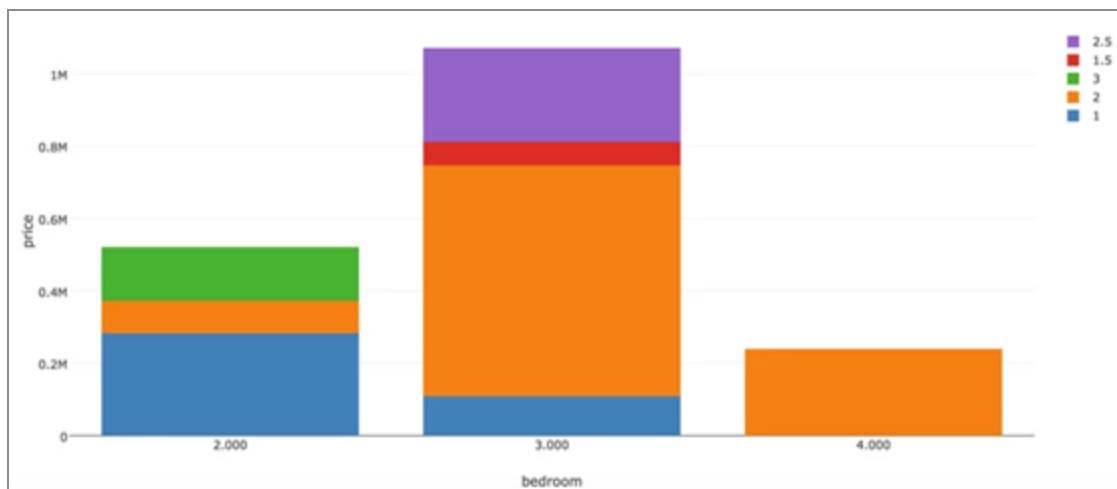


## Data Output

None. This is a terminal operator.

## Example

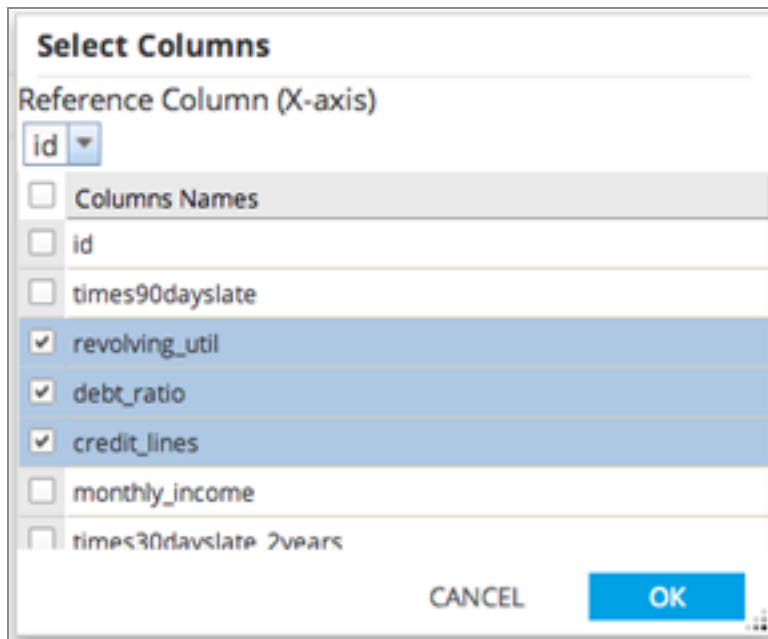
The following example shows housing sale prices (**Value**) by number of bedrooms (**Category**) for each number of baths (**Series**).



## Univariate Plot Chart

The Univariate Plot Chart allows users to choose a reference column (x-axis) and one or more Y columns from a dataset.

All of the columns must be of numerical type. A univariate chart is created for each Y column selected.

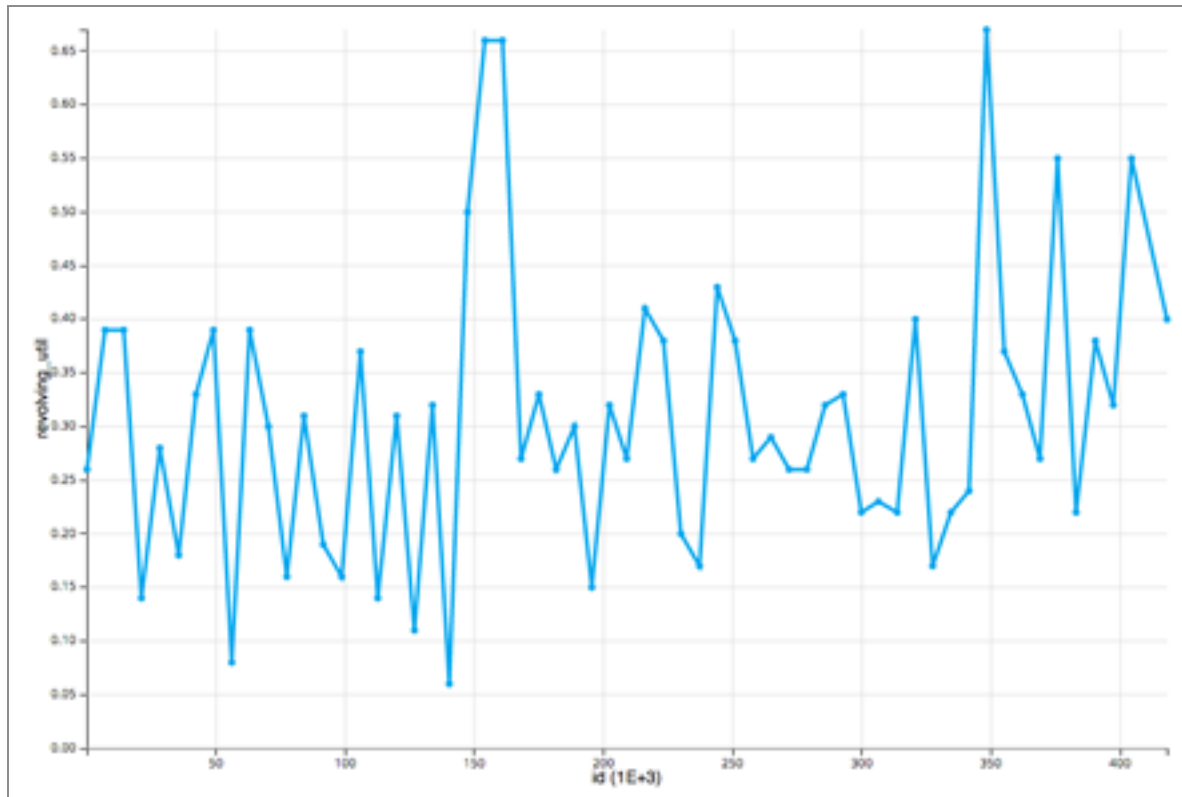


The image shows a 'Select Columns' dialog box. At the top, it says 'Select Columns'. Below that, there is a section titled 'Reference Column (X-axis)' with a dropdown menu showing 'id'. Below this, there is a list of columns with checkboxes: 'Columns Names' (unchecked), 'id' (unchecked), 'times90dayslate' (unchecked), 'revolving\_util' (checked), 'debt\_ratio' (checked), 'credit\_lines' (checked), 'monthly\_income' (unchecked), and 'times30dayslate 2years' (unchecked). The 'revolving\_util', 'debt\_ratio', and 'credit\_lines' rows are highlighted in blue. At the bottom right, there are 'CANCEL' and 'OK' buttons.

Reference Column (X-axis)
id

Columns
<input type="checkbox"/> Columns Names
<input type="checkbox"/> id
<input type="checkbox"/> times90dayslate
<input checked="" type="checkbox"/> revolving_util
<input checked="" type="checkbox"/> debt_ratio
<input checked="" type="checkbox"/> credit_lines
<input type="checkbox"/> monthly_income
<input type="checkbox"/> times30dayslate 2years

CANCEL OK



## Box and Whisker Chart

The Box-And-Whisker Chart allows users to choose an analysis value column, an analysis series column, and an analysis type column from a dataset.

The maximum value, minimum value, mean, median, Q1 & Q3 of the **Analysis Value** of each **Analysis Series** in each **Analysis Type** are plotted in Box-And-Whisker diagrams in the resulting graph.

- In order for Box-And-Whisker Chart to be enabled, database operators must output a table/view.

The Box-And-Whisker Chart is accessible from the **Explore** menu of the operator toolbar or by right-clicking an appropriate operator.

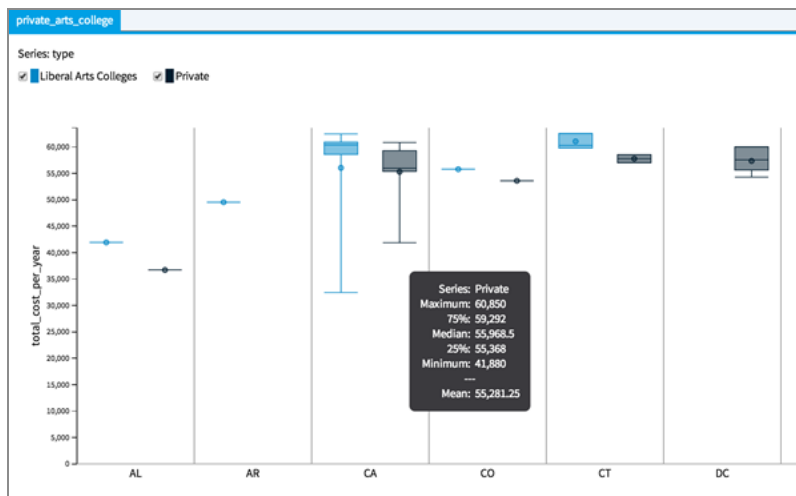
**Box Plot**

Analysis Value (Y-axis)

Analysis Series

Analysis Type (X-axis)

- **Analysis Value (Y-axis)** corresponds to the Y-axis.
- **Analysis Series** is grouping denoted by color.
- **Analysis Type (X-axis)** corresponds to the X-axis.



## Histogram

Analyzes the values of the selected fields of a data set, and generates a graphical representation of the frequency distribution of the numeric data.



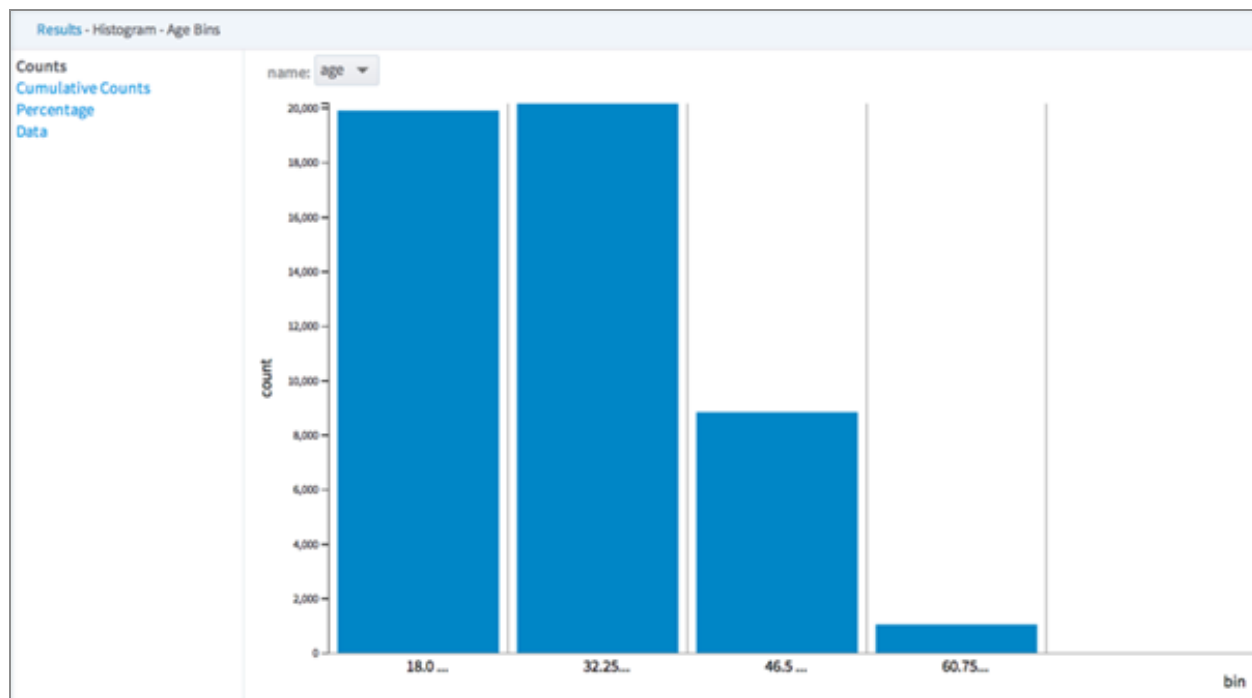


## Information at a Glance

Parameter	Description
Category	Explore
Data source type	DB, HD
Send output to other operators	No
Data processing tool	Pig

## Algorithm

Histogram analysis calculates data frequency for a specific column.



For each column specified, users input either the number of bins to generate or the width of the bins. A bin is an interval that is divided equally between minimum and maximum value or by the width.

For example, a specific column's minimum value is 0 and maximum value is 100. If the user specifies five bins, each bin covers 20 units. If 10 bins are specified, each bin covers 10 units.

Bounds of each bin are defined as (**Minimum**, **Maximum**].

**i Note:** When a user defines a minimum value, this value is included or displayed in the bin. The first bin begins at the lowest value above the defined minimum, and the last bin includes the defined maximum value.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Bins</b>	Select <b>Bin Configuration</b> to choose the available columns from the input data set for analysis.  See <a href="#">Bin Configuration dialog</a> .

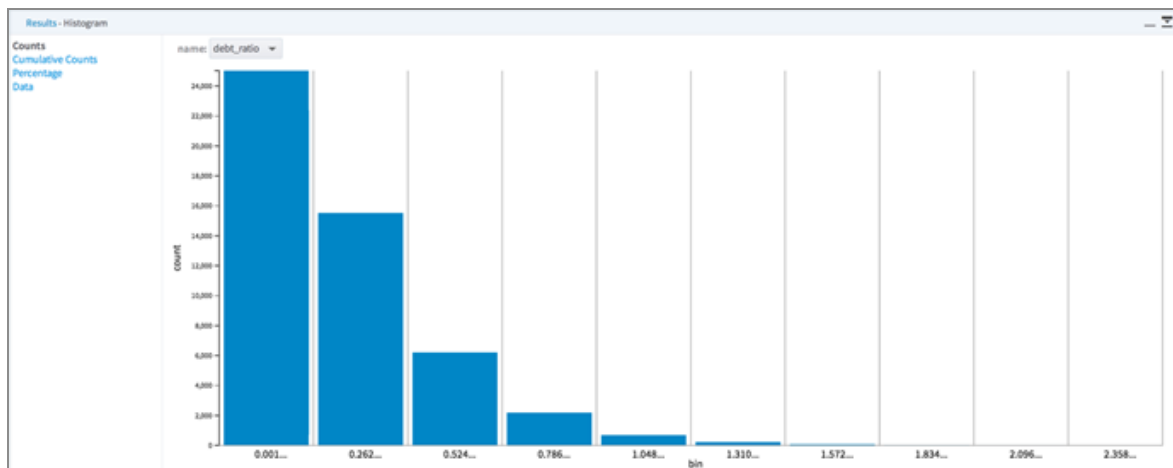
## Output

### Visual Output

Four sections are displayed: **Counts**, **Cumulative Counts**, **Percentage**, and **Data**.

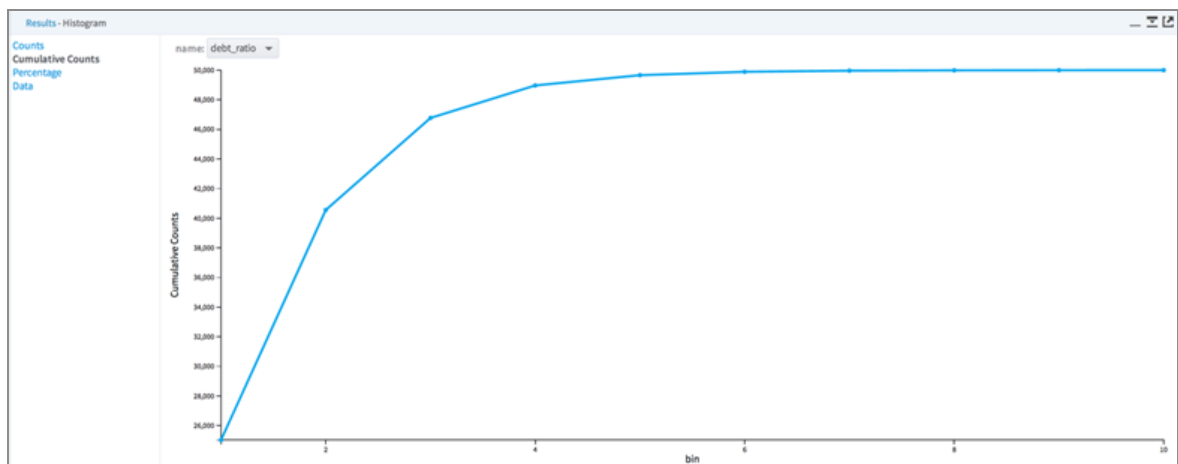
#### Counts

Displays the histogram for one column at a time according to the defined groups (bins). Users can select a column from the **name** dropdown list.



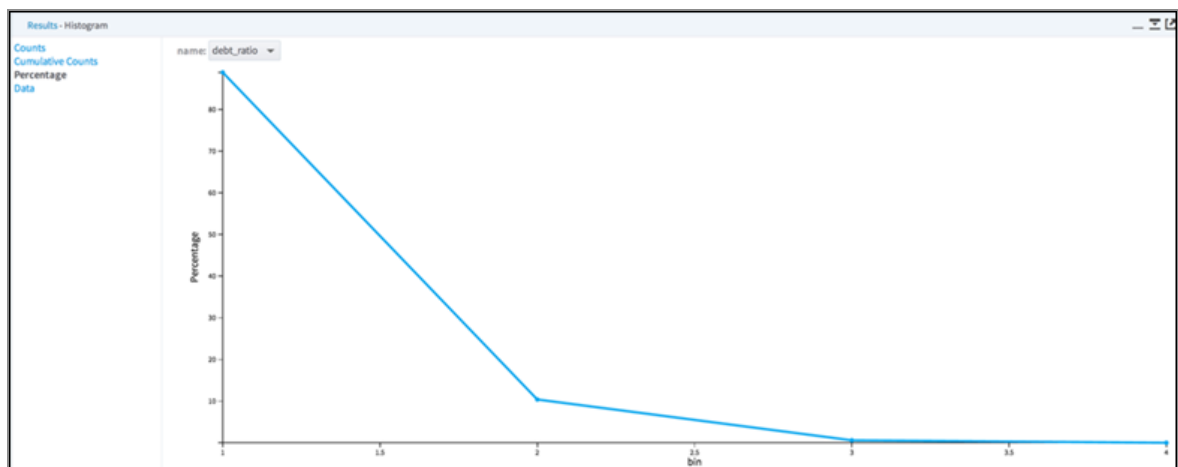
## Cumulative Counts

Displays a graph of the number of rows included with each additional bin.



## Percentage

Displays a graph showing what percentage of the input column each bin represents.



## Data

Summarizes information about each histogram, with numerical measures for:

- Bin name
- Bin number

- Bin start point
- Bin end point
- Count
- Percentage
- Cumulative counts
- Cumulative %

Results - Histogram							
Counts	name: debt_ratio ▼						
Cumulative Counts							
Percentage							
Data							
	name	bin	begin	end	count	Percentage	Cumulative Counts
	debt_ratio	1	0	0.66	44,449	88.898%	44,449
	debt_ratio	2	0.66	1.31	5,211	10.422%	49,660
	debt_ratio	3	1.31	1.97	318	0.636%	49,978
	debt_ratio	4	1.97	2.62	22	0.044%	50,000
							Cumulative %
							88.898%
							99.32%
							99.956%
							100%



**Note:** To learn more about the visualization available in this operator, see [Explore Visual Results](#).

## Data Output

None. This is a terminal operator.

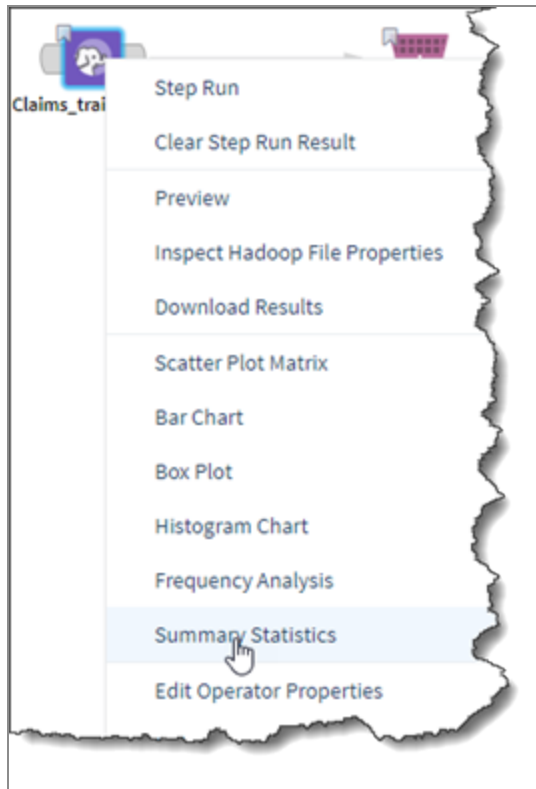
## Summary Statistics (right-click)

If an operator is associated with a data set, then you can right-click the operator and click **Summary Statistics** to select columns, and then display statistics about the selection.



**Note:** For information about the Summary Statistics operator, see the help for either [Summary Statistics \(DB\)](#) or [Summary Statistics \(HD\)](#).

The right-click menu option provides a quick view of information for the selected operator. An operator that has data associated with it (such as a data operator and transformation operators) show the **Summary Statistics** option in the right-click menu. You can quickly display summary information on any part of the data without having to configure and invoke an operator.



Unlike the Summary Statistics operator, this quick view does not specify the **Group By** option.

In the Select Columns dialog, select the checkboxes next to the columns to review.

After you select the columns for which to review summary statistics, a new window displays the tabular results for the selected fields. The table contents can be a subset from the following list.

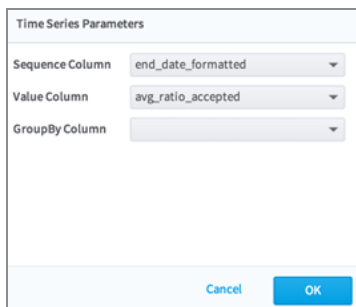
- Name.
- Data type.
- Count.
- Unique value count.
- Null value count.
- Empty value count.
- Zero value count.
- Min value.
- 25% (approx.) - Approximate 25% value for numerical columns.

- Median (approx.) - Approximate median value for numerical columns.
- 75% (approx.) - Approximate 75% value for numerical columns.
- Maximum value.
- Standard deviation.
- Average.
- Positive value count.
- Negative value count.
- Most Common (Value) - The most common value for the column.
- Most Common (Percentage) - The percentage of the total, which is the most common value.
- 2nd Most Common (Value) - The second most common value.
- 2nd Most Common (Percentage) - The percentage of the total, which is the second most common value.

## Time Series Chart

Use the Time Series Chart to visualize time series data.

- In order for Time Series Chart to be enabled, database operators must output a table/view.
- Time Series Chart is accessible from the **Explore** menu of the operator toolbar or by right-clicking an appropriate operator.



Time Series Parameters

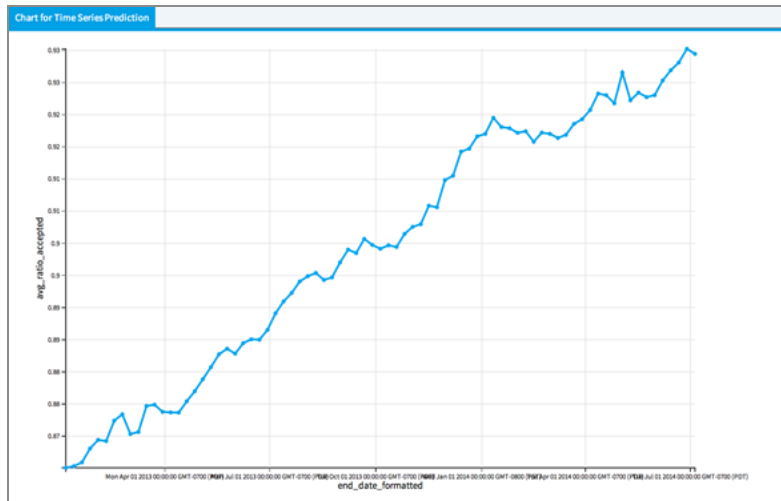
Sequence Column: end\_date\_formatted

Value Column: avg\_ratio\_accepted

GroupBy Column:

Cancel OK

- **Sequence Column** corresponds to the X-axis.
- **Value Column** corresponds to the Y-axis.



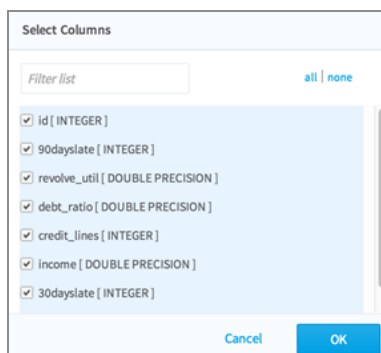
## Correlation Analysis

Correlation Analysis allows users to specify two or more numeric type attributes (columns) in a dataset for analysis.

The operator calculates the correlation between each pair of selected columns.

Use Correlation Analysis to view a correlation matrix of the specified columns.

- In order for Correlation Analysis to be enabled, database operators must output a table/view.
- Correlation Analysis is accessible from the **Explore** menu of the operator toolbar or by right-clicking an appropriate operator.



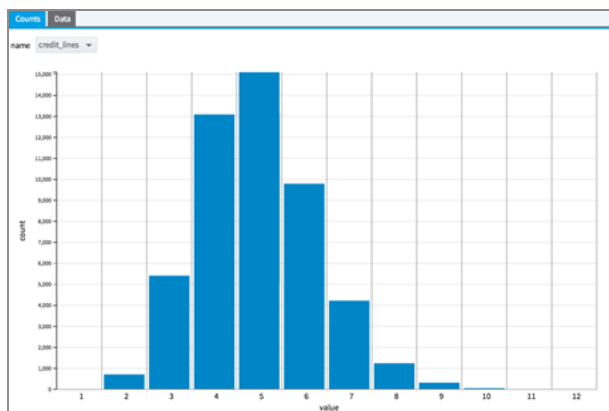


credit								
Column Name	id	90dayslate	revolve_util	debt_ratio	credit_lines	income	30dayslate	delinquent
id	1	0.0073	0.003	0.0011	-0.0078	-0.0012	0.0014	0.0069
90dayslate	0.0073	1	0.5301	0.051	-0.2888	-0.022	0.2717	0.3261
revolve_util	0.003	0.5301	1	0.0848	-0.1505	-0.0082	0.1443	0.1801
debt_ratio	0.0011	0.051	0.0848	1	-0.025	0.0055	0.0102	0.1819
credit_lines	-0.0078	-0.2888	-0.1505	-0.025	1	0.0682	-0.0748	-0.0921
income	-0.0012	-0.022	-0.0082	0.0055	0.0682	1	0.0025	-0.0202
30dayslate	0.0014	0.2717	0.1443	0.0102	-0.0748	0.0025	1	0.0906
delinquent	0.0069	0.3261	0.1801	0.1819	-0.0921	-0.0202	0.0906	1

## Frequency Analysis

Use Frequency Analysis to view frequency statistics of one or more selected columns.

- In order for Frequency Analysis to be enabled:
  - Database operators must output a table/view.
  - Hadoop operators must store results.
- Frequency Analysis is accessible from the **Explore** menu of the operator toolbar or by right-clicking an appropriate operator.
- Frequency Analysis displays the first 100 bins of the results.



Counts Data			
name credit_lines ▾			
name	value	count	Percentage
credit_lines	1	12	0.024%
credit_lines	2	714	1.428%
credit_lines	3	5,417	10.834%
credit_lines	4	13,093	26.186%
credit_lines	5	15,104	30.208%
credit_lines	6	9,795	19.59%
credit_lines	7	4,225	8.45%
credit_lines	8	1,246	2.492%
credit_lines	9	321	0.642%
credit_lines	10	60	0.12%
credit_lines	11	9	0.018%
credit_lines	12	4	0.008%

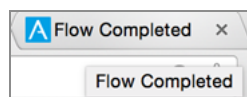
## Running a Workflow

To run a workflow, open the flow in the editor and click **Run** in the upper toolbar.

As the flow runs, the results console updates to inform you of the status of your flow.

Results - Status	
<div>Save Flow Result Publish to Workspace</div>	
[17:36:41]	Analytic Flow started running.....
[17:36:41]	credit started running.....
[17:36:41]	credit:Node result exists; skipping execution.
[17:36:41]	credit finished
[17:36:41]	demographics started running.....
[17:36:41]	demographics:Node result exists; skipping execution.
[17:36:41]	demographics finished
[17:36:41]	Table Join started running.....
[17:36:41]	Table Join:Node result exists; skipping execution.
[17:36:41]	Table Join finished
[17:36:41]	Normalization started running.....
[17:36:41]	Normalization:Node result exists; skipping execution.
[17:36:41]	Normalization finished
[17:36:41]	Random Sampling started running.....
[17:36:41]	Random Sampling:Node result exists; skipping execution.
[17:36:41]	Random Sampling finished
[17:36:41]	train started running.....
[17:36:41]	train:Node result exists; skipping execution.

If you open another browser tab while you are waiting for the flow to complete, a flashing indicator is displayed, indicating **Flow Completed** when it is finished.



- If you accidentally close the workflow or the browser tab, the workflow continues to run and you can return to it later and see the progress/output logs.
- You can open multiple flows in the same browser under one session. Each flow is in

its own window or tab.

- You can switch between tabs and have each tab keep the session active.
- You can edit a workflow, run a workflow, or check on the step-run results of a workflow that is already running.
- If you open the same workflow that is open in another browser session, a message about the existing active session is displayed.

**i Note:** The results of the successfully run flow can be saved or published to the workspace.

## Stepping Through a Workflow

The **Step Run** command executes all operators that are required to reach the selected operator. It also avoids running any operators that have not changed since the command was last run.

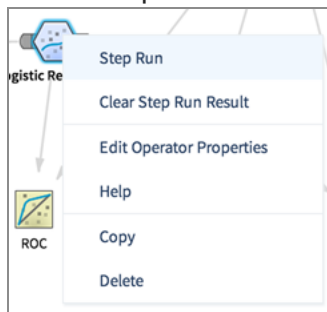
An open workflow.

### Procedure

1. To step run to a given operator, click on the operator and then on the **Step Run** link at the bottom menu toolbar.



Another option is to click on a particular operator and right-click to select **Step Run**.



All steps of the flow are processed up to the selected operator.

- If a workflow is being step run, the user can still edit parts of the workflow that are not included in the step run branch.

- Multiple operators can be selected at once for the step run processing, and only the relevant branches leading up to those operators are executed.
- The results of a step run appear below in the **Results** section. They can be saved between sessions and upon reopening a workflow.
- The step run results for an operator are not reset until that portion of the flow is run again or the parameters for an operator are changed.
- The user can exit the workflow and still return to it to find the saved step run results.
- Running a workflow after it has been step run executes only the operators that were not already processed by the step run, saving processing time and duplication.
- The step run results can be manually cleared so that part of the flow is re-executed the next time.
- Editing the parameters of an operator triggers the automatic clearing of previous step run.

## Stopping a Workflow

A workflow can be stopped at any time during processing.  
An open workflow.

To stop a running flow, click **Stop** in the upper menu toolbar.



Clicking **Stop** prevents TIBCO Data Science - Team Studio from running any further operations, but it does not stop the current operator from finishing execution in the database or Hadoop.

## Clearing a Workflow

The **Clear** command clears the entire workflow's run or step run results.  
An open workflow.

To clear a workflow, click **Clear** in the upper menu toolbar.



## Saving a Workflow

At any point, you can save a copy of a workflow, creating a new version of the workflow.

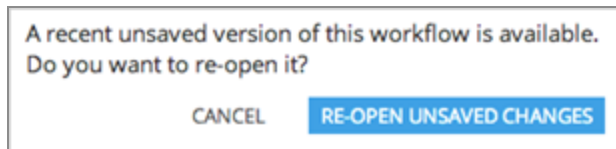


Upon saving, you can also attach comments to a particular version that can later be viewed as part of the **Flow History** (available from the **Actions** menu).

The version number is displayed next to the name of the workflow in the upper toolbar.

In the background, as the user edits a flow, TIBCO Data Science - Team Studio automatically saves a copy of the flow every time it is modified.

If the user closes the browser by mistake, or if the session times out, TIBCO Data Science - Team Studio prompts the user to restore the unsaved changes from the last session.



## Reverting a Workflow

Reverting a flow reverts the contents of the flow to the last saved version. This undoes any unsaved changes.

An open workflow.

To revert a workflow, click **Revert** in the upper menu toolbar.



## Running a Flow in Local Mode

TIBCO Data Science - Team Studio provides users the ability to run Hadoop flows in Local Mode.

Local Mode improves performance when datasets are small by running workflows locally and in memory.

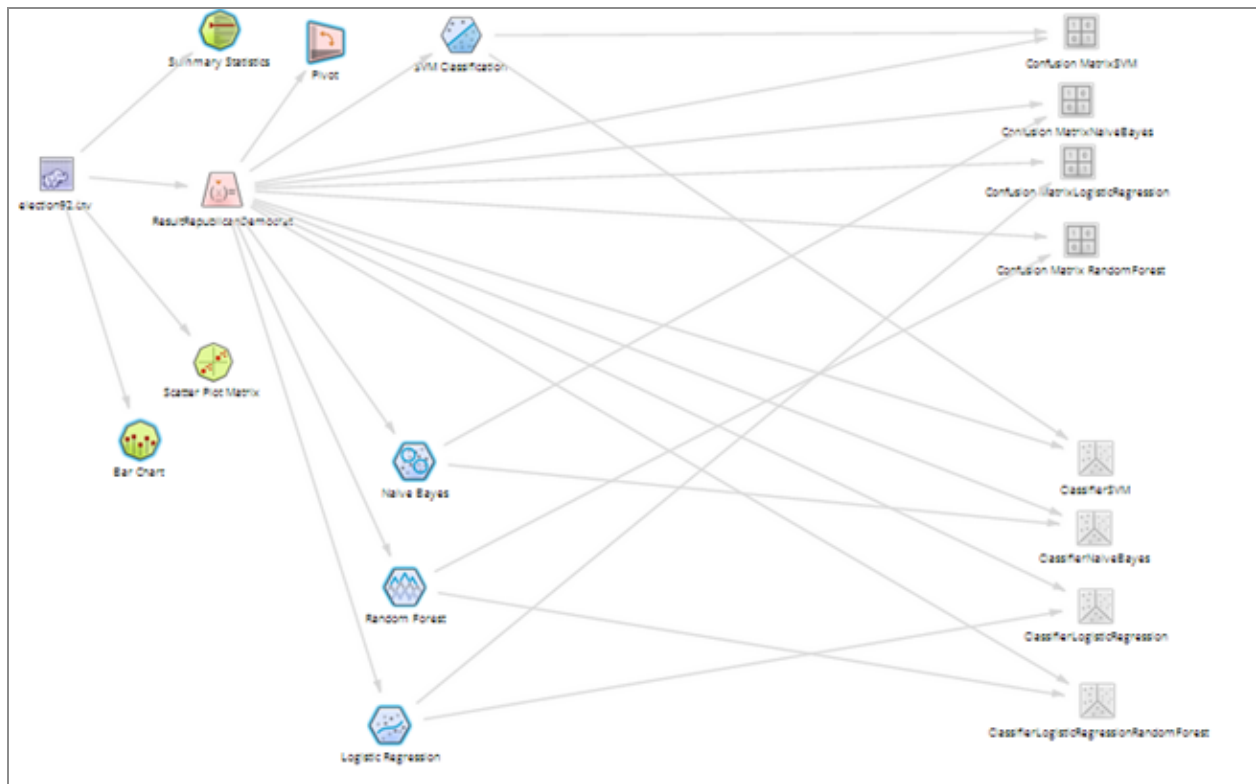
- Local Mode is turned on automatically when the sizes of the initial datasets in a flow are beneath a certain limit.
- The file size limit of running flows in Local Mode can be set by an administrator from the Preference dialog on the **User** menu.
- This limit can be set to 0 to turn off Local Mode.
- Operators running in Local Mode say (in local mode) in the result log, as shown below.

```
[16:28:44] Analytic Flow started running.....  
[16:28:44] credit.csv started running.....  
[16:28:44] credit.csv finished  
[16:28:44] demographics.csv started running.....  
[16:28:44] demographics.csv finished  
[16:28:44] Join started running.....  
[16:28:57] Join finished (in local mode)  
[16:28:57] Variable started running.....  
[16:29:04] Variable finished (in local mode)  
[16:29:04] Logistic Regression started running.....  
[16:29:24] Logistic Regression finished (in local mode)  
[16:29:24] Analytic Flow finished
```

## Running Workflow Branches in Parallel

When a workflow exceeds the threshold limit for running, it is processed by the server, using up to 12 parallel processing threads (by default).

An example of a workflow with branches running in parallel is the following:



In this example, several branches are being run in parallel, as indicated by the flashing blue border around the operators.

This design significantly decreases the processing time required for running complicated workflows.

## Handling Bad Data in Hadoop

Before processing each line of data in a MapReduce job, TIBCO Data Science - Team Studio first checks to see if the data is "clean".

A row is considered clean if, for each column being used for this operator, the corresponding value in the row is of the correct data type. If a row is dirty, it is filtered out. At the end of a run, the console indicates how many rows were filtered out due to invalid data.

```

[10:10:58] Naive Bayes started running.....
[10:11:00] NaiveBayes i
           Map:
           Reduce:
[10:11:02] NaiveBayesVisualization i
           Map:
           Reduce:
[10:11:03] Naive Bayes finished (in local mode)
[10:11:03] Analytic Flow finished
           Naive Bayes skipped 6 records, because they were empty, or contained non-numeric or null values in numeric columns.
           Click on individual operators to see their output.

```

In general, all models filter out bad data as described above. However, predictors do not filter out the bad data. Instead, predictors include the row but do not make a prediction in this case.


## Viewing Workflow Results





When a workflow is run, the **Results Console** is displayed below the **Workflow** canvas.

The screenshot displays the TIBCO Data Science - Team Studio interface. The top section shows a workflow canvas titled "Credit Prediction \*". The workflow includes operators such as "credit", "demographics", "Table join", "Normalization", "Random Sampling", "train", "validate", "Logistic Regression Prediction", "Logistic Regression", "LIFT", "ROC", and "Goodness of Fit". The bottom section shows the "Results - Status" console, which displays a list of log messages indicating the execution progress of the workflow, including messages like "Analytic Flow started running", "credit started running", "credit finished", "demographics finished", "demographics node result exists, skipping execution", "Table join started running", "Table join finished", and "Normalization started running".

By default, the **Results Console** is displayed as 1/3 of the screen.

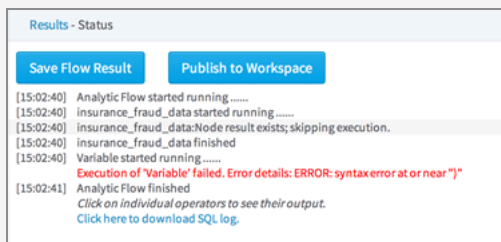


**Note:** The results display control icons , can change the **Results Console** view between **1/3rd view**, **2/3rd view**, or **separate window** view.

- The  icon minimizes the results.
- The  icon either increases the view to 2/3rd view or the  icon reduces it to 1/3rd view.
- The  icon opens the results in a separate window.

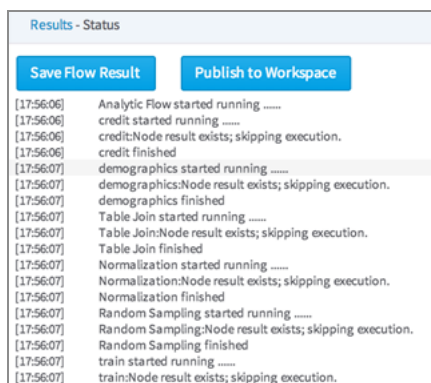
Click on the [Results - Status](#) link in the bottom left toggles the **Results Console** on and off.

In addition to the results log displaying all workflow processing details, any data issues encountered (for Hadoop datasets) are noted in red, as shown in the following example.



## Saving Flow Output

After running a flow, the results of the flow are displayed in a new tab/window. The user can choose to save these results as a permanent HTML file or publish them to the workspace.



## Save Flow Results

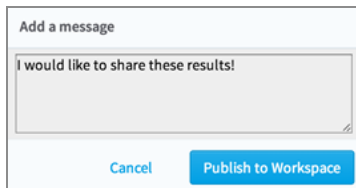
In the **Results** window, click **Save Flow Result** to export the results as an HTML file.

All of the results are provided for download in a .zip archive.

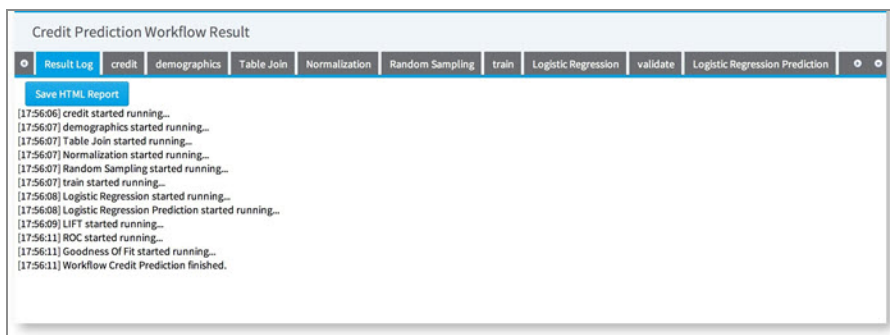
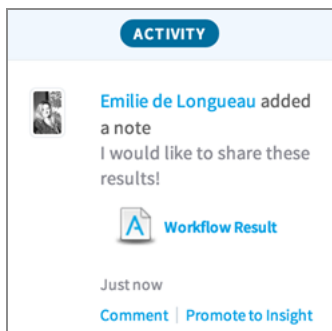
## Publish to Workspace

Publishing results to the workspace makes them accessible to the entire workspace.

The user can add a message to the saved results.



The results can then be accessed by anyone within the workspace.



## Results Management

The results of executing a workflow can be saved as part of the history of the workflow, accessible from the **Your Workflow Results** menu item on the **Actions** menu.

Workflow results are saved under the following circumstances:

- The most recent execution (or partial execution from a step run) is saved temporarily and indicated by an asterisk in the list of **Workflow Results**.
- After running or step-running a workflow, you can save the results permanently by clicking **Save Flow Results** in the **Results** panel.
- The most recent execution is saved permanently when closing a workflow.
- The results are saved when running a workflow from outside the workflow editor, by clicking **Run Now** in the **Workfile Actions**.
- The results are saved when running a workflow as part of a scheduled job.
- The results are saved when running a workflow from the TIBCO Data Science - Team Studio API.

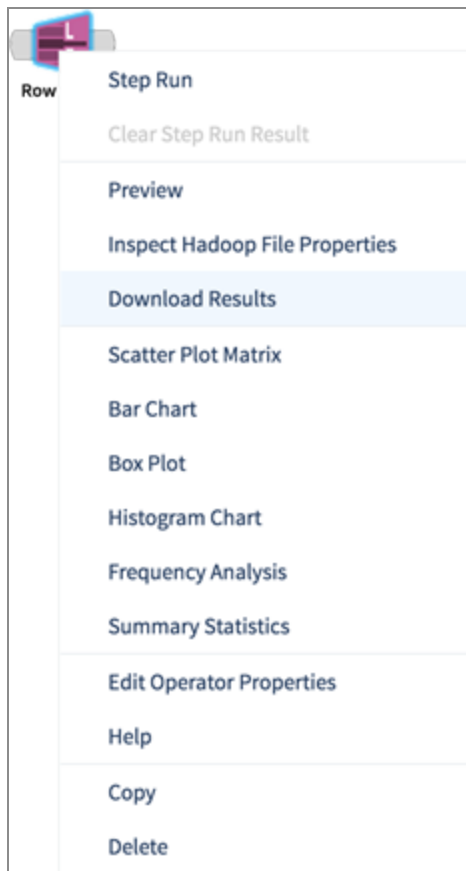
TIBCO Data Science - Team Studio workflow results can be managed and stored in various ways. For more information, see [Downloading Results](#).

## Downloading Results

When **Store Results?** is set to **true** for a particular operator, the user can select that operator and download the data results at any time (provided the flow has already been run).

### Downloading Operator Stored Results

To download results, right-click on the operator and select **Download Results**:



When the download is initiated, a dialog displays additional information about the results. Before the download begins, the user can choose whether to include header row in the download file.

**Download Results**

Name: numbers2.csv

include header row: ☐

Start at line:

Number of lines:

Close Download

**Download Results**

Name: Dataset

Number of lines:

Close Download

**i Note:** If the **Number of lines** is set to 0 and the **include header row** option is selected, the results contain the header information and no data.

	A	B	C	D	E	F	G	H
1	id	times90dayslate	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslate_2years	srsdlqncy
2								

## Viewing Database SQL

After running a flow, the workflow log is displayed below in the **Results** console, along with an option to download and view any executed SQL code.

Results - Status	
[18:10:54]	random sampling started
[18:10:54]	train started running.....
[18:10:54]	train finished
[18:10:55]	validate started running.....
[18:10:55]	Logistic Regression started running.....
[18:10:55]	validate finished
[18:10:55]	Logistic Regression finished
[18:10:56]	LIFT started running.....
[18:10:57]	ROC started running.....
[18:11:01]	Logistic Regression Prediction started running.....
[18:11:10]	LIFT finished
[18:11:10]	Goodness Of Fit started running.....
[18:11:14]	ROC finished
[18:11:19]	Logistic Regression Prediction finished
[18:11:21]	Goodness Of Fit finished
[18:11:21]	Analytic Flow finished
Click on individual operators to see their output.	
<a href="#">Click here to download SQL log.</a>	

To download the SQL log, click **Click here to download the SQL.log**.

You can view or save any database SQL code that was executed for the workflow.

```

1 SELECT * FROM ( select * from "demo"."credit") foo WHERE 0 = 1;
2 drop table "demo"."dic" CASCADE;
3 create TABLE "demo"."dic" as ( select "demo_credit_a"."times90dayslate" as
"times90dayslate","demo_credit_a"."revolving_util" as "revolving_util",
"demo_credit_a"."debt_ratio" as "debt_ratio","demo_credit_a"."credit_lines" as
"credit_lines","demo_credit_a"."monthly_income" as "monthly_income",
"demo_credit_a"."times30dayslate_2years" as "times30dayslate_2years",
"demo_credit_a"."srsdlqncy" as "srsdlqncy","demo_demographics_a"."id" as "id",
"demo_demographics_a"."age" as "age","demo_demographics_a"."num_dep" as "num_dep",
"demo_demographics_a"."edu" as "edu" from "demo"."demographics"
"demo_demographics_a" JOIN "demo"."credit" "demo_credit_a" on (
"demo_demographics_a"."id" = "demo_credit_a"."id") ) ;
4 SELECT * FROM ( select * from "demo"."dic") foo WHERE 0 = 1;
5 select avg((1.0*( "credit_lines" ))),variance((1.0*( "credit_lines" ))) from "demo".
"dic";

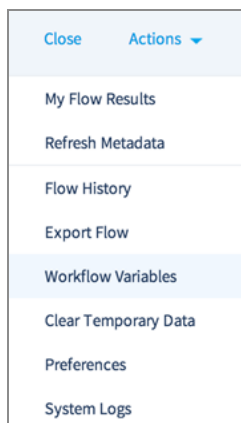
```

St length : 81324 lines : 84 Ln:1 Col:1 Sel:0|0 UNIX ANSI as UTF-8 INS

## Workflow Variables

It is possible to override the default workflow parameters using workflow variables.

- This practice fine-tunes the parameters for only the specified workflow.
- The value of the workflow variable is substituted appropriately at run time, which provides more flexibility at design time.
- To view a workflow's variables, on the **Actions** dropdown list box, click **Workflow Variables**.



## Default Workflow Variables

Every workflow contains default variables, and each variable begins with the @ symbol:

 A screenshot of a 'Workflow Variable Settings' dialog box. It contains a table with two columns: 'Variable' and 'Value'. The table lists several variables with their corresponding values. To the right of each row is a 'delete' button. At the bottom right of the dialog are 'Cancel' and 'OK' buttons.
 

Variable	Value	
@alpine.mapred.mapred.tasktimeout	30000	delete
@alpine.mapred.join.Hadoop_Join.mapr	20000	delete
@var	1=1	delete
@good_value	1	delete
@logistic_reg_parameter	4	delete
@flow_name	USFullCensus_With_Categories_588	
@user_name	emilie	
@default_schema	public	

- *@default\_schema*: used as the default schema for database output
- *@default\_tempdir*: used as the default directory for intermediate Hadoop files
- *@default\_prefix*: used as the default prefix for output tables and output files

- `@default_delimiter`: used as the default data delimiter for CSV and Hadoop files.

**i Note:** The default variables cannot be deleted, but their values can be changed to override the global defaults at the workflow level.

## Spark Workflow Variables

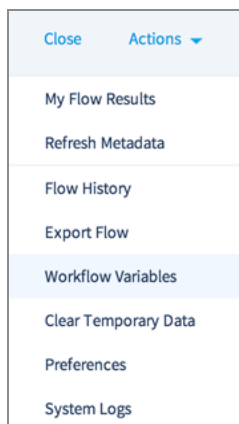
We do not support setting Spark variables on a workflow-wide level. It is a Spark best practice to optimize the Spark parameters per job or operator that you want to run. To make this process easier, we created [Spark Autotuning](#) to help you choose the most performant values for your Spark job settings. See also [Spark Optimization for Data Scientists](#) for more information.

## Defining New Workflow Variables

In addition to the default variables that every workflow contains, you can also create new workflow variables.

### Procedure

1. To define a workflow's variables, on the **Actions** dropdown list box, click **Workflow Variables**.



For flows that contain a sub-flow, the variables of the sub-flow are also displayed in the dialog shown below.

- The user can only change the value of sub-flow variables.
- If a variable in the sub-flow has the same name as a variable in the parent flow,

the user is warned that the value would be overridden.

The dialog box titled "Workflow Variable Settings" contains a table with two columns: "Variable" and "Value". To the right of each row is a "delete" button. At the bottom right are "Cancel" and "OK" buttons.

Variable	Value	
@alpine.mapred.mapred.task.timeout	30000	delete
@alpine.mapred.join.Hadoop_join.mapr	20000	delete
@var	1=1	delete
@good_value	1	delete
@logistic_reg_parameter	4	delete
@flow_name	USFullCensus_With_Categories_588	
@user_name	emilie	
@default_schema	public	

2. To create a new workflow variables, click **create**.
3. In the **Variable** field, enter the variable name.
4. In the **Value** field, enter the value of the variable.

**Note:** When defining a workflow variable for a table that contains period/dot ( . ) in its name, then the variable should be wrapped in " ". For example, "**demographics.parquet**".

5. Click **OK** to save the workflow variables, or click **Cancel** to discard changes.

## Removing Variables

To remove a variable, select it and click **Delete**.

## Overriding Hadoop Data Source Parameters Using Workflow Variables

It is possible to override the default Hadoop data source parameters using the workflow variable settings.

This fine-tunes the Hadoop data source settings for only the specified workflow.



The image shows a 'Workflow Variable Settings' dialog box. It contains a table with two columns: 'Variable' and 'Value'. There are five rows of user-defined variables, each with a 'delete' button to its right. Below the table are three rows of system-defined variables. At the bottom right are 'Cancel' and 'OK' buttons.

Variable	Value	
@alpine.mapred.mapred.task.timeout	30000	delete
@alpine.mapred.join.Hadoop_Join.mapred.task.timeout	20000	delete
@var	1=1	delete
@good_value	1	delete
@logistic_reg_parameter	4	delete
@flow_name	USFullCensus_With_Categories_568	
@user_name	emilie	
@default_schema	public	

- To view a workflow's variables, select **Workflow Variables** from the **Actions** dropdown list box.
- To create a new variable for overriding a default Hadoop setting (such as the amount of time before a task times out), click **create**.
- The default is 600,000 ms (or 10 minutes).
- To override a Hadoop variable for a specific workflow, click **create** and make a new variable called, for example, *@alpine.mapred.mapred.task.timeout*, where
  - *@alpine.mapred.* indicates it is the TIBCO Data Science - Team Studio override for a Hadoop variable and *mapred.task.timeout* is the official Hadoop variable name.
  - Set the value to 300000, for example.
- To override a Hadoop variable only for a specific operator task within a workflow, create a new variable called, for example, *@alpine.mapred.join.Hadoop\_Join.mapred.task.timeout*, where
  - *@alpine.mapred.* indicates it is the TIBCO Data Science - Team Studio override for a Hadoop variable,
  - *join* indicates it is for the Join operator,
  - *Hadoop\_Join* indicates the particular operator job that is being overridden, and
  - *mapred.task.timeout* is the official Hadoop variable name.
  - Set the override value (for this workflow's Join operators only) to 200,000, for example.

**i Note:** Note: Any of the possible Hadoop configuration parameters can be configured either here at the workflow level or at the Hadoop data source level. Here is a list of Hadoop configuration parameters:

- [mapred-default](#)
- [core-default](#)
- [hdfs-default](#)

**i Note:** For customizing specific operator tasks, the correct TIBCO Data Science - Team Studio operator name and job name must be referenced. Here is a list of TIBCO Data Science - Team Studio operators and job names:

[TIBCO Data Science - Team Studio Operator Job Names](#)

## TIBCO Data Science - Team Studio Operator Job Names

The following is a list of TIBCO Data Science - Team Studio job names that can be referenced for overriding specific Hadoop MapReduce data source parameters only for particular operator tasks.

**i Note:** Workflow variables created to reference these values apply to MapReduce jobs only, not to Pig, Spark, or Sqoop jobs.

- *SummarizerJob*
- *PCAJob*
- *PCA\_Q\_Job*
- *PCA\_Bt\_Job*
- *PCA\_Iter\_ABt\_Job*
- *PCA\_Generating\_PCs\_Job*
- *Convert\_Input\_Data\_to\_Compressed\_Sequence\_File*
- *SVM\_Prediction*
- *TimeSeries\_Sort*

- *DecisionTree\_Depth*
- *DecisionTree\_Parse\_Splits*
- *Goodness\_Of\_Fit*
- *Kmeans\_Post*
- *Kmeans\_Init*
- *Hadoop\_Union*
- *Hadoop\_Join*
- *Hadoop\_Distinct*
- *Hadoop\_Join\_preProcess*
- *Kmeans\_Iteration*
- *Kmeans\_Output*
- *Max\_Min\_Job*
- *Lift\_DataGenerator*
- *LinearRegression\_Predictor*
- *LinearRegression*
- *LinearRegression\_QQ*
- *LinearRegression\_Statistics*
- *LinearRegression\_Beta*
- *FeatureExtractor*
- *LogisticRegression\_Iterator*
- *logisticRegression\_One\_Pass*
- *ROC\_DataGenerator*
- *Distinct\_Job*
- *VariableSelection\_AlphaBetaCoefficients*
- *VariableSelection\_R2*
- *NaiveBayes*
- *ConfusionMatrix*

- *NaiveBayesVisualization*
- *NaiveBayesInitialJobForSparseData*
- *InformationGainVariableSelection*
- *Alpine\_Forest\_InMemory*
- *pivot*
- *ForestOOB*
- *Collapse*
- *PCA\_Apply*
- *Correlation*
- *Classifier\_Predictor*

## Determining Operator Job Names

Each operator's job name can be seen in the **Results** window while a workflow is running. These are the job names that can be referred to for variable settings overrides for Hadoop data source connections.

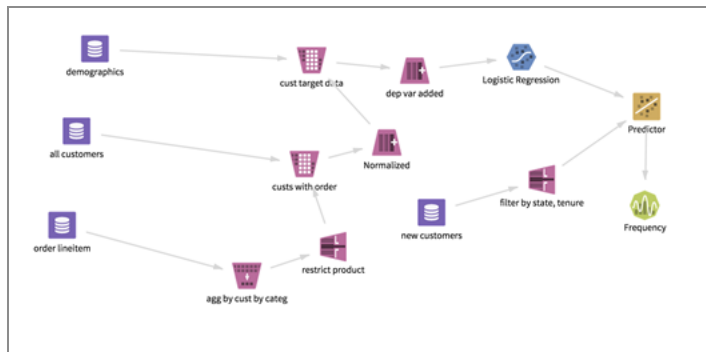
The following example displays the **Results** window for a K-Means workflow running various Hadoop MapReduce jobs (AlpineKmeans\_init, AlpineKmeans\_Iteration, AlpineKmeans\_Output, and AlpineKmeans\_Post).

[15:00:04]	Analytic Flow started running .....	
[15:00:04]	iris.txt started running .....	
[15:00:05]	iris.txt finished	
[15:00:05]	K-Means started running .....	
[15:00:06]	K-Means : <a href="#">AlpineKmeans_Init</a>	
	Map:	<div></div>
	Reduce:	<div></div>
[15:00:22]	K-Means : <a href="#">AlpineKmeans_Iteration_1</a>	
	Map:	<div></div>
	Reduce:	<div></div>
[15:00:37]	K-Means : <a href="#">AlpineKmeans_Iteration_2</a>	
	Map:	<div></div>
	Reduce:	<div></div>
[15:00:53]	K-Means : <a href="#">AlpineKmeans_Iteration_3</a>	
	Map:	<div></div>
	Reduce:	<div></div>
[15:01:07]	K-Means : <a href="#">AlpineKmeans_Output</a>	
	Map:	<div></div>
	Reduce:	<div></div>
[15:01:17]	K-Means : <a href="#">AlpineKmeans_Post</a>	
	Map:	<div></div>

# Touchpoints

Data scientists and engineers can use touchpoints to create user-friendly applications that harness TIBCO Data Science - Team Studio analytic workflows without any programming. Front-line business users can then use these interactive self-service tools to get the answers they need, when they need them.

With a touchpoint, you can turn a predictive model or data workflow.



...into an easy-to-use form for the business user.

first_name	last_name	city	state_code	country	customer_age	tenure
Ethan	Skinner	TOPAZ	CA	USA	28	4
David	Fasano	ONTARIO	CA	USA	34	2
Anthony	Ritter	REDDING	CA	USA	27	3
Joseph	Higbee	SCOTIA	CA	USA	28	3
Christian	Eskridge	REDLANDS	CA	USA	61	2
Nevaeh	Keating	CARMICHAEL	CA	USA	31	2
Madison	Jossey	LOS ANGELES	CA	USA	35	3
Ella	Peter	EL NIDO	CA	USA	27	2
Ella	Basile	MC FARLAND	CA	USA	32	3
Grace	Schall	OAKLAND	CA	USA	35	3
Hannah	Tice	KELLY	WY	USA	44	2
Hannah	Householder	CHEYENNE	WY	USA	52	2
Michael	Batista	ROZET	WY	USA	26	4
Sophia	Bolt	SANTA CRUZ	CA	USA	33	4
Ella	Quist	HARBOR CITY	CA	USA	25	2
Ashley	Topete	SAN FRANCISCO	CA	USA	26	2
Jonathan	Hardin	SAN JOSE	CA	USA	30	2

Touchpoints help people across the enterprise organization make better data-driven decisions.

- Business users can find new sales leads, users at risk of leaving, or most popular

markets using a touchpoint that runs trained models against new and custom datasets. A data analyst can use touchpoints to harness the power of TIBCO Data Science - Team Studio workflows in a modular, interactive format.

- An analytics developer or data scientist can use touchpoints to test out their models and tweak parameters easily.

To learn how to make your first touchpoint, see [Creating a Touchpoint](#).

## Creating a Touchpoint

In this tutorial, learn how to build, run, output, and publish results from your first touchpoint.

Creating a touchpoint begins with choosing a workflow. For this tutorial, use an example workflow. When you finish, you have a touchpoint that takes in user input and displays rows with fewer than that number of dependents. Then publish the touchpoint to the catalog, which is an organization-wide collection of touchpoints that business users can explore. To get started, make a workflow variable to hold the user input.



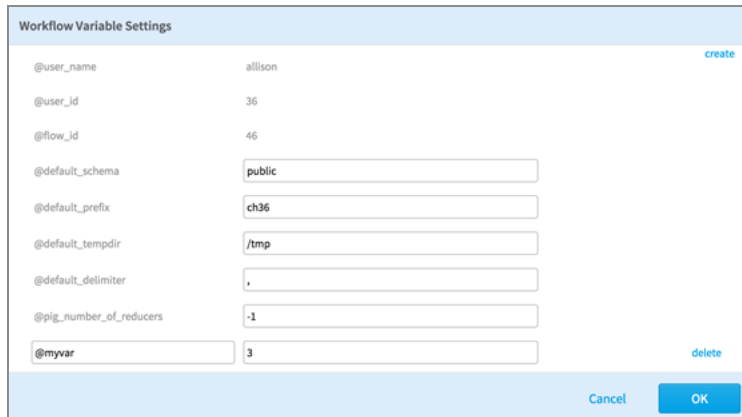
### Procedure

1. Create a variable called `@myvar` and set the default value to 3.

For more information on creating a workflow variable, see [Defining New Workflow Variables](#).



**Note:** All workflow variables must start with the character @.

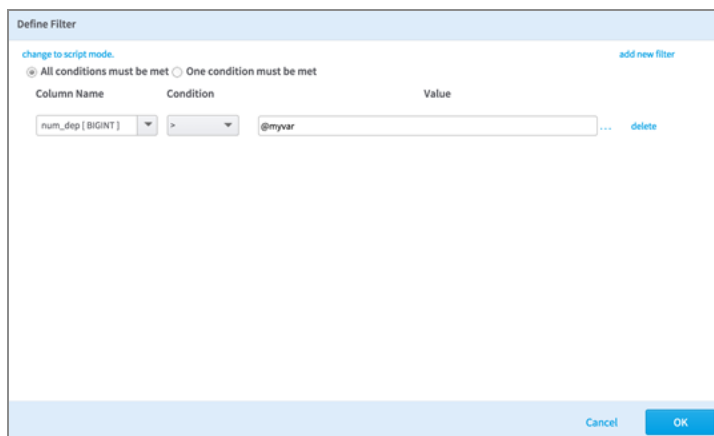


The 'Workflow Variable Settings' dialog box contains the following variables and values:

Variable	Value
@user_name	allison
@user_id	36
@flow_id	46
@default_schema	public
@default_prefix	ch36
@default_tempdir	/tmp
@default_delimiter	,
@pig_number_of_reducers	-1
@myvar	3

Buttons: create, delete, Cancel, OK

2. Set up the Row Filter operator to take input from the workflow variable.




The 'Define Filter' dialog box shows the following configuration:

- Mode: ☒ All conditions must be met (Other option: One condition must be met)
- Column Name: num\_dep [BIGINT]
- Condition: >
- Value: @myvar

Buttons: add new filter, delete, Cancel, OK

3. Navigate to the workflow you are interested in. Select the workflow, and then from the menu, click **Create Touchpoint**.



The 'Work Files' dialog box displays a list of workflows:

- touchpoint\_demo (selected)
- collabtest
- touchpoint
- tp2
- tp3


Buttons on the right: Upload a File, Create a SQL File, Create New Workflow, Run Now, Stop Running, **Create Touchpoint** (highlighted), View Touchpoint Usage, Add a Note, Add/Edit Tags, Rename, Copy, Delete.

The **Create a New Touchpoint** dialog is displayed.

**CREATE A NEW TOUCHPOINT** ✕

Create a new touchpoint with the selected Workflow.

**Workflow Details**

 **Name:** touchpoint\_demo  
**Last modified:** 08-13-2015 13:17:04

**Touchpoint Details**

**Name \***

**Description**

Give a general description of the Touchpoint and what it should do.

\* Required field

Cancel Continue

4. Enter the name and a short description for the touchpoint, and then click **Continue**.

**Note:** Valid names for touchpoints include numbers, letters, spaces, and any of the following characters: ( ) . - \_  
 Work files with other characters in the name are not created.

## What to do next

Specify the [Run Settings Tab](#) and [Adding Parameters to a Touchpoint](#) to your touchpoint.

## Run Settings Tab

Run settings determine how a touchpoint is run and what output is displayed.

View the run settings from the **Run Settings** tab.



### Run Settings

The run settings determine how the touchpoint is run and what output will be displayed.

#### Data Credentials


Specify the account that will be used to run the Touchpoint. The account must have credentials to access the source data used in the workflow, or else the analytics will be unable to run.

☒ Run using the Touchpoint creator's account (i.e. owner)  
☐ Run using the account of the end user of the Touchpoint (i.e. the person running it)

#### Output

##### Output to Display

Select which workflow output should be displayed to the end user.



```

graph LR
    demographics[demographics] --> RowFilter[Row Filter]
          
```

## Data Credentials

You can run a touchpoint as the owner or as an end user. Whichever you choose, the account must have permission to access the source data in the workflow. This is very important for users accessing Kerberized data sources. Otherwise, the analytics are unable to run.

**Note:** Be sure to select an output and save your touchpoint before moving on.

## Adding Parameters to a Touchpoint

To allow interactivity for the users of your touchpoint, set parameters for user input.

### Before you begin

You must have created a touchpoint, or you must have permission to edit a touchpoint.

To add a parameter, click **Add First Parameter**.

touchpoint\_demo

WORKFLOW
DETAILS
RUN SETTINGS
INPUTS

### Configure Touchpoint

#### Add the Touchpoint Inputs

Add and configure each of the input parameters that will be used to run the Touchpoint.

Add First Parameter

The screenshot shows a configuration window titled "Parameter 1". It contains several fields: "Workflow Variable" with a dropdown menu showing "@myvar", "Use Workflow's existing value as default" (unchecked), "Required Field" (checked), "Field Label" with the text "num\_dependents", "Description" with the text "display only people that have less than this number of dependents.", and "Data Type" with a dropdown menu showing "Number". On the right side, there is a "Variable Details" panel showing "Default Value" as "3".

Configuration options

<b>Workflow variable</b>	<ul style="list-style-type: none"><li>• Required.</li><li>• Comes from the section <a href="#">Workflow Variables</a>.</li><li>• Holds the user's input.</li><li>• <b>Use workflow's existing value as default:</b> By default, the touchpoint uses the value of the workflow variable as defined in the workflow. If the user does not choose an input, this value is used. You can see the value from the workflow in the box to the right labeled <b>Variable Details</b>.</li></ul>
<b>Field Label</b>	<ul style="list-style-type: none"><li>• Required.</li><li>• A user-friendly label for the input.</li></ul>
<b>Description</b>	<ul style="list-style-type: none"><li>• Optional.</li><li>• A short string that describes the purpose of the input.</li></ul>
<b>Data Type</b>	<ul style="list-style-type: none"><li>• Required.</li><li>• The type of data that to be input.</li><li>• Types allowed: <b>Text</b>, <b>Multiline Text</b>, <b>Number</b>, <b>Select Single Option</b>, <b>Select Multiple Options</b>, and <b>Date/time - Calendar</b>.<ul style="list-style-type: none"><li>◦ <b>Text:</b> a string.</li><li>◦ <b>Multiline Text:</b> a multiline string, with a larger text area to type into.</li><li>◦ <b>Number:</b> a numeric value.</li><li>◦ <b>Select Single Option:</b> create a dropdown box where a user can make a single choice.</li><li>◦ <b>Select Multiple Options:</b> create a series of checkboxes where</li></ul></li></ul>

the user can select multiple options.

- **Date/time - Calendar:** a calendar widget in which a user can choose a custom date (for example, to show all customers in a chosen time period).
- For more information about these options, see [Touchpoint Parameters](#).

## Editing Existing Parameters

On the right-hand preview panel, click the location icon to go to that parameter in the configuration panel, and then click and drag on the four-arrow icon to change the order of the parameters.

The screenshot shows a configuration panel with two parameters.   
Parameter 1 is a 'Number' type with a value of '10' and a description 'Supports numbers.'   
Parameter 2 is a 'Multiline Text' type with a value of 'here is the default text' and a description 'Supports longer, multiline strings.'   
Each parameter header has a four-arrow icon and a location icon.

## Testing a Touchpoint

After you have set up the inputs and outputs, test the touchpoint and preview the results. This gives you an opportunity to ensure that it works the way you want it to and that the results are what you want the end user to see.

Be sure you wait for it to finish; if you close the **Run** window while it is in progress, the test run stops.

Your results appear in a window where you can inspect or download them. If a note is present on an output operator, an asterisk appears next to the operator name. Hover on the asterisk to display the note.

**i Note:** If the test run produces no results, be sure that you have selected an output from the **Run Settings** tab and saved your changes.

TEST RESULTS FROM TOUCHPOINT

Row Filter

Visualization of rows limited to 20. Please refer to output file for full results

id	name	age	num_dep	edu
7	Kari Ball	38	2	2
61	Julia Ramsey	24	1	3
91	Roger Tate	25	0	3
139	Katelyn Rowe	28	1	2
153	Richard Mills	33	1	5
161	Kristy Hernandez	44	1	2
173	Gregory Jordan	21	0	1
183	Jordan McDaniel	25	0	2
185	Rachel McKinney	29	0	6
189	Garrett Franklin	26	2	2
207	Claudia Burns	41	1	1
217	Jack Moss	44	0	3
219	Troy Adams	23	1	2

Download Results

Close Window

## Downloading Touchpoint Test Results

Touchpoint test results appear in a window where you can inspect or download them.

To download results, click **Download Results** in the **Test Results from Touchpoint** dialog.

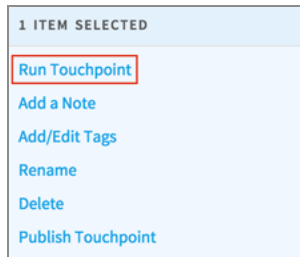
You can also download results from data tables as a .csv file.

## Running a Touchpoint

Once you've tested your touchpoint and are happy with the results, you can run it from the workspace screen or by clicking **Run** in the upper right of the touchpoint configuration screen.

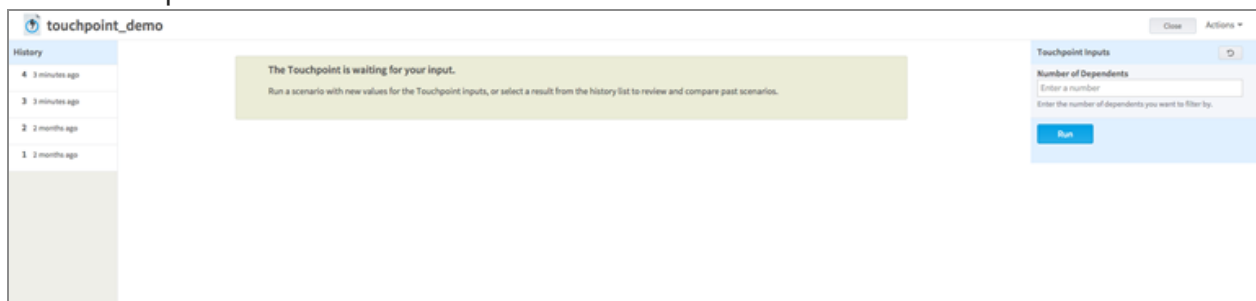
Running a touchpoint is optimized so that it runs only the operators necessary to produce your chosen results.

**Note:** Right now, only people who are members of your workspace can run the touchpoint you created. If you want people across your organization to be able to run the touchpoint, consider [publishing](#) it.

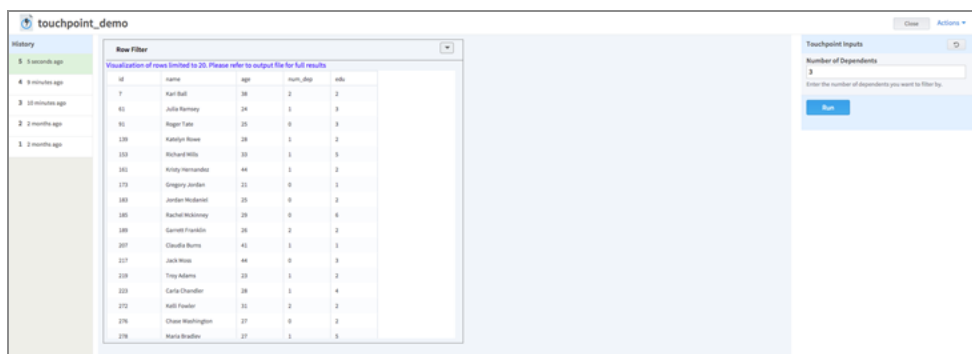


The run window is split into three parts: **History**, **Outputs**, and **Inputs**.

- **History** shows the history of this touchpoint's previous runs. You can see which parameters and outputs were used in previous runs. This is useful for adjusting parameters or running similar jobs.
- **Outputs** is the main window where outputs (tables, graphs, etc.) from the touchpoint are displayed.
- **Inputs** contains the touchpoint inputs. This is where you interact with the touchpoint.



Each time you run the touchpoint, a new entry is recorded and saved in the history panel on the left. Select an entry to display the results of that particular run. You can download the results as well. Click the pencil icon on the history entry to add a description for that particular run.



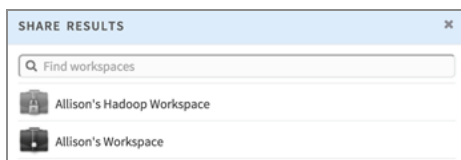
## Action Menu

You can select **Actions** from the touchpoint run screen to view the result log, download the results as an HTML report, or place the results in a separate tab. For a published touchpoint, you can also choose to share the results with another workspace you are a member of.

### Share These Results

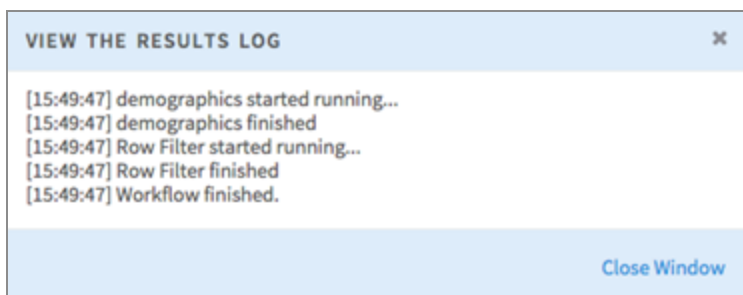
**Share these results** allows you to share the results of this touchpoint run with another workspace. You must be a member of the workspace to share with them.

**Note:** This option is available only for published touchpoints.



### View Results Log

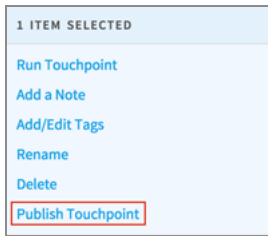
**View the results log** shows the workflow log from running the underlying operators. The log shows only the operators that were run leading up to the operator selected for the touchpoint, skipping other branches and leaf operators. If there are any errors, the workflow log reflects the same. This view is useful for troubleshooting.



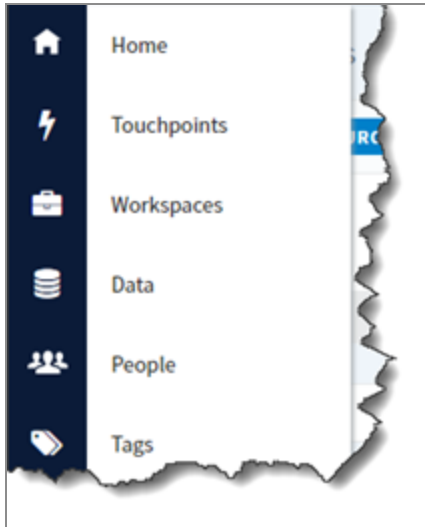
### HTML Report

**Save report (HTML)** downloads a .zip file that includes a folder of resources and an index.html. Here is an example:





2. After you confirm your choice, the touchpoint is available in the catalog. Navigate to the catalog from the touchpoints link in the menu.



From the catalog, users can run touchpoints, download results, and discover new touchpoints created by other users.

## Touchpoint Parameters

There are multiple types of user input that can be defined for a touchpoint. Besides accepting plain text and number input boxes, single-select, multiple-select, or date/time inputs are also available.

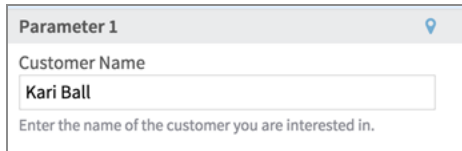
**Important:** When a user inputs data into a touchpoint, that data is not changed in any way before it is passed to the workflow variable, so the formats must match.

Be sure to review how the workflow variables are used in the analytic workflow and adapt the formatting for the touchpoint parameters accordingly. The touchpoints application does not modify the input passed to the workflow.



## Text Touchpoint Parameter

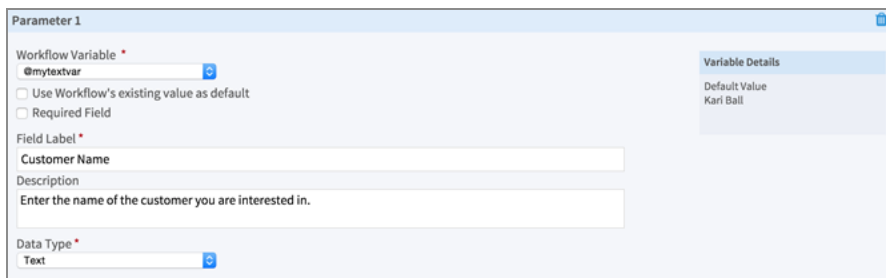
The text input appears as a text form field to the end user. It accepts up to 64 characters as input.



The screenshot shows a form titled "Parameter 1" with a location pin icon. It contains a text input field labeled "Customer Name" with the value "Kari Ball". Below the input field is a description: "Enter the name of the customer you are interested in."

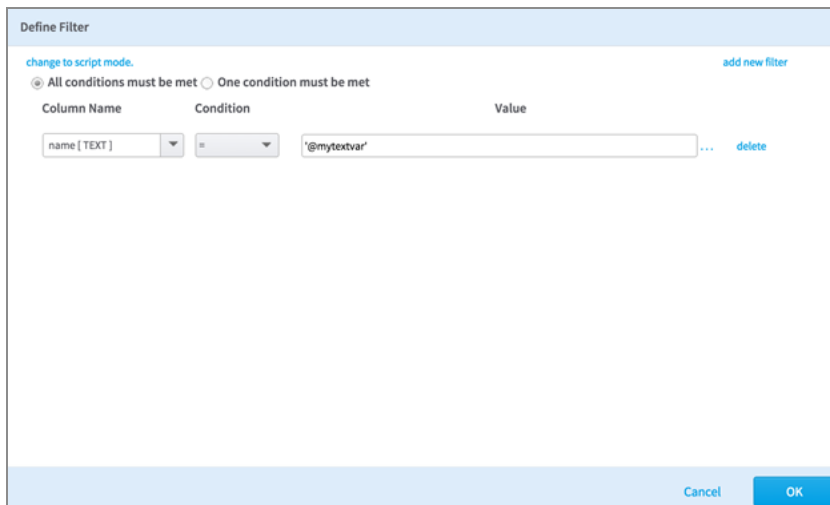
In the example below, the user can enter the name of a customer he or she is interested in learning more about. The touchpoint then filters the data set for all rows that contain that name.

In the touchpoint configuration:



The screenshot shows the "Parameter 1" configuration dialog. It includes fields for "Workflow Variable" (set to "@mytextvar"), "Field Label" (set to "Customer Name"), and "Description" (set to "Enter the name of the customer you are interested in."). The "Data Type" is set to "Text". A "Variable Details" panel on the right shows the "Default Value" as "Kari Ball".

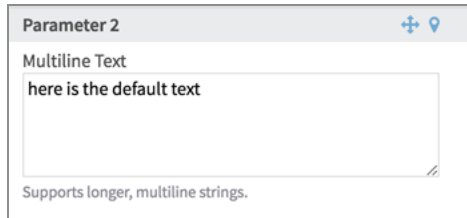
In the workflow where the variable is used, the person who is building the workflow adds the quotes to the variable at the workflow level. After the value of Kari Ball is passed to the variable `@mytextvar`, the query reads `name = 'Kari Ball'`.



The screenshot shows the "Define Filter" dialog. It has a tab for "change to script mode." and a section for "All conditions must be met" (selected). Below this is a table with columns "Column Name", "Condition", and "Value". The table contains one row: "name [ TEXT ]", "=", and "'@mytextvar'". There are "add new filter" and "delete" buttons. At the bottom are "Cancel" and "OK" buttons.

## Multiline Text Touchpoint Parameter

The text input appears as a text area field to the end user. It accepts up to 64 characters as input. There is no limit to the size of this input.



The screenshot shows a configuration window titled "Parameter 2". It features a "Multiline Text" label above a large text area containing the text "here is the default text". Below the text area, a note states "Supports longer, multiline strings.".

In the touchpoint configuration:

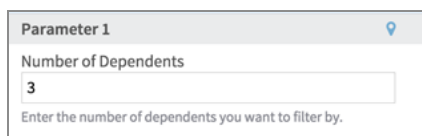


The screenshot shows the "Parameter 2" configuration window in a more detailed touchpoint configuration mode. It includes a "Workflow Variable" dropdown set to "@text", a checked "Use Workflow's existing value as default" checkbox, and an unchecked "Required" checkbox. The "Field Label" is set to "Multiline Text", with a description: "The field label is the name of the variable that will be displayed to the person using the Touchpoint." The "Description" field contains the text "Supports longer, multiline strings." The "Data Type" is set to "Multiline Text". A "Variable Details" panel on the right shows the "Default Value" as "here is the default text".

Workflow usage is identical to the Text parameter above.

## Number Touchpoint Parameter

The number input appears as a text form field to the end user. It accepts up to 20 digits as input.



The screenshot shows a configuration window titled "Parameter 1". It features a "Number of Dependents" label above a text input field containing the number "3". Below the input field, a note states "Enter the number of dependents you want to filter by.".

In the touchpoint configuration:

**Parameter 1**

Workflow Variable \*  
@myvar

☐ Use Workflow's existing value as default  
☐ Required Field

Field Label \*  
Number of Dependents

Description  
Enter the number of dependents you want to filter by.

Data Type \*  
Number

**Variable Details**  
Default Value  
3

In the workflow where the variable is used:

**Define Filter**

[change to script mode.](#) [add new filter](#)

☒ All conditions must be met ☐ One condition must be met

Column Name	Condition	Value
num_dep [INTEGER]	=	@myvar

... delete

Cancel OK

## Single-Select Option Touchpoint Parameter

The single-select option input appears as a dropdown box where the user can choose one option.

**Parameter 2**

Product Category \*  
Stocks

Select the product category for which you want to find the clients most likely to be interested.

**Parameter 2**

Select...  
Bonds  
✓ Stocks  
Mutual Funds

clients most likely to be interested.

The **Option** box shows what the end user sees when running the touchpoint. The **Value** box is what the person configuring the touchpoint sees, and what is passed along to the analytic workflow.

The **Value** must have the same format as the original workflow variable. In the example below, `@product_category_id` describes a number that corresponds to a product category in the data source. In the workflow view, the value is passed to a Row Filter operator as `product_category_id=@product_category_id`. We know looking at the data source that `@product_category_id` is a number, so our **Options** should also correspond to numbers. So when a user makes a choice "Bonds", the Value 25 is passed to the workflow.

In the touchpoint configuration:

Parameter 2

Workflow Variable \*  
@product\_category\_id

☐ Use Workflow's existing value as default  
☒ Required Field

Field Label \*  
Product Category

Description  
Select the product category for which you want to find the clients most likely to be interested.

Data Type \*  
Select Single Option

Enter each option the user can choose. If the option has an internal value that must be used, enter that with the option, and it will be passed to the workflow instead of the main option label.

#	Option *	Value
1	Bonds	25
2	Stocks	29
3	Mutual Funds	49

Add Option

Variable Details  
Default Value  
29

In the workflow where the variable is used:

Define Filter

AND OR NOT BETWEEN IN LIKE = < > <= >=

1 product\_category\_id=@product\_category\_id

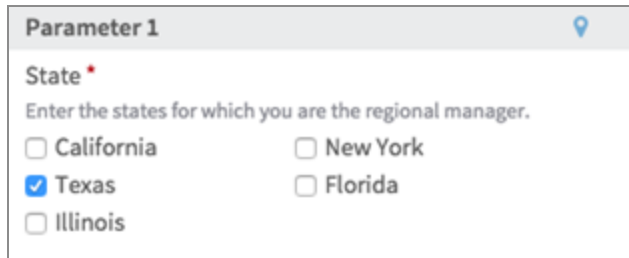
Available Columns  
customer\_id  
product\_category\_id  
sum(item\_quantity)

change to simple mode.

Cancel OK

## Multiple-Select Option Touchpoint Parameter

The multiple-select option input appears as a list of checkboxes that the user can select.



**Parameter 1**

**State \***

Enter the states for which you are the regional manager.

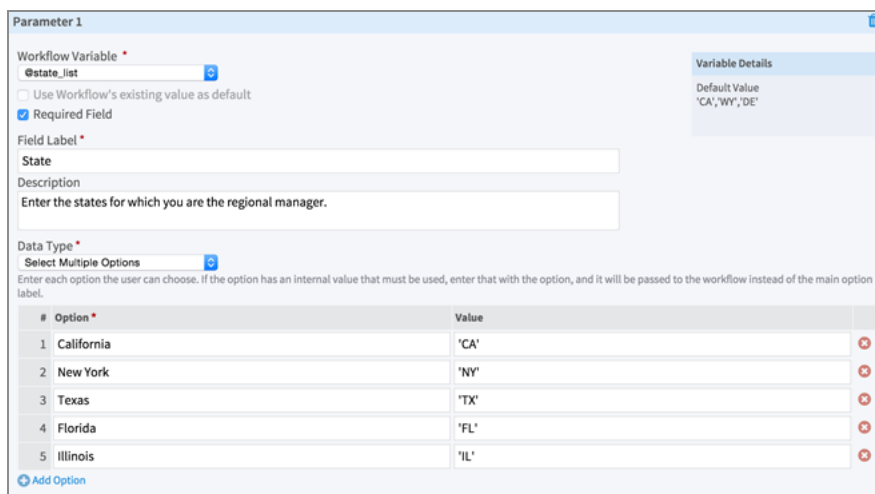
☐ California ☐ New York

☒ Texas ☐ Florida

☐ Illinois

Since the user can choose more than one option, the value(s) are usually queried as a list. See the **Variable Details** for more information about the default format. In the **Row Filter** query to the right, `@state_list` is enclosed in parentheses. This is because when running the query, `@state_list` becomes `'CA','WY','DE'`, for example, making the full query `state_code in ('CA','WY','DE')`. Without the parentheses, that query would not work. Parentheses are not added by the touchpoint to allow users to use their variables in the most flexible ways possible.

In the touchpoint configuration:



**Parameter 1**

Workflow Variable \*  
@state\_list

☐ Use Workflow's existing value as default  
☒ Required Field

Field Label \*  
State

Description  
Enter the states for which you are the regional manager.

Data Type \*  
Select Multiple Options

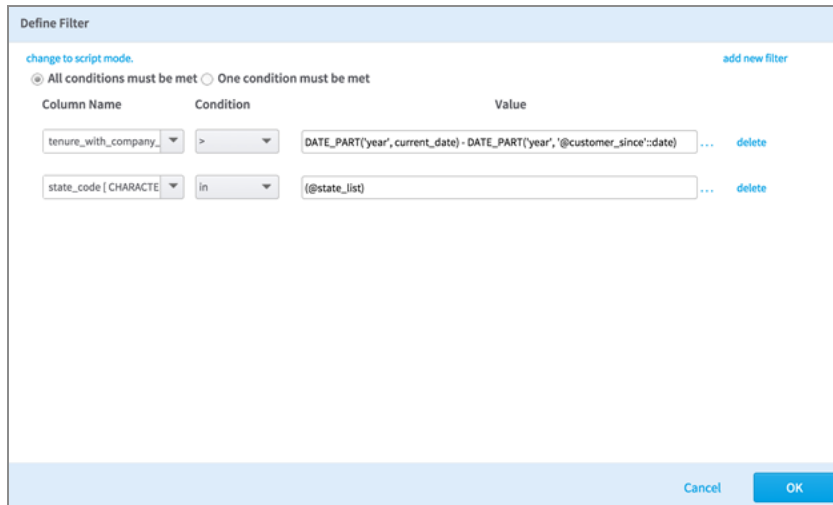
Enter each option the user can choose. If the option has an internal value that must be used, enter that with the option, and it will be passed to the workflow instead of the main option label.

#	Option *	Value
1	California	'CA'
2	New York	'NY'
3	Texas	'TX'
4	Florida	'FL'
5	Illinois	'IL'

Add Option

**Variable Details**  
Default Value  
'CA','WY','DE'

In the workflow where the variable is used:



Define Filter

[change to script mode.](#) [add new filter](#)

☒ All conditions must be met ☐ One condition must be met

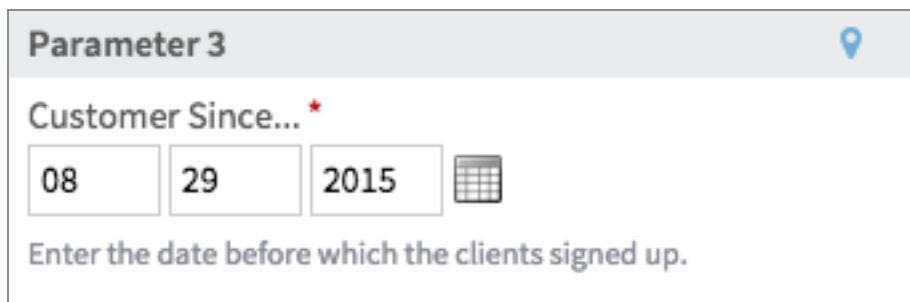
Column Name	Condition	Value	
tenure_with_company_	>	DATE_PART('year', current_date) - DATE_PART('year', '@customer_since::date)	... delete
state_code [ CHARACTER	in	(@state_list)	... delete

Cancel OK

## Date/Time Touchpoint Parameter


The date/time input appears as a date picker where the user can type in or select a date graphically.

**Note:** Only International Standard ISO-formatted dates are accepted as input via a workflow variable for the DateTime Calendar option. Any variables must be in the format YYYY-MM-DD. Other date formats must be changed to ISO format or passed as a **text** option in the touchpoint.

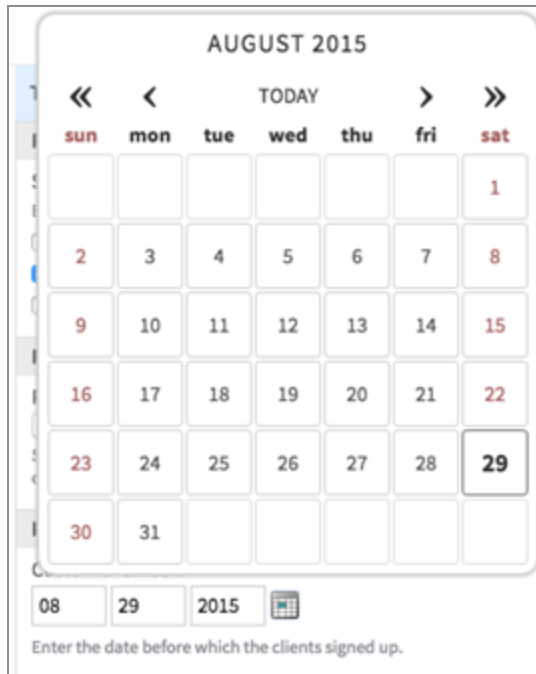


**Parameter 3**

Customer Since... \*

08 29 2015 

Enter the date before which the clients signed up.



AUGUST 2015

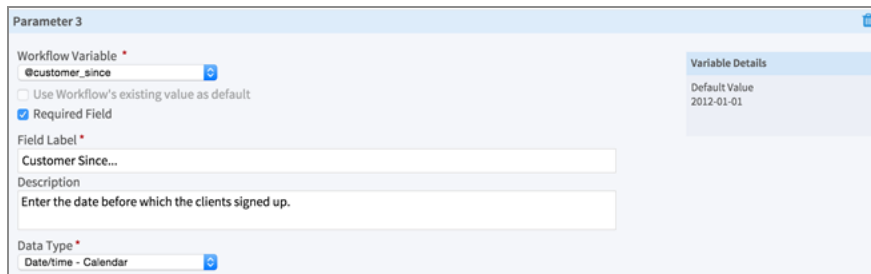
« < TODAY > »

sun	mon	tue	wed	thu	fri	sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

08 29 2015

Enter the date before which the clients signed up.

Again, consider how the variable is used in the workflow for formatting tips. Usually, dates must be surrounded with quotes in filters and queries.



Parameter 3

Workflow Variable \*

@customer\_since

☐ Use Workflow's existing value as default

☒ Required Field

Field Label \*

Customer Since...

Description

Enter the date before which the clients signed up.

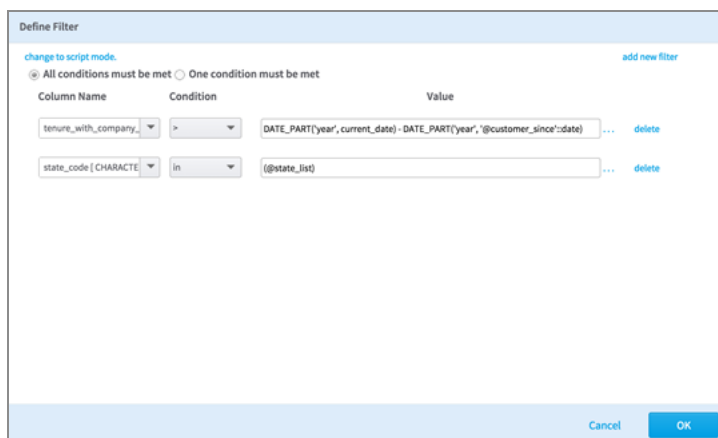
Data Type \*

Date/time - Calendar

Variable Details

Default Value

2012-01-01



Define Filter

change to script mode. add new filter

☒ All conditions must be met ☐ One condition must be met

Column Name	Condition	Value	
tenure_with_company	>	DATE_PART('year', current_date) - DATE_PART('year', '@customer_since::date)	delete
state_code   CHARACTER	in	@state_list	delete

Cancel OK

# Search

To search TIBCO Data Science - Team Studio objects by name, provide the name of a dataset, workspace, or user in the **Search** box. Accessing the search function from this field automatically defaults to **Search in All** and **Show All Results**.

- The Search page displays the results of your query.
- See the table of searchable TIBCO Data Science - Team Studio fields listed in this topic for details on what information is indexed and searchable.
- Data sources are indexed by default daily at midnight with the data source owner's credentials. A TIBCO Data Science - Team Studio administrator can configure the time and frequency, as described in *TIBCO® Data Science - Team Studio System Requirements*.
- Search also supports an auto-complete feature that matches names of objects.

When you submit a search string from any of the pages, the search page is displayed. Search does the following.

- Defaults to **Search in All of** and **Show All Results**.
- Displays all the recent activity associated with the search object.

Results are included in primary TIBCO Data Science - Team Studio objects such as datasets, users, and so on.


## Search Page Options

Use these options to narrow the scope of your search to your own files or a specific object group.

- **Search in:** From the dropdown menu, you can narrow the scope of your search to your own files by selecting **My Workspaces**. The default is **All of Team Studio**.
- **Show:** Use this dropdown menu to narrow your search to a specific object group. The search options are as follows.
  - **All Results** (the default)
  - **Work Files**



- **Other Files**
- **Hadoop Files**
- **Datasets**
- **Data Sources**
- **Workspaces**
- **People**
- **Engines**

 **Note:** Wildcards are supported.

The following table lists the searchable fields in TIBCO Data Science - Team Studio, and which objects are provided in the search results for each field.

Item	Indexed Fields
Account/Person	Name
	First Name
	Last Name
	Email Address
	Title
	Department
	Notes
Data Source	Name
	Description
	Host
	Port

Item	Indexed Fields
Workspace	Name
	Description
Sandbox	Database Name
	Schema Name
Work File	Name
	Description
Insights, Notes, Comments	Content
Dataset	Content
	Object Name
GPDB Table and View	Name
	Description
	Database Name
	Schema Name
GPDB Column	Name
	Description
	Database Name
	Schema Name
HDFS Files	Name
	Directory

# Tags

Tags are keywords or labels that can be assigned to items in TIBCO Data Science - Team Studio. Tags provide a mechanism for identifying and informally categorizing material in the application.

As examples of tag classification, workflows can be tagged by their usage or algorithm, datasets can be tagged by their users, users can be tagged by their expertise, and so forth.

You can add tags for all of the following.

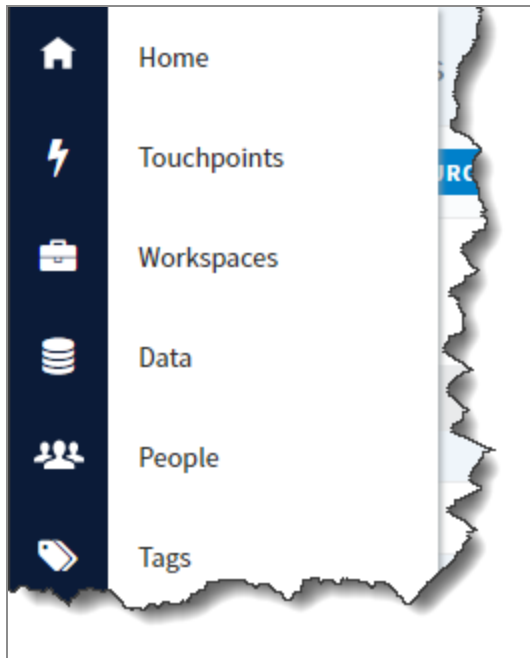
- Data sources
- People
- Workspaces
- Datasets
- Work files






Except for the Business User role, any account can create a new tag, name it, and label objects with it.

## Viewing Tags

You can have a view of primary classification tags from the primary navigation menu.

On the primary navigation menu, select **Tags**.



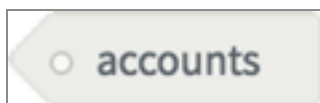
133 Tags		Page 1 of 3	Previous	Next
Name	Number of Items			
 ABS facet	1			
 accounts	1			
 Alpiners	48			
 asda	0			
 awesome	1			



**Note:** Tags do not have a description field associated with them, so it is important to give the tag a name that is self-describing.

## Navigating with Tags

When you view a list of tags, you see the number of items that are tagged with that tag.



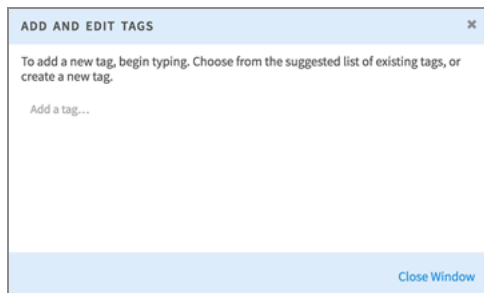
To view a list of all the items to which you have access that are tagged with a particular tag, click the name of that tag on the tags page or enter its name in the search field. When you click the name of a tag in any list, a search is initiated for other items with that tag. A list of tagged items to which you have access appears.

# Adding and Editing Tags

You can add or create a tag only when you have an object selected that can be tagged.

## Procedure

1. To add a tag, click **Add/Edit Tags** in the contextual sidebar. This opens a dialog.



2. Enter a few characters of the name you have chosen.

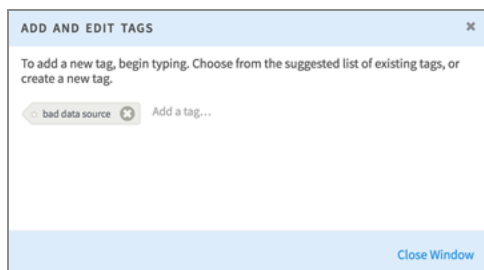
A list of all existing tags that include that letter sequence opens.



3. Choose the one you want by clicking it. Alternatively, to create a new tag, type its name and press Enter.

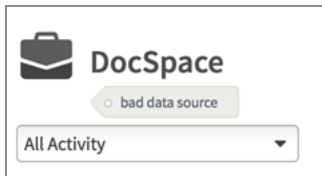
In the above case, no existing tags matched your input.

4. Press Enter to create the new tag. The next time you select **Add/Edit Tags**, this dialog appears:



**Note:** Tags listed in blue are all the tags attached to the highlighted object. In this example, there is only one tag.

The tag appears next to your tagged object. Here is an example of a tagged workspace:



Tagged workspaces can also be seen in the list of workspaces:

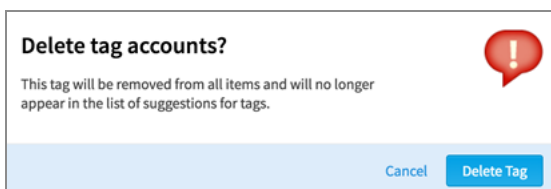


## Deleting a Tag

Only an administrator can delete tags.

### Procedure

1. From the primary sidebar, choose **Tag** to see a list of existing classification tags.
2. Select a desired tag, then choose **Delete Tag**.

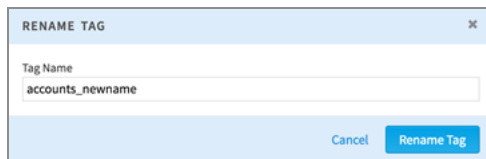


## Renaming a Tag

Use this procedure to rename a tag.

### Procedure

1. From the primary sidebar, select **Tags** to see a list of existing classification tags.
2. Select a desired tag, then click **Rename Tag**.

A dialog box titled "RENAME TAG" with a close button (X) in the top right corner. It contains a text input field labeled "Tag Name" with the text "accounts\_newname" entered. At the bottom, there are two buttons: "Cancel" and "Rename Tag".

## Jupyter Notebooks

You can integrate Jupyter Notebooks running Python code with TIBCO Data Science - Team Studio.

For more information about Jupyter Notebooks, see [jupyter.org](https://jupyter.org).

If your organization has installed Jupyter Notebooks, Python, and required Python packages on a server, you can create Jupyter Notebooks and integrate them into your workflow.

## Creating a Jupyter Notebook

You can create a notebook from the TIBCO Data Science - Team Studio **Work Files** tab.

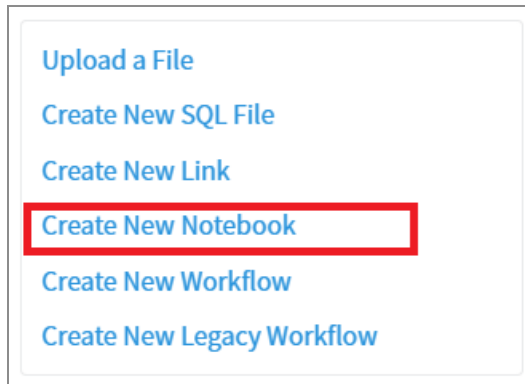
### Before you begin

You must have access to a TIBCO Data Science - Team Studio server that is configured to work with Jupyter Notebooks for Team Studio .

### Procedure

1. From the TIBCO Data Science - Team Studio menu, click **Workspaces**.
2. Click the name of the workspace in which to create a Jupyter Notebook.
3. Click **Work Files** tab.

The Workfile action menu displays the options for work files.



4. Click **Create New Notebook**.
5. Provide a file name for the notebook, and optionally provide a description.

The file extension for Jupyter Notebooks is `.ipynb`. The name you provide, with that extension, is displayed in the **Work Files** list after you create the Jupyter Notebook.

**i Note:** Valid names for notebooks include numbers, letters, spaces, and any of the following characters.

(

)

.

-

\_

Work file names containing other characters cannot be created.

Initially, the work file space displays the message Preview Unavailable.

6. Click **Open Notebook** to begin defining the new Jupyter Notebook.

## Installing Python Packages

Follow this step to install packages from within the notebook environment. Start in the Jupyter Notebooks environment in the TIBCO Data Science - Team Studio workspace.



## Before you begin

- You must have access to a server running Python and Jupyter Notebooks for Team Studio.
- You must have created a Jupyter Notebook.

To install the packages, enter `!pip3 install packagename` at the command prompt.

The following example demonstrates installing the publicly-available package PyPI.

```
In [2]: ! pip3 install PyPI
Collecting PyPI
  Downloading https://files.pythonhosted.org/packages/32/d2/8fb6f6a37af408d52dae28b4ff999e0d11b936729661c93d242ffbdb0a77/pypi-2.1.tar.gz
Building wheels for collected packages: PyPI
  Running setup.py bdist_wheel for PyPI ... done
  Stored in directory: /data/.cache/pip/wheels/0c/96/86/b26df386ec06e1fc42a4dab6b33dcfbc85f1cc3002278ff525
Successfully built PyPI
Installing collected packages: PyPI
Successfully installed PyPI-2.1
```

## Python Packages Required for Jupyter Notebooks in TIBCO Data Science - Team Studio

TIBCO Data Science - Team Studio requires specific Python packages for data analysis. Download these package versions to run Jupyter Notebooks for Team Studio. (Dependent packages download by default.)

Package name	Version number
jupyterhub	1.5.0
notebook	6.4.8
numpy	1.22.2
pandas	1.3.5
scikit-learn	1.0.2
scipy	1.8.0

Package name	Version number
seaborn	0.11.2
tensorflow	2.7.0

## Initializing PySpark for Spark Cluster

If you have configured a Spark compute cluster in the Chorus **Data** section, you can use PySpark to read and write table data. This notebook can be exposed as a [Python Execute](#) operator which can be used by the downstream operators in a workflow.

This method is far more efficient than other methods for medium or large data sets and is the only viable option for reading data sets having a size of more than a few GB.

**i Note:** Only one data set is initialized for one data source.

You can initialize and use PySpark in your Jupyter Notebooks for Team Studio.

To perform this, start in the Notebooks environment in TIBCO Data Science - Team Studio. This option reads data that is available for Apache Spark 3.2 or later cluster.

**i Note:** If an error appear while trying to write data on the shared volume, then add the following line to the `PYSPARK_SUBMIT_ARGS` environment variable:

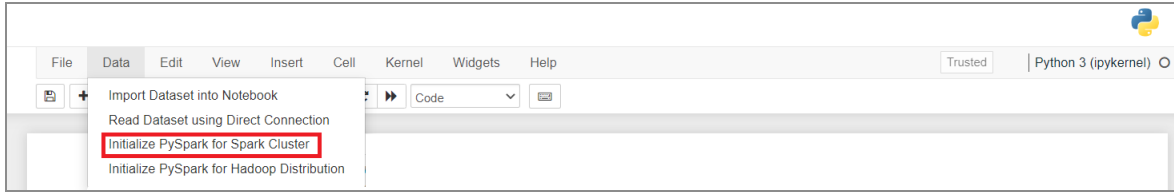
```
--conf spark.hadoop.fs.permissions.umask-mode=000
```

If you are using an EMR Cluster as Spark Cluster, then add the following line to the `PYSPARK_SUBMIT_ARGS` environment variable:

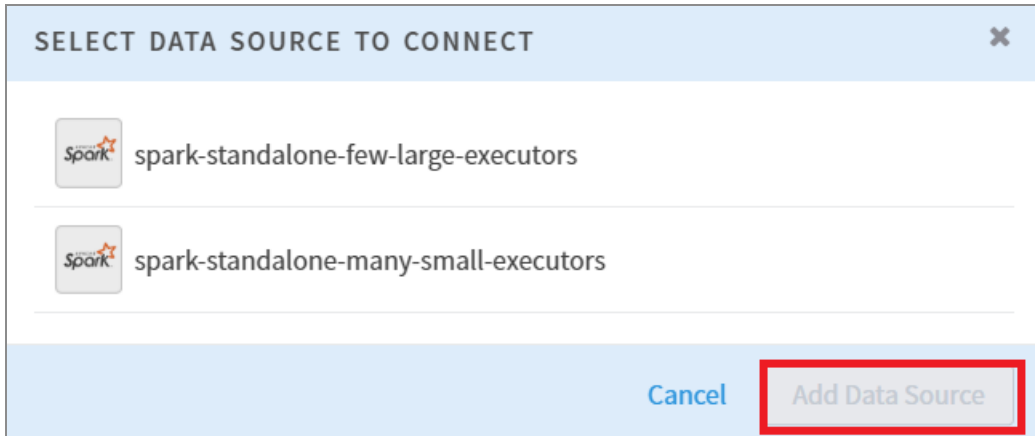
```
--packages org.apache.hadoop:hadoop-aws:3.3.1
```

### Procedure

1. [Create a new notebook.](#)
2. Click **Data**, and then select **Initialize PySpark for Spark Cluster**.



3. A SELECT DATA SOURCE TO CONNECT dialog appears. Select an existing Spark Cluster data source, and then click **Add Data Source**.



4. A bit of code is inserted into your notebook. This facilitates the communication between the data source and your notebook. If you want to read more data source, repeat step 1 to step 3 (maximum limit is 3 for inputs to a python execute operator). To run this code, press shift+enter or click **Run**.

Now, you can run other commands by referring to the comments in the inserted code.

The commands use the object `cc` which is an instantiated object of class `ChorusCommander` with the required parameters for the methods to work correctly. The generated code sets the `sqlContext` argument to the initialized Spark Session in the `cc.read_input_table` method call. You can set `spark_options` dictionary argument to pass additional options having driver property set to the auto-detected JDBC Driver class.

5. To read the data sets, uncomment the lines in the generated code for that data set along with the `_props` variable having the corresponding `spark_options`.
6. To use the notebook as a python execute operator, change the `use_input_substitution` parameter from `False` to `True` and add the `execution_label` parameter for data sets to be read. The `execution_label` value should start from

string '1', followed by '2', and '3' for subsequent data sets. For more information, see `help(cc.read_input_table)`.

7. The generated `cc.read_input_table` method call returns a Spark Data Frame. You can modify, copy, or perform any other operations on the Data Frame as required.
8. Once the required output Spark Data Frame has been created, write it to a target table using the `cc.write_output_table`.
  - To enable the use of output in downstream operators, set `use_output_substitution=True`.
  - To drop any existing table having the same name as the target, set the `drop_if_exists` parameter to `True`.
  - Use the `spark_options` argument as in the `read_input_table` for the driver. You can also write to a different target data source by updating `cc.datasource_name = '...'` to the required value and the driver in `spark_options` accordingly.

For more information, see `help(cc.write_output_table)`.

9. Run the notebook manually for the metadata to be determined so that the notebook can be used as a [Python Executor](#) operator in legacy workflows.

## Initializing PySpark for Hadoop Distribution

If you have configured a Hadoop Cluster (EMR/CDH) in the Chorus **Data** section, you can use PySpark to read and write HDFS data. This notebook can be exposed as a [Python Execute](#) operator which can be used by the downstream operators in a workflow.

This method is far more efficient than other methods for medium or large data sets and is the only viable option for reading data sets having a size of more than a few GB.

You can initialize and use PySpark in your Jupyter Notebooks for Team Studio. Start in the Notebooks environment in TIBCO Data Science - Team Studio.

### Before you begin

This prerequisite update applies only if you created a notebook before version 6.5.0 of TIBCO Data Science - Team Studio using the `Initialize Pyspark for Cluster` function. This is required to accommodate Spark upgrades in the system.

1. Regenerate the PySpark context by clicking **Data > Initialize Pyspark for Cluster**.
2. Change the previously-generated code to the following:

```
os.environ['PYSPARK_SUBMIT_ARGS']=
"--master yarn-client --num-executors 1 --executor-memory 1g --
packages com.databricks:spark-csv_2.10:1.5.0,com.databricks:spark-
avro_2.11:3.0.1
pyspark-shell"
```

If you do not have access to a Hadoop cluster, you can run your PySpark job in local mode. Before running PySpark in local mode, set the following configuration.

1. Set the PYSPARK\_SUBMIT\_ARGS environment variable as follows:

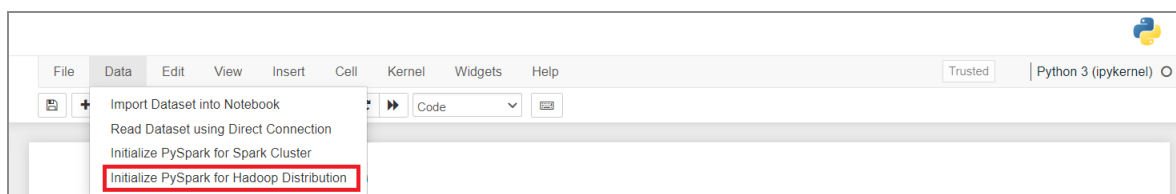
```
os.environ['PYSPARK_SUBMIT_ARGS']= '--master local pyspark-shell'
```

2. YARN\_CONF\_DIR environment variable as follows:

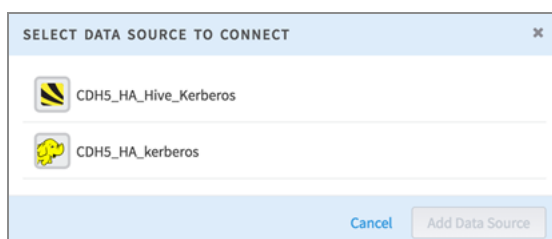
```
os.environ['YARN_CONF_DIR'] = ''
```

## Procedure

1. [Create a new notebook](#).
2. Click **Data**, and then select **Initialize PySpark for Hadoop Distribution**.



3. A SELECT DATA SOURCE TO CONNECT dialog appears. Select an existing Hadoop data source, and then click **Add Data Source**.



4. A bit of code is inserted into your notebook. This facilitates the communication

between the data source and your notebook. If you want to read more data source, repeat step 1 to step 3 (maximum limit is 3 for inputs to a python execute operator). To run this code, press shift+enter or click **Run**.

Now, you can run other commands by referring to the comments in the inserted code.

The commands use the object `cc` which is an instantiated object of class `ChorusCommander` with the required parameters for the methods to work correctly. The generated code sets the `sqlContext` argument to the initialized Spark Session in the `cc.read_input_file` method call. You can set `spark_options` dictionary argument to pass additional options to Spark Data Frame Reader for the CSV format.

5. To read the data sets, uncomment the lines in the generated code for that data set along with the `_props` variable having the corresponding `spark_options`.
6. To use the notebook as a python execute operator, change the `use_input_substitution` parameter from `False` to `True` and add the `execution_label` parameter for data sets to be read. The `execution_label` value should start from string '1', followed by '2', and '3' for subsequent data sets. For more information, see `help(cc.read_input_file)`.
7. The generated `cc.read_input_file` method call returns a Spark Data Frame. You can modify, copy, or perform any other operations on the Data Frame as required.
8. Once the required output Spark Data Frame has been created, write it to a target table using the `cc.write_output_file`.
  - To enable the use of output in downstream operators, set `use_output_substitution=True`.
  - To overwrite any existing files in the same path as the target, set the `overwrite_exists` parameter to `True`.
  - You can set `spark_options` dictionary argument to pass additional options to Spark Data Frame Writer for the CSV format.

For more information, see `help(cc.write_output_file)`.

**Note:**

- If this is not a terminal operator in the workflow, then do not set `header=True` because subsequent operators use operator metadata and not the header line in the output file.
- You can use `comma(,)` as a delimiter argument in `write_output_file` for compatibility with other operators in a workflow, or do not use a delimiter argument.

9. Run the notebook manually for the metadata to be determined so that the notebook can be used as a [Python Executor](#) operator in legacy workflows.

## Sample commands

Use the following sample commands to write a table:

```
cc.write_output_table(ds1_adult_csv, table_name='adult_outemr2_
hadoop.csv', schema_name='Compute_s3_local',
database_name='', sqlContext=spark, spark_options=ds1_props,
overwrite_exists=True, drop_if_exists=True, use_output_
substitution=True)
```

Use the following sample commands to write a file:

```
cc.datasource_name = 'EMR535_Large3'
cc.write_output_file(ds1_adult_csv, file_path='/tmp/adult_out.csv',
sqlContext=spark, file_type='csv', spark_options={}, overwrite_
exists=True)
```

## Creating a Custom Environment for Running Jupyter Notebooks

You can create custom environments that give you more control over the environment.

**! Important:** A Jupyter Notebook used by the Python Execute operator always runs under the owner of the workspace it is contained in. This means that only environments known by that account are available to the Python Execute operator.

## Before you begin

You must have access to a server running Python and Jupyter Notebooks for Team Studio.

Create the environment from within a Jupyter Notebook.

Add the following code snippet to a cell, and then run the cell.

```
!conda create --clone python3 -n MyTest
!conda install -n MyTest -y docker-py
!conda run -n MyTest python -m ipykernel install --user --name MyTest
```

## Result

A clone of the base Python 3 environment is created.

## What to do next

After you create the environment and close the current Jupyter Notebook instance, the new environment is available to the current user.

For more information about managing custom environments, see [conda documentation](#).

For an example that provides verbose output and a list of packages installed, see [Uploading and Running the Conda Environment Example](#).

# Uploading and Running the Conda Environment Example

The example file `Conda.ipynb` is included on the TIBCO documentation site for your information. You can download it and run it in TIBCO Data Science - Team Studio.

This example creates a custom environment similar to the simple example presented in [Creating a Custom Environment for Running Jupyter Notebooks](#); however, it is more verbose and lists the packages installed.



## Before you begin

You must have Jupyter Notebooks installed.

## Procedure

1. From the TIBCO® Data Science Team Studio [docsite](#), under the **Product Guides** list, click **Conda Notebook Sample** to download a sample Notebook file, or click [here](#) to download the sample Notebook file.

The file `Conda.ipnyb` is downloaded to your computer.

2. Open TIBCO Data Science - Team Studio, and then open your workspace.
3. Click **Work Files** tab.
4. From the right-hand menu, click **Upload a File**.

The Upload File dialog is displayed.

5. Click **Select a File**, and then browse to the location where you downloaded `Conda.ipnyb`
6. Upload the file.

The file name is displayed in the Work Files list.

7. Click the file title to display it, and then click **Open Notebook**. A bit of code appears in your notebook.
8. From the menu, click **Cell > Run All**

The example runs, displaying information and a list of the packages installed.

## Result

The custom environment is created.

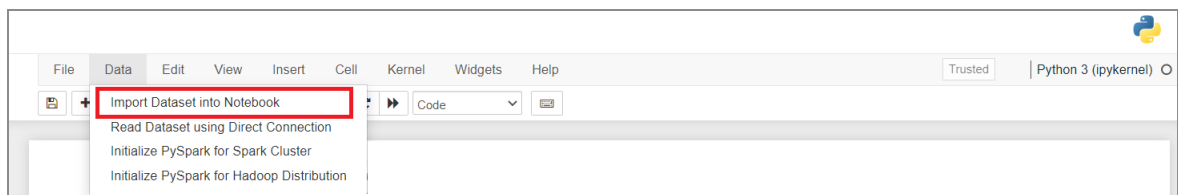
# Adding Your Data to a Notebook

The notebook reads a table, perform various operations, and then write the output to the table.

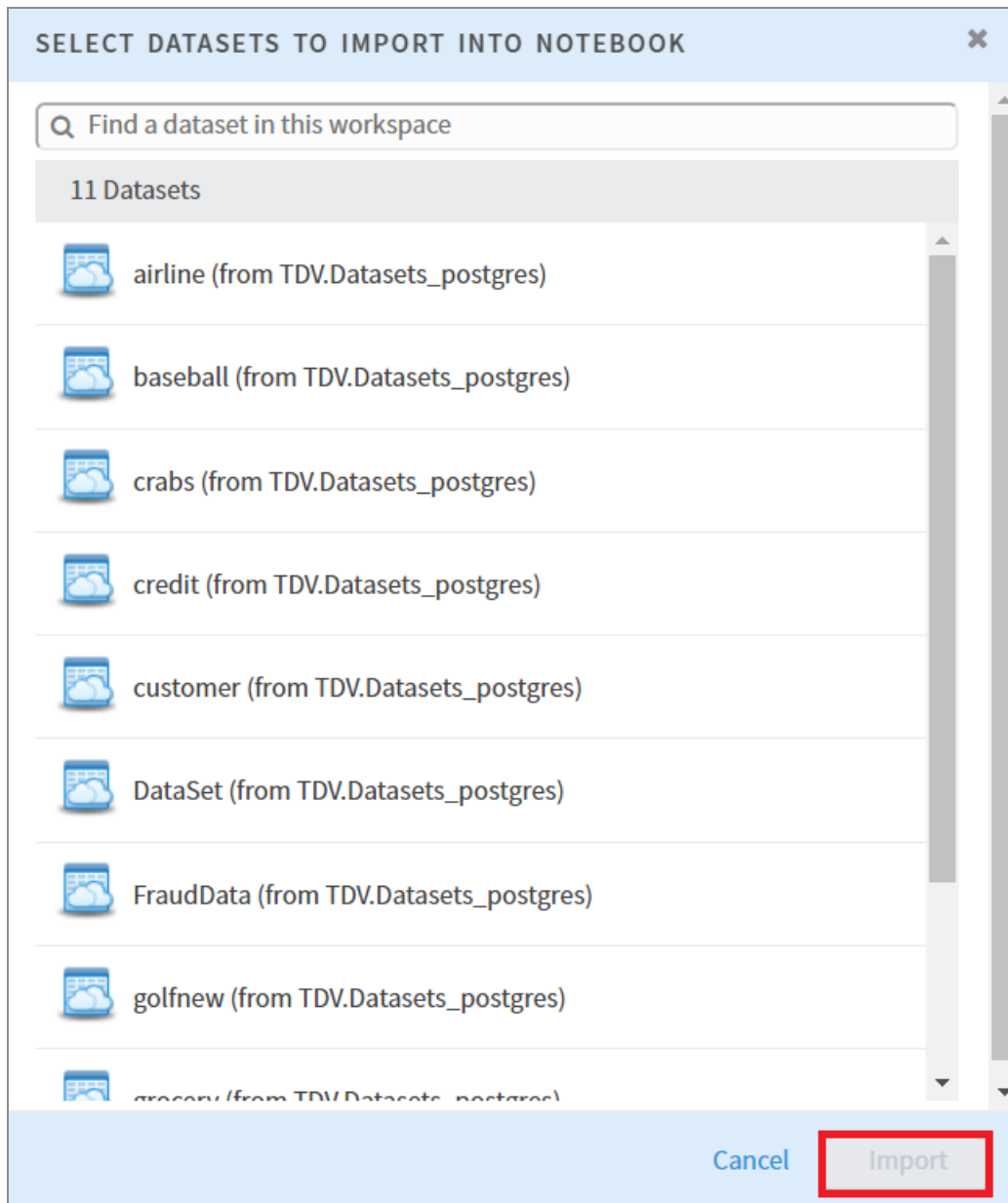
Perform the following steps to add your data to a notebook for processing.

## Procedure

1. Associate the dataset with your workspace.
2. From the menu, select **Data**, and then select **Import Dataset Into Notebook**.



3. A SELECT DATASETS TO IMPORT INTO NOTEBOOK dialog appears. Select the dataset to import, and then click **Import**.



4. A bit of code is inserted into your notebook. This facilitates the communication between the data source and your notebook. To run this code, press shift+enter or click **Run**.

```
In [ ]: # NOTE: Use this only for small datasets having less than say 500K rows.
# For larger datasets, use direct connection or one of the PySpark options.
cc.datasources_name = 'TDV'
# NOTE: names that do not fall under SQL unquoted identifiers are quoted internally by double quotes or backticks.
# Backticks are added for MySQL, MariaDB, Hive, Hive2 and BigQuery while double quotes are used for the rest.
# If your database is not among these or uses some other quote character, pass 'quotechar' argument below.
df_credit = cc.read_input_table(table_name='credit', schema_name='', database_name='Datasets_postgres',
                               use_input_substitution=False)
```

5. If you run `df_credit` (df\_ the name of the dataset imported into your notebook) in the Python notebook, it shows a preview of the data imported.

In [7]: `df_credit`

Out[7]:

	id	times90dayslate	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslate_2years	srsdlqncy
0	2	0	0.26	0.324	4	1956.592205	0	0
1	6	0	0.25	0.927	6	3983.456356	0	0
2	52	0	0.46	0.372	7	3595.370678	0	0
3	76	2	0.88	0.154	3	1729.964479	2	0
4	84	0	0.14	0.690	7	2507.165171	0	0
...	...	...	...	...	...	...	...	...
49995	403983	0	0.24	0.427	4	1744.117114	0	0
49996	403999	1	0.61	0.307	5	3308.104026	0	0
49997	404019	0	0.15	0.304	7	4465.674432	0	0
49998	404021	0	0.27	0.345	6	3198.668753	0	0
49999	404047	0	0.29	0.356	4	3156.833186	0	0

50000 rows × 8 columns

Now, you can run other commands by referring to the comments in the inserted code. The `write_output_table` is added manually on the same pattern as the generated code for writing the output to a table.

## Reading Your Data Using Direct Connection in a Notebook

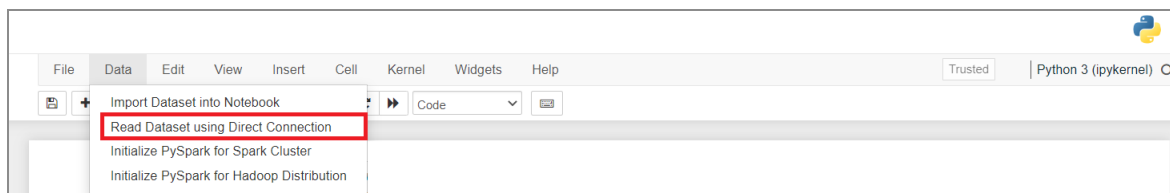
This option is used to read a table, perform various operations, and then write the output to the table. It is more efficient than [Import Dataset into Notebook](#) option because the data are extracted directly from the source by Python. This can only be used with small data sets because the entire data is loaded in memory.

You can use this option to read or write files and expose those notebooks as a python execute operator that downstream operators can use in a workflow. This makes the data read much more efficiently than the default mode which uses Chorus REST APIs. There is no limit to the data set size that can be read using this mode. This only depends on the available memory in the notebook process or container.

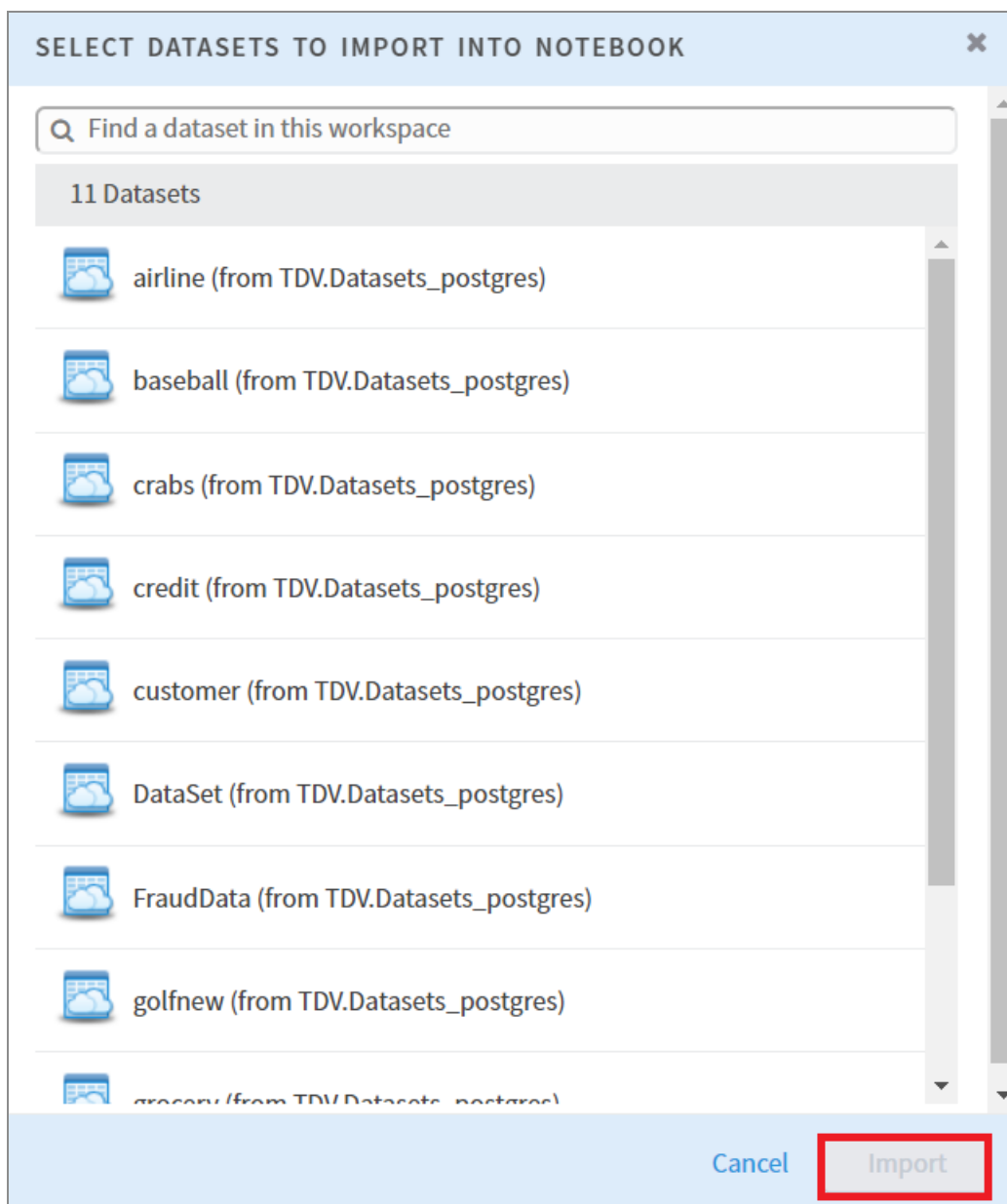
Perform the following steps to read your data in a notebook by using the direct connection.

### Procedure

1. Associate the data set with your workspace.
2. From the menu, select **Data**, and then select **Read Dataset using Direct Connection**.



3. A SELECT DATASETS TO IMPORT INTO NOTEBOOK dialog appears. Select the data set to import, and then click **Import**.



4. A bit of code is inserted into your notebook. This facilitates the communication between the data source and your notebook. If you want to read more data sets, repeat step 1 to step 3 (maximum limit is 3 for inputs to a python execute operator). To run this code, press **shift+enter** or click **Run**. The commands use the object `cc` which is an instantiated object of class `ChorusCommander` with the required parameters for the methods to work correctly.

```
In [12]: # NOTE: Use this only for small datasets having less than 1-2M rows else Pandas is slow and resource heavy.
# For larger datasets, use one of the PySpark options (with to_pandas_on_spark() to get a Pandas DataFrame).
# Or you can skip loading the entire table into the DataFrame rather process piece-wise using custom code like:
# from ChorusCommander.Log import stderr_redirected
# ds79_props = cc.get_tds_jdbc_properties(79)
# with cc.jdbc_connect(ds79_driver, ds79_url, ds79_props, fetchsize=1000) as ds79_conn, stderr_redirected(os.devnull):
#     ds79_purchase = cc.read_sql(ds79_conn, 'select * from ...')
ds79_driver = 'cs.jdbc.driver.CompositeDriver'
ds79_url = 'jdbc:compositesw:dbapi@10.69.30.31:19401?dataSource=Data&domain=composite&traceLevel=all&sharedVolumeMountPoint=/mnt/tdv'
cc.datasource_name = 'TDV'
# NOTE: names that do not fall under SQL unquoted identifiers are quoted internally by double quotes or backticks.
# Backticks are added for MySQL, MariaDB, Hive, Hive2 and BigQuery while double quotes are used for the rest.
# If your database is not among these or uses some other quote character, pass 'quotechar' argument below.
# Reduce fetchsize parameter if your row is large (e.g. multi-MB blobs) or increase to 10K if its small enough.
ds79_purchase = cc.read_input_table(table_name='purchase', schema_name='Datasets_postgres', database_name='',
                                   jdbc_driver=ds79_driver, jdbc_url=ds79_url, fetchsize=1000,
                                   use_input_substitution=False)
```

```
INFO 2022-09-05 10:13:28.770 cs.jdbc.driver.DriverOptions@3943a2be
url jdbc:compositesw:dbapi@10.69.30.31:19401?dataSource=Data&domain=composite&traceLevel=all&sharedVolumeMountPoint=/mnt/tdv/qat1&allowHttpInsecure=true&unsupportedMode=false&enableFlood=false
host 10.69.30.31
port 19401
user admin
password *****
domain composite
catalog null
dataSource Data
traceLevel all
traceFile null
traceFolder null
tracePerf null
testMode null
fetchRows -1
fetchBytes -1
requestTimeout -1
connectTimeout -1
```

5. If you run `ds79_purchase` (`ds79` the name of the data set imported into your notebook) in the Python notebook, it shows a preview of the data imported.

In [13]: ds79\_purchase

Out[13]:

	tx_no	item
0	1	Cake
1	1	Apple
2	2	Wine
3	1	Beef
4	3	Beer
...	...	...
87	39	Cake
88	40	Pork
89	40	Wine
90	41	Coke
91	41	Beer

92 rows x 2 columns

Now, you can run other commands by referring to the comments in the inserted code.

6. The generated code detects and sets the appropriate `jdbc_driver` argument in the `cc.read_input_table` method call which enables the direct connection mode. For some data sources that are unknown to Chorus, the JDBC Driver class name which is set in `jdbc_driver` may not be detected correctly, so change them if it is not as expected by the data source. You can also pass the `jdbc_url` argument to explicitly override the JDBC URL.
7. To use the notebook as a python execute operator, change the `use_input_substitution` parameter from `False` to `True` and add the `execution_label` parameter for data sets to be read. The `execution_label` value should start from string '1', followed by '2', and '3' for subsequent data sets. For more information, see `help(cc.read_input_table)`.
8. The generated `cc.read_input_table` method call returns a Pandas Data Frame. You can modify, copy, or perform any other operations on the Data Frame as required.
9. Once the required output Pandas Data Frame has been created, write it to a target table using the `cc.write_output_table`. To enable the use of output in downstream operators, set `use_output_substitution=True`. Use the same argument as in `read_input_table` to use direct connection mode for the write. Add the `drop_if_exists`

parameter to `True` to drop any existing table having the same name as the target. For more information, see `help(cc.write_output_table)`.

**Note:** You can use `comma(,)` as a delimiter argument in `write_output_file` for compatibility with other operators in a workflow, or do not use a delimiter argument.

10. Run the notebook manually for the metadata to be determined so that the notebook can be used as a [Python Executor](#) operator in legacy workflows.

## Incorporating Notebooks in a Workflow

Notebooks can be integrated in the visual workflow editor using the Python Execute operator.

Use this operator to run a Jupyter notebook that is stored in your current workspace using the inputs and outputs defined in your analysis. You can run a Jupyter notebook from any step within your workflow. For more information, see the operator topics.

- [Python Execute \(HD\)](#)
- [Python Execute \(DB\)](#)



# Workflow Operators

---

Operators are the building blocks of TIBCO Data Science - Team Studio workflows. Each one represents a unit of computation or a data source. Operators can perform tasks such as data loading and transformation or complete more advanced tasks such as training and evaluating machine learning models.

The following topics describe the most common operator concepts. The Workflow Operator Reference provides details for setting properties for each operator.

## Operator Actions

You can manipulate the operators and the connections between them in the workflow editor. Also, each operator has properties that you can set and edit in its dialog.

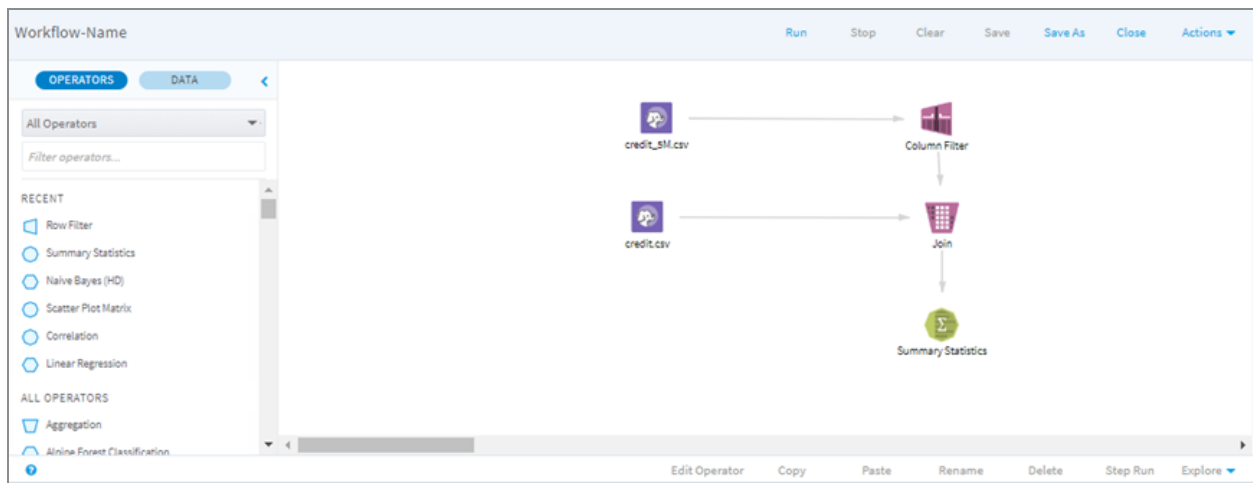
For information about using the operators in the TIBCO Data Science - Team Studio workflow editor, see [Explore Visual Results](#).

## Connecting Operators

A workflow is a collection of operators and data. You can create a workflow within this editor by dragging data and operators onto the canvas. You can move or manipulate elements with the mouse.

Start in the Workflow canvas.

Data operators and operations are connected with arrows.



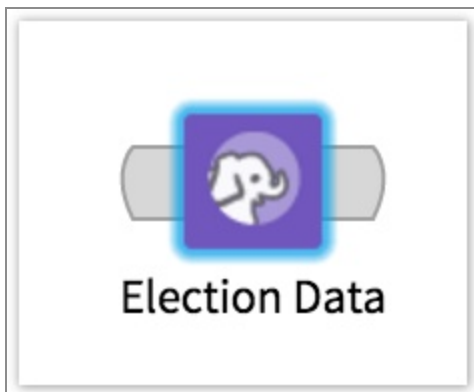
## Before you begin

- You must have a data source for the workflow.
- You must have set the properties for the operator.
- You must have a target operator to connect to.

## Procedure

1. Hover the mouse pointer over the data source.

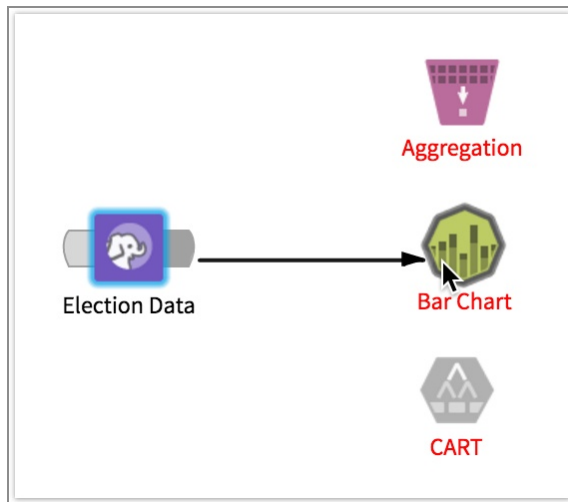
Two tabs are displayed on either side of the data icon.



2. Click and hold the mouse button over a tab, and then drag the mouse pointer to create an arrow.

An arrow is displayed on the canvas.

3. When the pointer is over the target operator, release the mouse button.



**Note:** Operators that you can connect to remain colored, while operators that you cannot connect to directly appear grayed out.

## Result

The two operators are now connected.

# Selecting Multiple Operators

You can select more than one operator at a time.  
Start in the workflow editor.

## Before you begin

You must have more than one operator in the workflow editor.

## Procedure

1. Click and drag a "lasso" around the desired operators.

The selected operators are highlighted.



**i Note:** You can also select multiple operators using the Control key (on Windows or Linux) or the Command key (on Mac OS X).

2. On your keyboard, press **Delete**.

## Editing Operator Properties

You can find and edit an operator's properties using one of the three available methods. Start in the workflow editor.

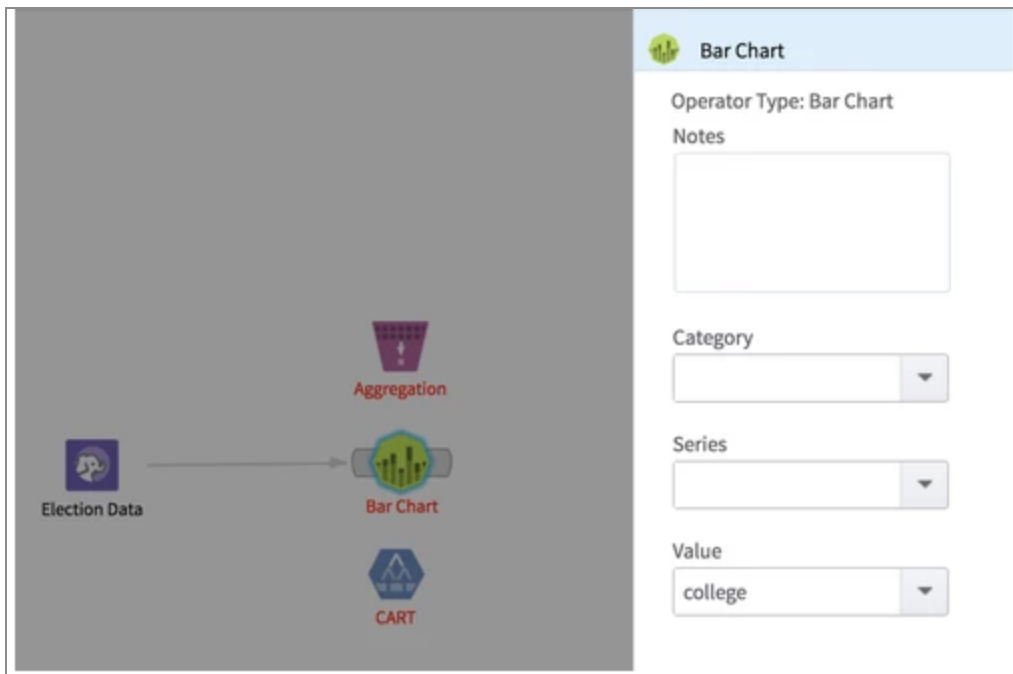
### Before you begin

You must have placed an operator in the workflow editor.

### Procedure

1. Open the property dialog for the operator using one of the following methods.
  - Select the operator, and then, on the toolbar, click **Edit Operator**.
  - Double-click the operator.
  - Right-click the operator, and from the context menu, click **Edit operator properties**.

The operator dialog is displayed.



2. Edit the properties as needed.  
Required properties are highlighted.  
See the individual operator topics for property details.

**i Note:** When you enter content in the **Notes** field, a yellow asterisk is displayed on the operator.

## Deleting Operators

When you delete an operator, all connections to or from the selected operator are removed.

Start in the workflow editor.

### Before you begin

You must have an operator in the workflow editor.



### Procedure

1. Select the operator to delete.
2. On the operator toolbar, click **Delete**.

Alternatively, you can right-click the operator, and then click **Delete**, or press the Delete key on the keyboard.

## Result

The operator is deleted.

For more information, see [Operator Actions](#).

# Moving Connections

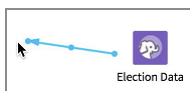
You can move a connection from one operator to another. As with creating a connection, the operator's color indicates valid and invalid connections. Start in the workflow editor.

## Before you begin

You must have connected operators.

## Procedure

1. Select the connection arrow.
2. Click and drag the dot located at either the head or tail of the arrow.



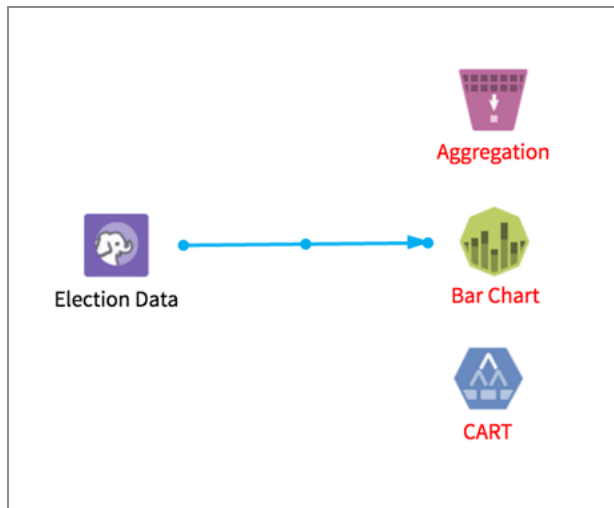
3. Drop the arrow on a new valid target operator.

The connection is changed to the new target operator.

# Deleting Connections

In a workflow - a collection of operators and data - you sometimes need to delete connections between operators.

Start in the Workflow canvas. Operators and data are connected with arrows.



### Before you begin

You must have a data source for the workflow, and you must have multiple connected operators.

To delete a connection arrow, first select the arrow. You can press **Delete** on the keyboard, use **Delete** on the toolbar, or double-click the dot in the middle of the arrow to remove it.

### Result

The connection is now removed. For more information, see [Workflow Operators](#).

## Data Management

With Team Studio data operators, you can manage multiple data sets or files. You can also move data between databases or from a Hadoop data source to a database, or from a database to Hadoop data source. Team Studio data operators provide the means to manage your data.

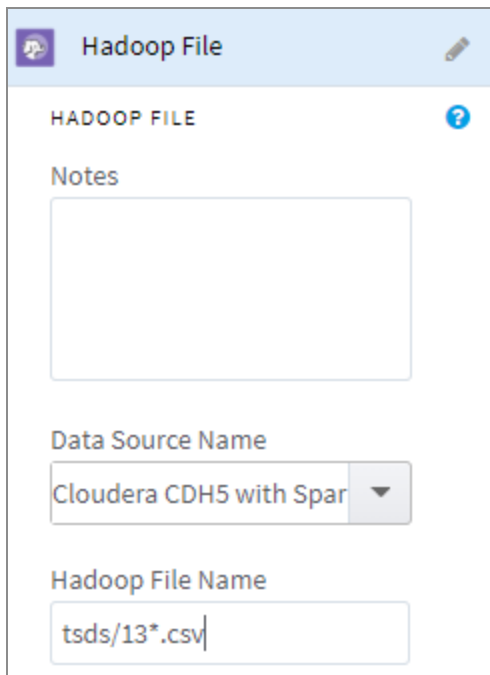
## Selecting Groups of HDFS files

TIBCO Data Science - Team Studio Hadoop users can select multiple, similarly-structured files from the Hadoop File operator.

To use this feature and include all needed files, you can add a Hadoop data source to your workflow, and then modify the **Hadoop File Name** text manually with wildcard characters.

The Hadoop file structure configuration is based on the first file, alphabetically, from this list, and is applied to all of the other files. Succeeding operators treat these files as a single, unified input.

For example, to select all of the .csv files that begin with "13" from a /user/tsds/ HDFS directory, set **Hadoop File Name** to /user/tsds/13\*.csv.

The screenshot shows the 'Hadoop File' operator configuration window. At the top, there's a title bar with a TIBCO icon and the text 'Hadoop File'. Below the title bar, the main area is labeled 'HADOOP FILE' with a help icon. Underneath, there's a 'Notes' section with a large empty text box. Below the notes, there's a 'Data Source Name' dropdown menu currently showing 'Cloudera CDH5 with Spar'. At the bottom, there's a 'Hadoop File Name' text input field containing the text '/tsds/13\*.csv'.

You can also use wildcards to specify multiple directories; for example, /user/\*/13\*.csv.

To be able to specify multiple directories, you must meet the following requirements.

- The TIBCO Data Science - Team Studio OS user has read permission for each file in the /user/\*/13\* hadoop file path.
- Each data node must be sending heartbeats to the network.
- Each data node is accessible from the TIBCO Data Science - Team Studio server.
- The selected files have the same Hadoop file structure.



## Wildcard patterns

A file name pattern can be composed of regular characters and any of the following wildcard characters.

Character or pattern	Matches
?	Matches any single character.
*	Matches zero or more characters.
[abc]	Matches a single character from character set {a,b,c}.
[a-b]	Matches a single character from the character range {a...b}. Note that character a must be lexicographically less than or equal to character b.
[^a]	Matches a single character that is not from character set or range {a}. Note that the ^ character must occur immediately to the right of the opening bracket.
\c	Removes (escapes) any special meaning of character c.
{ab,cd}	Matches a string from the string set {ab, cd}.
{ab,c{de,fh}}	Matches a string from the string set {ab, cde, cfh}.

## Data Exploration

You can explore data using one of the visual operators in the Exploration category.

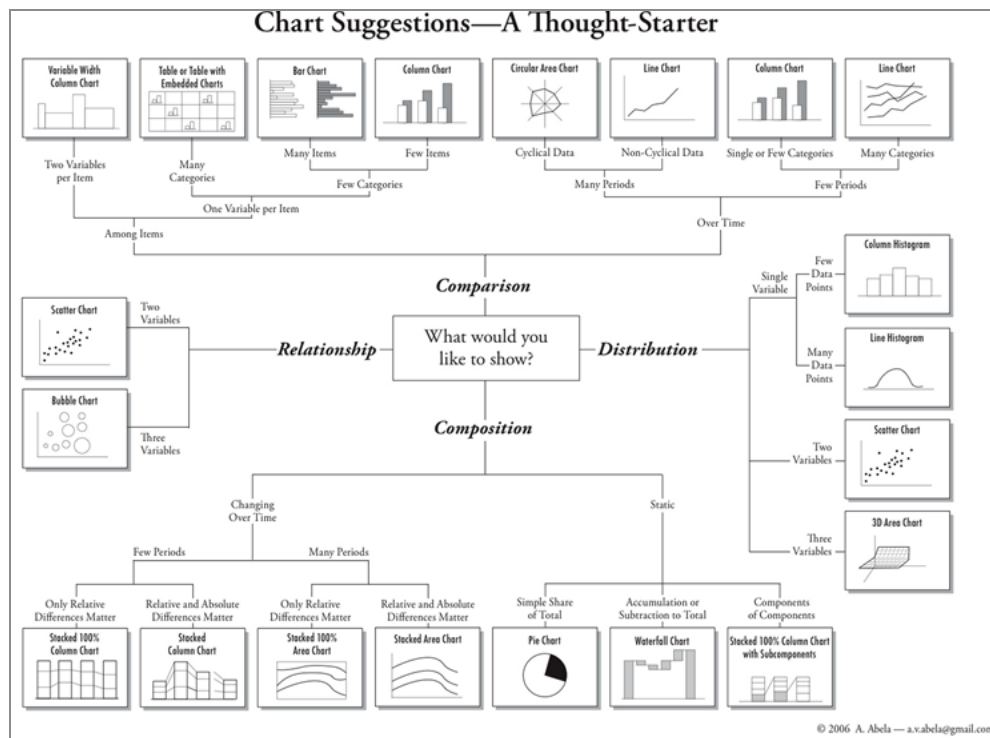
You can use the operator properties to fine-tune the visual output, run the operator, and then view the chart or graph in Team Studio. The correlation operator can provide information about [Correlation and Covariance](#).

Most [Exploration Operators](#) that produce visual output do not generate data to send to succeeding operators (they are terminal operators). [Line Chart](#), [Summary Statistics](#) (either for [Summary Statistics \(DB\)](#) or [Summary Statistics \(HD\)](#)), and [Variable Selection \(DB\)](#) are the exceptions, producing data that you can send to a subsequent operator. See their help for more information.

## Visualizing data with charts and graphs

Data visualization is at its most effective when you know which visualization type gives you the best understanding of the data.

The following graphic shows the thought process that a data modeler can use to decide among all of the charts or graphs available for [visualizing data](#).



## Explore Visual Results

TIBCO Data Science - Team Studio uses Plotly charts and graphs to enhance how you interact with your data. Plotly is available to the Explore operators, to many Model operators, and for all associated results shown in Touchpoints and HTML reports. The topics in this section discuss tips and tricks to use these visualizations to the fullest extent. Note the location of each of these features in the diagrams.

## Navigating the Results Panel

You can see the results of running a workflow in the Results panel.

Perform these tasks from a workflow that you have open for editing.



## Before you begin

You must have read and write access to the workflow.

## Procedure

1. In the workflow, select an operator.
2. In the lower left corner, click **Results** (indicated in the image by 1).

Optionally, in the lower right corner, you can click the left-most icon displaying the small up arrow (also labeled 1 in the image). When the Results panel is open, this icon is displayed as a small down arrow, which you can use to close the Results panel.

3. To increase the size of the Results panel, click **Expand**.

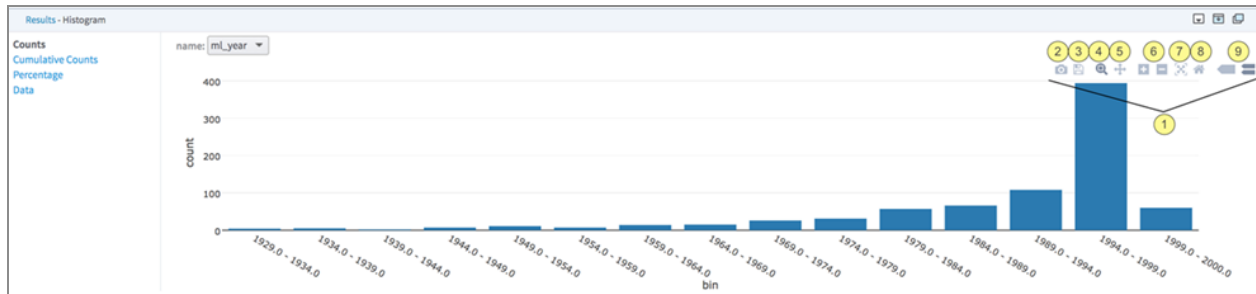
(Click again to return to the original panel size.)










Alternatively, you can open your visual results in a new window.

## Plotly Charts and Graphs

Plotly visualizations are available for Exploration operators and several Model operators.

After you access the Results panel, you can use the additional features provided with Plotly charts and graphs to help you explore your data. These additional features, shown in the following screen capture, are highlighted by numbers that correspond to the descriptions provided.



Reference number	Description
	To find the available actions, hover in the top right corner of your visualization.
	To download the visualization to your computer as a PNG file, use the Camera icon.
	To export your visualization to the Plotly online chart editor, use the <b>Save</b> icon. From the Plotly online chart editor, you can modify colors, data sort orders, and other characteristics of your visualization.
	To zoom for more detail of a visualization area, click and drag that area. This is the default action when you open the Plotly chart. If you change the default action, you can return to this option by clicking the associated icon, labeled in the image
	To switch from zoom to pan, use the <b>Pan</b> icon.
	To zoom manually, use <b>[+]</b> and <b>[-]</b> .
	To autoscale your visualization, use the crossed arrows icon.
	To return to the original view, click the <b>Home</b> icon.
	To display hover information, choose one of the two options: One shows values on both axes, and the other shows only the primary axis value.

# Correlation and Covariance

Correlation and covariance are statistical mathematical expressions that describe the degree at which two random variables or sets of random variables change their values in similar ways.

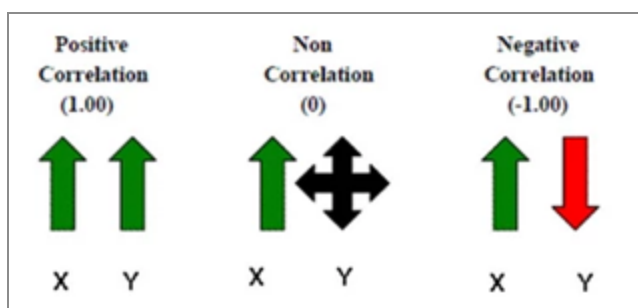
You can explore correlation and covariance and learn more about the operator algorithm by reading the help for the Correlation operator (either [Correlation \(DB\)](#), [Correlation \(HD\)](#), or [Correlation](#)).

## Correlation

The correlation between two random variables is a statistical measure of the relationship between the movements of the two variables. This relationship, which is expressed by what is known as the correlation coefficient, is represented by a value between -1.00 and +1.00.

- A correlation coefficient of +1.00 indicates that two variables move in the same direction at all times. If variable *X* gains in value, we would expect variable *Y* to gain in value as well.
- A correlation coefficient of 0 indicates that the movements are totally random; that is, a gain by variable *X* provides no insight into the expected movement of variable *Y*.
- A correlation coefficient of -1.00 indicates that the two variables move in opposite directions at all times. If variable *X* gains in value, we would expect variable *Y* to decrease in value.

The following image illustrates these relationships:



## Covariance

The covariance between two random variables is a statistical measure of how much the variables change together. Covariance is similar to the correlation algorithm, but without the data normalization.

- If the variables show similar changes, the covariance is positive.
- If the variables show opposite behavior, the covariance is negative.
- The sign of the covariance shows the tendency in the linear relationship between the variables.
- The magnitude of the covariance is not easy to interpret.

Variance is a special case of covariance, in which the two attributes are identical (that is, the covariance of an attribute with itself).

**i Note:** The [Correlation \(DB\)](#), [Correlation \(HD\)](#), and [Correlation](#) operators can calculate the covariance.

## Information Value and Weight of Evidence Analysis

Information Value analysis is a data exploration technique that helps determine which columns in a data set have predictive power or influence on the value of a specified dependent variable.

See [Information Value](#) operator for more information.

Information value analysis is a popular tool for banks, for example, providing a set of variables that help determine which credit card customers are most likely to default. The Information Value operator defines IV and WOE as follows:

- IV - A numerical value that quantifies the predictive power of an independent continuous variable  $x$  in capturing the binary dependent variable  $y$ . IV is helpful for reducing the number of variables as an initial step in preparing for Logistic Regression, especially when there are a large amount of potential variables. IV is based on an analysis of each individual independent variable in turn without considering other predictor variables.
- WOE - Closely related to the IV value, WOE measures the strength of each grouped attribute in predicting the desired value of the Dependent Variable.

The following table provides a standard rule of thumb for using the Information Value to understand the predictive power of each variable.

Information Value	Predictive Power
$< 0.02$	Useless
$0.02 - 0.1$	Weak
$0.1 - 0.3$	Medium
$0.3 - 0.5$	Strong
$> 0.5$	Suspiciously good; too good to be true

Typically, variables with medium and strong predictive powers are selected for model development. However, some schools of thought would advocate just the variables with medium IVs for a broad based model development.

In the above example, the times 90 days late, times 30 days late over 2 years, age, number of dependents, and education level of each person was assessed to see how predictive of paying late on a loan that variable was. Using the suggested results assessment chart above, number of times 30 days late over 2 years is a medium predictor (IV=0.13231212) of whether the person would pay late on a loan in the future.

## Data Transformation

Team Studio provides operators to clean and prepare data for use before modeling.

For example, some operators do not allow null values, so you might need to clean your data set by removing null or missing values. Your data might include data points that lie outside of the threshold for your consideration, so you might want to replace these outliers.

## Aggregation Methods for Batch Aggregation

In contrast to the Aggregation operator, which forces you to configure each aggregation separately, with the Batch Aggregation operator, you can select many numeric columns for each aggregation method and compute all these aggregations at once. The result is a wide data set that contains the grouping column and a column for each of the aggregations.

*Available Aggregation Methods*

<b>Aggregation Parameter Name</b>	<b>Prefix/Suffix</b>	<b>Formula</b>	<b>Performance Implications</b>
Count	group_size	The number of non-null members of this group.	
Minimum	min	The lowest value in each group.	
Maximum	max	The highest value in each group.	
Sum	sum	The sum of all of the values in each group.	
Mean	mean	Sum/Count for each group.	Online calculation performed using Spark SQL
Variance	var	The population variance:  $\text{avg}(\text{col} * \text{col}) - \text{avg}(\text{col}) * \text{avg}(\text{col})$	Online calculation performed using Spark SQL (v 1.5.1)
Standard Deviation	sd	The square root of the above.	Online calculation performed using Spark SQL (v 1.5.1)
Distinct	distinct	The number of distinct values in the group.	Calculated using Spark SQL (v 1.5.1) Slow if there are many distinct values within each group, or if no group was selected.



Aggregation Parameter Name	Prefix/Suffix	Formula	Performance Implications
Median <sup>1</sup>	median	<p>The middle element of the group. Specifically, we calculate median as the nth largest element in the group where</p> $n = \frac{(count+1)}{2}$	Expensive. Unlike the other values, cannot be calculated using highly performant Spark SQL optimizer. Requires an additional shuffle step. Might reach memory limitations if there are many groups.

## Outliers in Numerical Data

Your numerical data might include data that lies outside of a specified percentile threshold.

You can use the Replace Outliers operator (available for both database and Hadoop) to manage those outliers. For details about using the Replace Outliers operator, see the reference topics for your data source type ([Replace Outliers \(DB\)](#) and [Replace Outliers \(HD\)](#)).

Replace Outliers replaces all of the values above and below the specified percentile threshold with the maximum/minimum value within that threshold. Rather than providing an absolute minimum and maximum value to use as the thresholds, you can choose a lower percentile (**Lower Boundary %**) and upper percentile (**Upper Boundary %**).

For example, consider the following data set.

---

<sup>1</sup>This definition varies slightly from the formal definition of median. In cases where the number of elements is odd, the answer is the same. Usually the median averages the two middle elements, but we select the one at the count/2 position. Our answer might be lower than an averaged answer, depending on the range of the data. For example, suppose that the values in one group are (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Using count/2 to select the middle position, we get the fifth element, which is 4. Other statistical packages might calculate it as  $AVG(4,5) = 4.5$ . Our method of getting median is somewhat faster for distributed data, and on large data this difference is negligible.

StringColumn	Column1	Column2
first	0	5
second	10	5
third	20	5
fourth	30	5
fifth	40	10
sixth	50	5
seventh	60	8
eighth	70	8
ninth	80	9
tenth	90	5

If you set **Lower Boundary (%)** = 30.0 and **Upper Boundary (%)** = 90.0, the output is as follows.

StringColumn	Column1	Column2
first	0-20	5
second	10-20	5
third	20	5
fourth	30	5
fifth	40	10-9
sixth	50	5

StringColumn	Column1	Column2
seventh	60	8
eighth	70	8
ninth	80	9
tenth	90 80	5

For Column1, the ninth element is 80, and the third element is 20, so all the numbers above 80 and below 20 are replaced with those values. Because there are ten elements, the lower threshold is the third element (30% of 10 = 3) and the upper-threshold ninth element is the upper threshold because (90% of 10 = 9).

For Column2 (which is not sorted), the ninth element is 9. Thus, everything greater than 9 is replaced. The third element is 5. So everything lower than 5 (in this case, nothing) is replaced.

Now that you understand how replacing outliers works, try using the Replace Operators operator (either [Replace Outliers \(DB\)](#) or [Replace Outliers \(HD\)](#)).

## Creating a Join condition for a database join

You can perform a join on two tables using the Join operator. Start in the Join Properties dialog.

### Before you begin

You must have two tables in your workflow that can be joined. You must have specified the properties in the [Join](#) operator Properties dialog.

### Procedure

1. From the Join operator Properties dialog, click **Define Join Conditions**.  
The Join Properties - Database dialog is displayed.
2. If an existing Join condition exists, deselect it by clicking **Clear**.
3. Select the options to add for the new join condition.

For more information about the options, see [Join Properties - Database dialog](#).

a. Set **Left Table** first.

**Note:**

- For different join types, the same table can appear only once on the right side of **JOIN CONDITIONS**.
- When the selected **Join Type** is **CROSS JOIN** or **FULL OUTER JOIN**, there are no join conditions.

## Key-Value Pairs Parsing Example using the Variable Operator

You can use the Variable operator to parse data fields that are stored in a key-value pair format, such as JSON, dictionaries, and database STRUCT formats.

The following example illustrates how to refer to the state part of an address value stored as a STRUCT data type in an example Hive data source.

In this example, **address** is declared as a STRUCT data type column that stores the address fields as name-value pairs:

address
<pre>{   "segmentid": "CA",   "deliveryline": "ea TyqvQ",   "postdirection": "qV",   "smartsecstreetname": "Hr4FR8xx",   "city": "XoU6",   "cityabbr": "OH",   "state": "OH",   "zip": "45692",   "zip4": "9763",   "datereported": "07292014",   "isgeneraldelivery": false,   "ismilitary": false,   "isinvalidaddress": false,   "rentownbuy": "rentownbuy",   "ispobox": false,   "isruralroute": false,   "hasstandardaddress": true,   "address": {     "type": "S",     "streetname": "zwmh8",     "streetnumber": "xJa1e",     "streettype": "is187",     "apartme"   } }</pre>
<pre>{   "segmentid": "CA",   "deliveryline": "Wqv XTLC6 bkZMG",   "postdirection": "KF2",   "smartsecstreetname": "F6LvifCTm",   "city": "1OoC",   "cityabbr": "OH",   "state": "OH",   "zip": "45612",   "zip4": "9730",   "datereported": "02012005",   "isgeneraldelivery": false,   "ismilitary": false,   "isinvalidaddress": false,   "rentownbuy": "rentownbuy",   "ispobox": false,   "isruralroute": false,   "hasstandardaddress": true,   "address": {     "type": "S",     "streetname": "H76oh cb5s6",     "streetnumber": "aHS",     "streettype": "aHS",     "apartmentnumber": "aHS",     "urbanization": "aHS",     "stdaddrrecordcode": "S",     "scfstring": "456",     "phone": "qVJqQBwSFK",     "phc"   } }</pre>

Within the Variable definition configuration screen, the SQL expression refers to the column name (address) and the value label (state) to parse the actual value from the source STRUCT data type.

Variable Name	Data Type	SQL Expression		
address_street	VARCHAR	address.smartsecstreetname	multi	Delete
address_city	VARCHAR	address.city	multi	Delete
address_state	VARCHAR	address.state	multi	Delete
address_zip	VARCHAR	address.zip	multi	Delete

The SQL expression used would be <column name>.<value label>. In this example, we use address.state.

For information about the Variable operator, see [Variable \(DB\)](#) or [Variable](#).

## datetime Format Conversion Examples

You can use the Variable operator to convert datetime formats.

From the [Variable \(DB\)](#) operator, using the Define Variables dialog, you can specify a different output datetime format than that of the input column.

Variable Name	Data Type	Pig Expression		
convert_date	datetime yyyy-MM-dd	datetime_ddMMYYYY	multi	Delete

- chararray
- int
- long
- float
- double
- bytearray
- sparse
- datetime
- datetime yyyy-MM-dd
- datetime HH:mm:ss
- datetime yyyy-MM-dd 'HH:mm:ss
- datetime yyyy-MM-dd'T'HH:mm:ss.SSSZ
- datetime MM-dd-yyyy
- datetime MM/dd/yyyy
- datetime dd-MM-yyyy
- datetime yyyy-MM-dd HH:mm
- datetime yyyy-MM-dd'T'HH:mm:ss.SSSZ

You can override the default list of `datetime` data types (set by the Administrator in the Datetime Formats Preferences dialog, detailed in the *TIBCO® Data Science - Team Studio System Requirements*) as long as the [Joda-Time API](#) formatting is followed.

You can use a Pig Script user-defined function (UDF) to generate a new value from a `datetime` string, such as the `GetMonth` and `GetDay` functions shown in the following image.

Variable Name	Data Type	Pig Expression	multi	Delete
Month_1	int	GetMonth(datetime_yyyyMM)	multi	Delete
Month_2	int	GetMonth(datetime_YYYYMMdd)	multi	Delete
Day_1	int	GetDay(datetime_yyyyMM)	multi	Delete
convert	datetime_yyy	datetime_ddMMYYYY	multi	Delete

For a full list of available `datetime` related Pig functions, see [Apache Pig DateTime Functions](#).

Because some Pig functions have issues handling null values, instead use the following Team Studio alternatives for these functions.

Pig Function	Alpine Alternative
AddDuration	AddDurationAlpine
SubtractDuration	SubtractDurationAlpine
MillisecondsBetween	MillisecondsBetweenAlpine
SecondsBetween	SecondsBetweenAlpine
MinutesBetween	MinutesBetweenAlpine
HoursBetween	HoursBetweenAlpine
DaysBetween	DaysBetweenAlpine
WeeksBetween	WeeksBetweenAlpine
MonthsBetween	MonthsBetweenAlpine
YearsBetween	YearsBetweenAlpine



**Tip:** For assistance writing Pig Functions, click **Pig Syntax**.

## Spark SQL Syntax and Expressions

The following table lists the Spark SQL syntax and expressions for Numbers:

Function	Description	Example
ABS()	Computes the absolute value.	ABS(-3.2) returns 3.2
CEIL()	Computes the ceiling of the given value.	CEIL(2.4) returns 3
EXP()	Computes the exponential of the given value.	EXP(1) returns 2.71828
FLOOR()	Computes the floor of the given value.	FLOOR(4.6) returns 4
LOG()	Computes the natural logarithm of the given value.	LOG(1) returns 0
RAND()	Generate a random column with independence and identically distributed (i.i.d.) samples from U[0.0, 1.0].	RAND() returns [0.0,1.0]
ROUND()	Returns the value of the column e rounded to 0 decimal places.	ROUND(2.4) returns 2
SQRT()	Computes the square root of the specified float value.	SQRT(4) returns 2

The following table lists the Spark SQL syntax and expressions for Strings:

Function	Description	Example
CONCAT()	Concatenates multiple input strings together into a single string	CONCAT('a', 'b') returns ab
INSTR(str, String substring)	Locate the position of the first occurrence of a substring in the given string.	INSTR('entree', 'e') returns 1
LOCATE(String substr, str, int pos)	Locate the position of the first occurrence of a substring.	LOCATE('e', 'entree', 2) returns 5
LOWER(str)	Converts a string to lowercase.	LOWER('Alpine') returns alpine
UPPER(str)	Converts a string to uppercase.	UPPER('Alpine') returns ALPINE
REVERSE(str)	Reverses the string and returns it as a new string.	REVERSE('foobar') returns raboof
SUBSTRING(str, int pos, int len)	Substring starts at pos and is of length len when str is String type or returns the slice of byte array that starts at pos in byte and is of length len when str is Binary type.	SUBSTRING('Alpine', 2, 2) returns lp
TRIM(str)	Trim the spaces from both ends for the specified string.	TRIM(' a b ') returns a b
LENGTH(str)	Computes the length of a given string.	LENGTH('foobar') returns 6
TRANSLATE(src, String matchingString, String replaceString)	Translate any character in the src by a character in replaceString.	TRANSLATE('Alpine', 'Alp', 'F') returns Fine
REGEXP_EXTRACT(e, String exp, int groupIdx)	Extract a specific group matched by a Java regex, from the specified string.	REGEXP_EXTRACT('this:that', '(.*):(.*)', 2)



Function	Description	Example
		returns that
REGEXP_REPLACE(e, String pattern, String replacement)	Replace all substrings of the specified string value that match regexp with rep.	REGEXP_REPLACE("foo bar", "oo ar \\s", "") returns fb

The following table lists the Spark SQL syntax and expressions for Date-Timestamps:

Function	Description	Example
DAYOFMONTH()	Extracts the day of the month as an integer from a given date/timestamp/string.	DAYOFMONTH('2003-05-10') returns 10
HOUR()	Extracts the hours as an integer from a given date/timestamp/string.	HOUR('2003-10-12 15:02') returns 15
MINUTE()	Extracts the minutes as an integer from a given date/timestamp/string.	MINUTE('2003-10-04 05:12:23') returns 12
DATEDIFF( DATE_ADD (('datetime'), 2), 'datetime')	Returns the difference in date.	DATEDIFF( DATE_ADD (('2012-10-17'), 2), '2012-10-17') returns 2
CAST(TO_DATE(datetime_column) AS STRING)	Returns the input date as string in format 'yyyy-MM-dd'.	CAST(TO_DATE(2012-10-12) AS STRING) returns '2012-10-12'
CAST(UNIX_TIMESTAMP ('datetime_column') AS BIGINT)	Returns local Unix time in seconds.	CAST(UNIX_TIMESTAMP ('2003-01-29') AS BIGINT) returns 1043827200

Function	Description	Example
<p><b>Note:</b>  Spark SQL only supports the format 'yyyy-MM-dd HH:mm:ss' for timestamps and 'yyyy-MM-dd' for dates. All input <b>datetime</b> columns are implicitly converted to this timestamp format when loading the data frame and converted to the initial (or new specified) format when writing it back to the TIBCO Data Virtualization.</p> <p>If you are using strings with a different datetime format, then the format needs to be explicitly cast to the correct timestamp format: <code>UNIX_TIMESTAMP('03/03/2012', 'MM/dd/yyyy')</code>.</p>		

The following lists the Spark SQL syntax and expressions for Conditionals:

- `CASE WHEN LOWER(referring_url) LIKE 'www%.org%' THEN 'Non-Profit' ELSE 'Other' END`
- `CASE WHEN LOWER(referring_url) RLIKE '^www.*\\.org$' THEN 'Non-Profit' ELSE 'Other' END`

For more information, refer to the [Spark SQL documentation](#).

## Data Modeling and Model Validation

Team Studio provides a robust set of modeling operators and operators that provide validation for models.

For example, you can create classification models, logistic and linear regression models, or K-Means models. You can perform time series analysis or apply association rules.

## Cluster Analysis Using K-Means

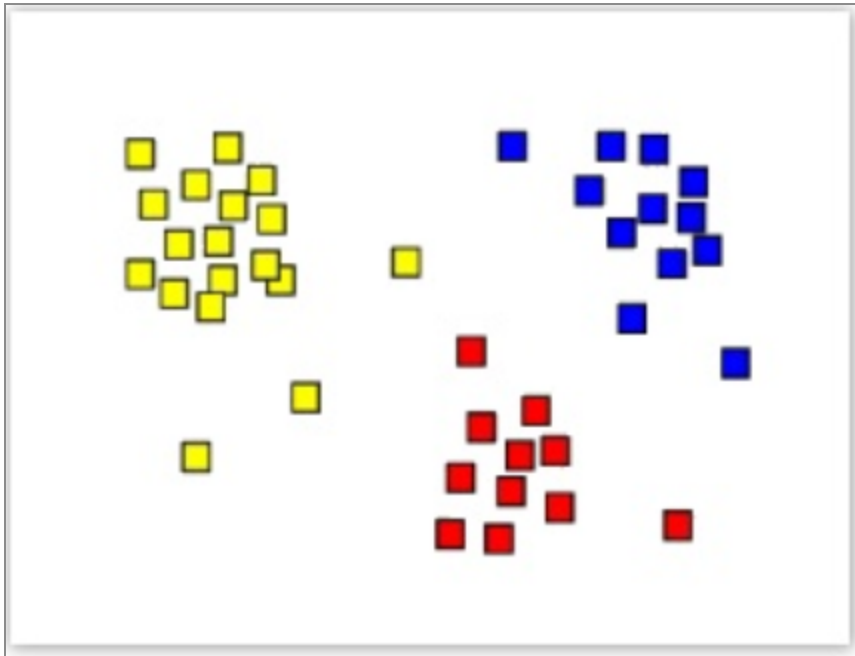
One of the most well-known and commonly used partitioning methods for cluster analysis.

Cluster analysis is a method of predictive analytics based on assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar to each other than to those in other clusters (in terms of their various attributes).

For detailed information about configuring K-Means operators in your workflow, see the Operator help.

- [K-Means \(DB\)](#)
- [K-Means \(HD\)](#)

A fundamental type of cluster analysis is partitioning, which organizes the objects of a set into several exclusive groups or clusters. Given a dataset of  $n$  observations and a  $k$  number of clusters to form, the K-Means analysis organizes or "partitions" the data into the  $k$  specified number of clusters that minimizes the square mean distance of the members from the center of the cluster. For example, the following diagram shows the partitioning of a dataset into three sub-groups or "clusters", as indicated by their separate coloring.



When deciding whether to use the K-Means Operator, it is helpful to understand that clustering is an "undirected" or "unsupervised" data mining technique, meaning that since its goal is to discover the structure of the data as a whole, there is no target value to actually be predicted. Therefore, no distinction is made between dependent and independent variables for the K-Means Operator.

## General Principles

Clustering techniques are used for combining observed examples into clusters (groups) which satisfy two main criteria:

- Each group or cluster is homogeneous; examples that belong to the same group are similar to each other.

- Each group or cluster should be different from other clusters, that is, examples that belong to one cluster should be different from the examples of other clusters.

This algorithm has as an input a predefined number of clusters, which is the  $k$  in its name.

Means stands for an average, an average location of all the members of a particular cluster.

When dealing with clustering techniques, one has to adopt a notion of a high dimensional space, or space in which orthogonal dimensions are all attributes from the table of data we are analyzing. The value of each attribute of an example represents a distance of the example from the origin along the attribute axes.

**i Note:** In order to use this geometry efficiently, the values in the data set must all be numeric (categorical data must be transformed into numeric ones) and should be normalized in order to allow fair computation of the overall distances in a multi-attribute space.

The K-means algorithm is a simple, iterative procedure, in which a crucial concept is the one of centroid.

- Centroid is an artificial point in the space of records that represents an average location of the particular cluster.
- The coordinates of this point are averages of attribute values of all examples that belong to the cluster.

The steps of the K-means algorithm[1] are:

1. Select randomly  $k$  points (it can be also examples) to be the seeds for the centroids of  $k$  clusters.
2. Assign each example to the centroid closest to the example, forming in this way  $k$  exclusive clusters of examples.
3. Calculate new centroids of the clusters. For that purpose, average all attribute values of the examples that belong to the same cluster (centroid).
4. Determine whether the cluster centroids have changed their "coordinates". If yes, start again from step 2). If not, cluster detection is finished and all examples have their cluster memberships defined.

Usually this iterative procedure of redefining centroids and reassigning the examples to clusters needs only a few iterations to converge. The K-Means algorithm computes a quality measure to optimize for the model which is calculated as follows:

$$E = \sum_{i=1}^k \sum_{p \in C_i} \text{dist}(p, C_i)^2$$

where  $k$  is the number of clusters,  $p$  is a member of cluster  $C_i$ , and  $\text{dist}$  is the distance of  $p$  from the center of  $C_i$  cluster.

## Analyzing K-Means Results

Once the cluster analysis is completed, the clusters need to be interpreted in order to have some real value for a data mining project.

Overall, after analyzing the various K-Means Operator results output, the modeler/business user might then decide to do one of the following:

- Refine the cluster analysis: for example, by choosing a different set of attributes, cluster points,  $k$  value, or distance calculation metrics.
- Use the cluster results holistically: the modeler might take the results of the K-Means analysis and use the information "as is". For example, cluster membership can be used as a label for a classification system. Interpreting the clusters from a business perspective (that is, most loyal customer cluster or violent criminal cluster) might be very informative for understanding what the naturally occurring clusters actually are.
- Target a specific cluster: based on the cluster results, it might make sense for a business to, for example, apply a marketing campaign to only a specific cluster of customers.

Run a Decision Tree or CART Classification Analysis: running a descriptive data mining technique, such as the Decision Tree or CART Operator, next based on the defined cluster groups might be useful for understanding the classification rules involved.

## K-Means Use Case

The K-Means operator is useful in analyzing various dataset types, including biological datasets (such as human genetic clustering, genotype assigning, sequence analysis, organism community analysis), medical data sets (such as tissue and blood clustering, IMRT segmentation), business and marketing data sets (such as market research clustering of the population of consumers, shopping cart item grouping, social network analysis), and social science data sets (such as crime analysis of hot spots of a similar crime and student/school clustering).

## Telecommunications Data Cluster Analysis

The following model demonstrates using a k-means cluster analysis model to get an overall understanding of the various naturally occurring telecommunication clusters of cell phone usage.

This use case exemplifies how cluster analysis can be applied to market research for partitioning a population of users or consumers into market groupings. This can be useful, for example, for market segmentation, product positioning, and targeting marketing campaigns.

### Datasets

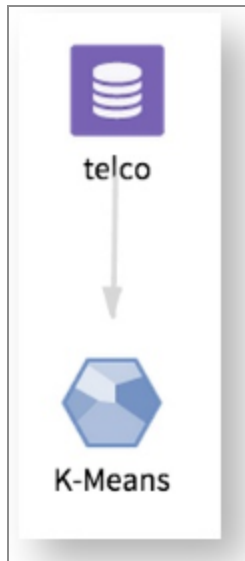
This dataset tracks various usage metrics for each customer (**Weekend\_minutes**, **Offpeak\_minutes**, **Peak\_minutes**) along with some customer demographic information.

The dataset used for this case is of 4MB. It contains 59 columns and 15654 observations, the first few rows and columns are shown below:

Cust_ID	Contract_C	Package	Gender	Age	Marital_Status	Living_Credit	Graduation	Job_Type	Income	Peak_min	Weekend_min	Offpeak_min	Offpeak_min	Peak_min	Weekend_min	Offpeak_min	Selfnet_min	Fixed_min	Othermb	Voicemail	Voicemail	SMS_05	Peak_min	Weekend_min	Offpeak_min	Offpeak_min
00000004	09/09/2004	PACK_B	Male	42	Married	Owner	University	Leader	15_30K	5.58	0.38	0	0	2	1	0.83	0	0	3	0.83	0	0	0	0	0	0
00000785	08/08/2004	PACK_B	Male	40	Married	Owner	Highschool	Labourer	30_60K	4.23	5.35	1.33	4	36	44	10.02	0	0	83	10.02	1	3.2	3.33	0	0	0
00000798	08/04/2004	PACK_B	Male	27	Single	Owner	University	Labourer	Over_60K	49.82	13.73	7.43	11	51	12	40.1	0.42	3.55	84	64.07	10	40.38	6.82	2.57	9	
00000821	04/02/2005	PACK_B	Female	26	Single	Other	University	Labourer	Over_60K	0	0.12	0	0	0	1	0.12	0	0	1	0.12	0	0	3.22	1.03	1	
00000799	03/10/2005	PACK_B	Female	50	Married	Owner	Highschool	Labourer	15_30K	17.4	0.62	0.13	2	27	1	1.88	3.55	10.9	30	18.15	10	7.43	0.88	1.83	3	
00000833	12/02/2004	PACK_A	Female	26	Married	Owner	University	Public_Employee	Over_60K	0	0	0.45	1	0	0	0.45	0	0	1	0.45	0	0	0	0	0	
00000416	12/09/2004	PACK_B	Male	32	Divorced	Owner	Highschool	Labourer	15_30K	17.27	10.1	11.07	4	15	7	11.42	8.25	18.97	36	38.79	0	11.52	13.83	12.67	7	
00000784	04/11/2005	PACK_A	Female	30	Married	Owner	University	Public_Employee	15_30K	2.71	3.95	6.35	2	11	7	11.73	1.38	4.96	20	28.1	0	21.93	13.25	7.88	10	
00000881	01/04/2005	PACK_B	Male	49	Married	Owner	Highschool	Executive	15_30K	19.73	25.32	66.27	15	20	19	90.07	8	13.25	60	111.32	0	13.75	106.43	182.23	30	
00000882	02/06/2005	PACK_B	Male	30	Married	Owner	Highschool	Labourer	30_60K	44.57	19.57	6.75	13	90	27	20.08	7.68	43.12	100	70.88	20	19.3	109.95	4.23	12	
00000106	05/12/2005	PACK_B	Male	48	Married	Owner	Highschool	Leader	15_30K	1.18	3.03	0	0	7	8	1.4	0.4	2.35	15	4.32	0	0	0	0	0	
00000207	02/01/2005	PACK_A	Male	24	Single	Owner	Highschool	Labourer	15_30K	49.43	5.62	5.43	10	94	9	23.2	0	33.48	100	96.25	11	44.18	1.02	4.22	10	
00000505	02/12/2005	PACK_B	Female	19	Married	Owner	University	Public_Employee	Below_15K	11.32	6.53	6.98	36	37	19	6.7	8.82	10.12	21	21.5	58	21.7	5.58	15.94	52	
00000793	04/14/2005	PACK_A	Female	28	Single	Owner	Highschool	Executive	15_30K	56.2	53.62	69.98	38	46	46	91.73	0	104.28	126	106.02	21	9.62	133.03	37.48	18	
00000158	03/17/2006	PACK_A	Female	31	Married	Owner	University	Public_Employee	15_30K	22.78	12.18	15.57	12	15	10	44.6	0.52	9.42	37	54.53	0	38.97	54.37	28.03	24	
00000122	10/09/2004	PACK_B	Male	19	Single	Family_Member	University	Executive	30_60K	0	0	0.07	1	0	0	0.07	0	0	1	0.07	0	0	0	0	0	
00000012	11/15/2004	PACK_B	Male	45	Married	Owner	Highschool	Labourer	15_30K	6.23	4.78	2.73	7	8	9	5.1	0	8.65	22	13.75	2	0	0.02	0	0	
00000135	11/04/2004	PACK_B	Male	31	Married	Owner	Highschool	Labourer	15_30K	0	0.45	0	0	0	3	0.45	0	0	3	0.45	0	0	0	0	0	
00000193	12/05/2004	PACK_B	Male	35	Married	Other	Highschool	Public_Employee	Below_15K	0.12	0.27	0	0	1	3	0.36	0	0	4	0.36	0	0	0	0	0	
00000206	10/10/2004	PACK_A	Male	43	Married	Owner	Highschool	Executive	30_60K	18.05	3.9	8.43	5	103	9	19.07	3.83	67.88	118	90.38	1	24.9	15.53	2.83	9	
00000225	12/05/2004	PACK_B	Male	34	Married	Owner	Highschool	Executive	15_30K	110.92	14.9	22.03	27	92	30	51.73	23.17	64.77	118	146.95	31	115.48	59.92	27.38	64	
00000277	01/17/2005	PACK_A	Male	32	Widow	Owner	Highschool	Labourer	15_30K	0	0	0.07	1	0	0	0.07	0	0	1	0.07	0	0	0	0	0	
00000039	11/14/2004	PACK_B	Male	22	Married	Owner	Highschool	Public_Employee	30_60K	18.63	19.23	13.4	15	24	14	8.35	4.25	38.07	43	50.67	10	16.73	17.73	17.4	20	
00000345	10/31/2004	PACK_B	Male	32	Married	Owner	Highschool	Labourer	15_30K	0.08	0	0	0	1	0	0.08	0	0	1	0.08	0	0	0	0	0	
00000354	12/05/2004	PACK_B	Male	26	Married	Owner	University	Public_Employee	Over_60K	0	13.18	0	0	0	2	0	0	13.18	2	13.18	0	2.98	0	0	0	

## Workflow

The setup of this Analytic Flow is simply the data source followed by the K-Means operator.



The K-Means Operator configuration properties can be kept at their default values. The **ID** column should be set to the Cust\_ID field.

As a quick assessment, the modeler might select three cell phone usage metrics of weekend\_minutes, peak\_minutes, and offpeak\_minutes for a quick K-Means analysis.

## Results

Running the K-Means analysis produces the following results:

The Summary results show the number of clusters, as defined by the *K* value selected, and the **Average Distance Measurement** of 41.66, which has "minutes" as the units of measure for the cell phone usage.

Results - K-Means-1	
Summary Cluster Profiles K-Means Center Points Warning Text Scatter Plots	Cluster Column Name : "alpine_cluster" Cluster Count : 3 Average Distance Measurement : 41.65967083467749

The **Cluster Profiles** tab displays the different natural states for each of the usage metric variables and the profile for each cluster group in those states.

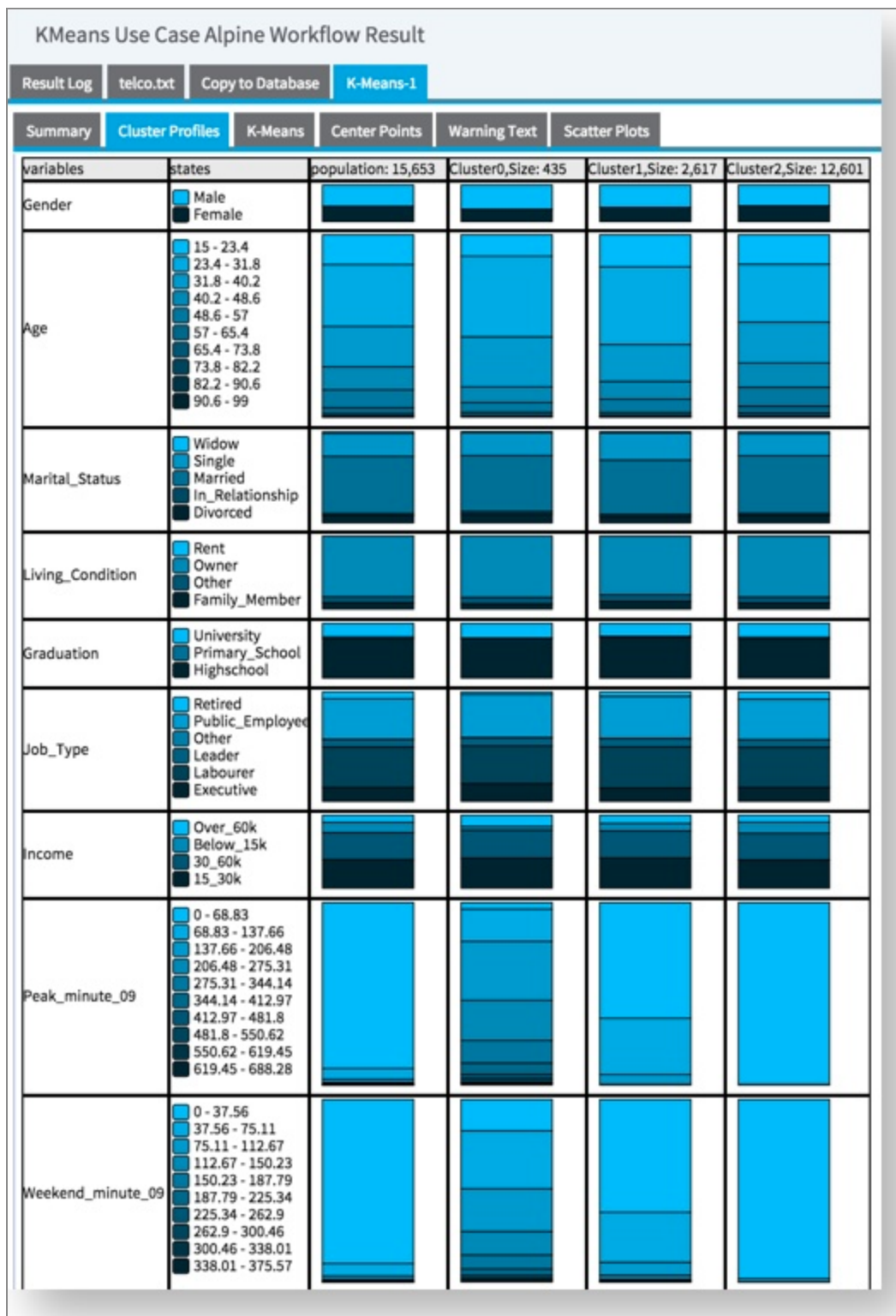
In this example, **Cluster0** has a variety of users falling into the different number of usage minute groupings (and the lowest user population of 435 users). **Cluster1** has only low

usage users for *Peak\_minutes* and a small fraction of 2nd lowest usage groupings for *Weekend\_minute* and *Offpeak\_minute* variables (and a population size of 2,617 users).

**Cluster2** has 100% lowest usage users for all three variables (and the highest population of 12,601 users).

**Cluster1** and **Cluster2** do not look very different from each other, but **Cluster0** has unique characteristics, with a higher percentage of overall cellphone usage.





The **K-Means** tab displays the assignment of each individual Cust\_ID (cellphone user) to a

particular cluster. This assignment information could be very valuable for targeting a marketing campaign that, for example, at only **Cluster0** users who show the highest usage levels overall.

Results - K-Means 1

Summary  
Cluster Profiles  
K-Means  
Center Points  
Warning Text  
Scatter Plots

	Gender	Age	Marital_Status	Living_Condition	Graduation	Job_Type	Income	Peak_minutes_0	Weekend_minu	Offpeak_minutes	Offpeak_nr_00	Peak_nr_00	Weekend_nr_00	Cust_ID	alpine_cluster
	Female	23	Single	Owner	Highschool	Executive	15_30k	5.67	0.85	0.5	3	12	5	ID1255577	2
	Male	43	Married	Owner	Highschool	Labourer	30_60k	8.73	0	0	2	23	2	ID0605092	2
	Male	26	Single	Family_Member	Highschool	Public_Employee	15_30k	76.77	102.57	87.72	96	134	85	ID1255309	1
	Male	22	Single	Family_Member	Highschool	Public_Employee	30_60k	19.5	33.38	51.8	81	47	71	ID0956429	1
	Female	22	Single	Owner	Highschool	Public_Employee	30_60k	20.28	3.87	19.67	18	38	13	ID1451628	2
	Male	29	Married	Owner	Highschool	Executive	30_60k	117.22	29.52	77.5	62	104	35	ID1652573	1
	Female	31	Single	Owner	Highschool	Labourer	30_60k	1.22	0.35	2.5	4	3	5	ID0352204	2
	Male	32	Married	Owner	Highschool	Executive	15_30k	24.2	2.02	0.68	2	25	3	ID0651368	2
	Male	29	Single	Other	Highschool	Labourer	30_60k	0.52	0.1	0.78	2	1	2	ID0333703	2
	Female	31	Single	Owner	University	Public_Employee	15_30k	1.42	1.03	0.3	3	10	4	ID0350017	2
	Female	44	Married	Owner	Highschool	Public_Employee	30_60k	30.82	6.97	8.43	7	41	20	ID0720396	2
	Male	20	Single	Family_Member	University	Leader	30_60k	19.6	4.28	6.18	8	43	21	ID0252843	2
	Female	26	Married	Owner	University	Public_Employee	Over_60k	0.17	0	0	0	2	0	ID0451727	2
	Male	49	Married	Owner	Highschool	Labourer	15_30k	0.38	0	0	0	2	0	ID0720792	2
	Female	24	Single	Family_Member	Highschool	Labourer	15_30k	0	0	0	4	7	0	ID0156016	2
	Male	22	Married	Other	Highschool	Labourer	30_60k	20.03	2.95	8.37	24	47	19	ID0452536	2
	Female	26	Divorced	Owner	Highschool	Public_Employee	Below_15k	46.3	6.87	14.42	16	44	6	ID0354489	2
	Male	21	Married	Family_Member	Highschool	Leader	30_60k	15.12	10.68	75.53	64	50	26	ID0555439	1
	Male	23	Married	Owner	Highschool	Executive	30_60k	1.62	13.75	7.67	36	4	15	ID1251960	2
	Male	34	Married	Owner	Highschool	Labourer	15_30k	6.7	30.48	32.83	12	4	8	ID1828612	2
	Female	52	Married	Owner	Highschool	Public_Employee	15_30k	2.83	4.87	0	0	4	13	ID1256666	2

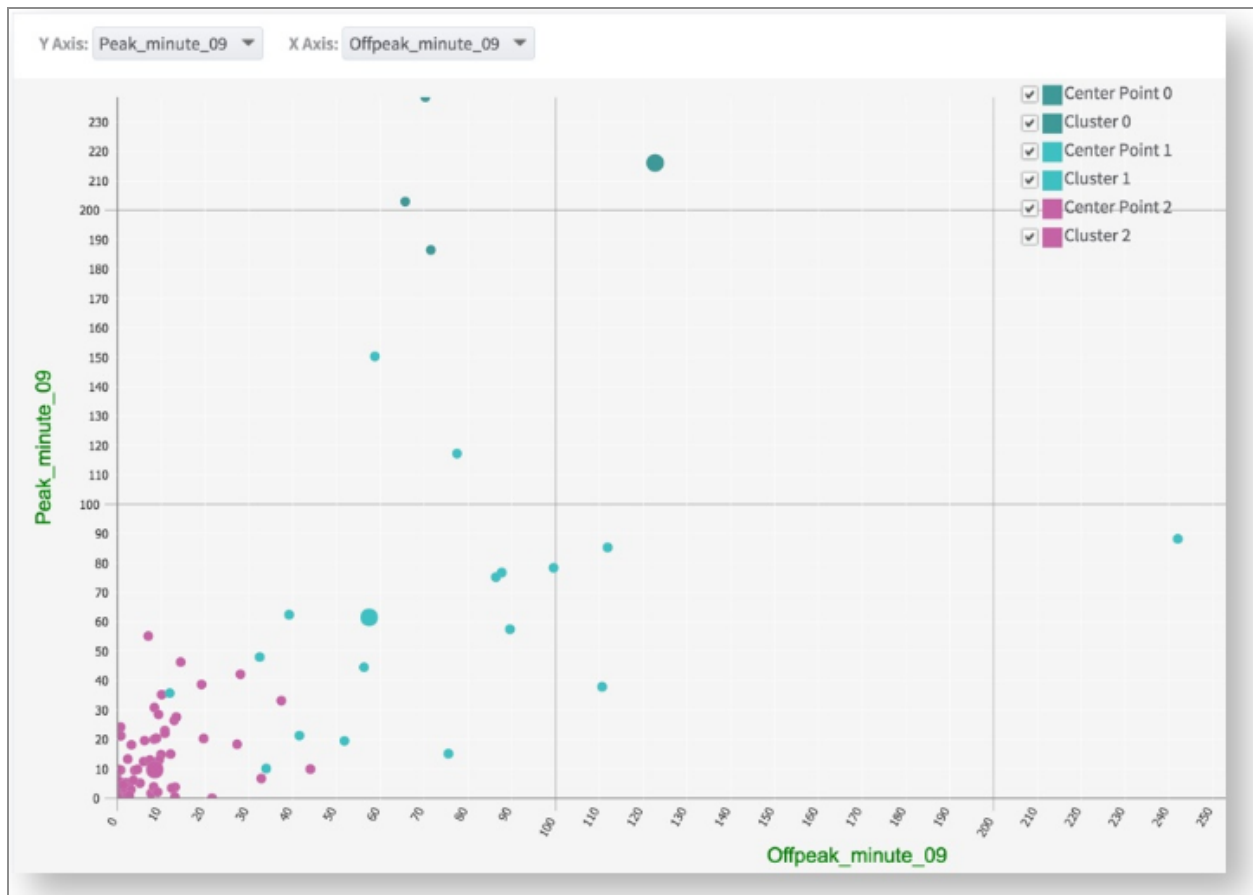
The **Center Point** tab displays the center point values for the cell phone usage of all three clusters. For example, **Cluster0** has a center of 205 *Peak\_minutes*, 92 *Weekend\_minutes*, and 141 *Offpeak\_minutes*. **Cluster2** has a center of only 9.2 *Peak\_minutes*, 5.5 *Weekend\_minutes*, and 6.5 *Offpeak\_minutes*.

Results - K-Means 1

Summary  
Cluster Profiles  
K-Means  
Center Points  
Warning Text  
Scatter Plots

	Cluster	Gender_Male	Age	Marital_Status	Marital_Status	Marital_Status	Marital_Status	Living_Condit	Living_Condit	Living_Condit	Graduation_00	Graduation_Pi	Job_Type_Pec	Job_Type_Pec	Job_Type_00i	Job_Type_Lea	Job_Type_Lat	Income_Over
	0.0000000	0.6666667	32.7943200	0.0124155	0.2512416	0.6141532	0.0312925	0.0000000	0.8424638	0.0809421	0.2671233	0.0068493	0.0228310	0.3949772	0.0322831	0.0730094	0.3447488	0.1329680
	1.0000000	0.6145316	32.2080830	0.0164456	0.2852199	0.5861759	0.0217973	0.0026709	0.8022944	0.0806319	0.3351816	0.0156788	0.0413002	0.3847056	0.0305093	0.0722753	0.3778202	0.1139579
	2.0000000	0.5736926	35.2849200	0.0233333	0.2395238	0.6342064	0.0240476	0.0023941	0.8344445	0.0793651	0.2498626	0.0354762	0.0677778	0.3684921	0.0023048	0.0687302	0.3671429	0.0993968

The **Cluster** tab displays the different usage metric variables charted against each other with the clusters shown in a different color. For this example, **Cluster0** has the widest scatter of data points with the highest usage values. **Cluster1** fits in the middle and **Cluster2** has the lowest usage values.



This Cluster graph can be displayed using a different combination of the variables to see how the clustering looks across different variable dimensions. For example, choosing the Y Axis to be *Peak\_minutes* with the X Axis as *Weekend\_minutes* shows the following results.



The clusters seem to be even more distinct when involving the *Weekend\_minutes* dimension.

Since  $K$  needs to be determined in advance for K-Means, the modeler might want to experiment with varying the  $K$  amount in order to see if the *Distance* value is decreased or if the clusters show more natural groupings or not.

## Patterns in Data Sets

A pattern in a data set is referred to as an "Association." It can be a set of items, subsequences, substructures, and so on, that occurs frequently in a dataset. It is also known as a frequent pattern.

You can use the Association Rules operator to discover frequent patterns in your data. For details about using the Association Rules operator, see [Association Rules](#).

Specific examples of the inherent regularities in data that Association Rule modeling can find include the following.

- Frequent itemsets: For example, a set of items, such as milk and bread, might appear frequently together in a transaction dataset.
  - What products are often purchased together? This is commonly referred to as Shopping Basket Analysis or Market Basket Analysis.
- Frequent substructures: A substructure can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets of subsequences. For example, a substructure such as certain DNA structures that are sensitive to a new drug might occur frequently in a biotechnology drug testing dataset.
  - What are substructures of the data associated with a certain event?

Association Rules are *if/then* statements that define relationships between seemingly unrelated data. An example of an association rule might be "If a customer buys a dozen eggs, he is 80% likely to also purchase milk."

- The *if* part of the rule is often referred to as the premise.
- The *then* part of the rule is the conclusion

Therefore, a premise can be considered to be an item or condition that is found frequently in combination with the conclusion item or condition.

The Association Rules performs the following analysis.

1. Finds the frequent item sets in the data by applying the parallel version of FP-Growth algorithm from MLLib.
2. Runs a parallel Association Rules algorithm to produce a list of uncovered Association Rules that have a single item as the consequent and which meet the modeler's specified Support, Confidence, and Lift criteria.
  - **Support** is an indication of how frequently the items appear together in the input.
  - **Confidence** indicates the percentage of times the *if/then* statements have been found to be true.
  - **Lift** measures the ratio of the observed support to that expected if the antecedent and the consequent were independent (the degree to which the antecedent and consequent are dependent on another, which makes the rule

valuable).

## Use Cases

Association Rules modeling is useful when analyzing unsupervised transactional data that is categorical in nature. Finding such frequent patterns can be applied to a variety of business use cases, such as shopping basket analysis. It studies customers' buying habits by searching for item sets that are frequently purchased together (or in sequence). Other common use cases include cross-marketing, product clustering, catalog design, store layout, sales campaign analysis, Web log ("click stream") analysis, and DNA sequence analyses.

## Alpine Forest Operators

TIBCO Data Science - Team Studio provides modeling, evaluation, validation, and prediction operators for regression (continuous) or classification (categorical) machine learning applications.

Operator	Data source type	Type of modeling
<a href="#">Alpine Forest Regression</a>	Hadoop	Applies an ensemble algorithm to make a numerical prediction by aggregating (majority vote or averaging) the numerical regression tree predictions of the ensemble. Applies to continuous (numerical) data.
<a href="#">Alpine Forest Classification</a>	Hadoop	Applies an ensemble classification method of creating a collection of decision trees with controlled variation. Applies to categorical data.
<a href="#">Alpine Forest Evaluator</a>	Hadoop	Provides model accuracy data that illustrates the classification model's accuracy for each possible predicted value, and an error convergence rate graph.
<a href="#">Alpine Forest - MADlib</a>	Database	Applies a MADlib function to generate multiple decision trees, the combination of which is used to make a prediction based on several independent columns.

Operator	Data source type	Type of modeling
<a href="#">Alpine Forest Predictor - MADlib</a>	Database	Uses the model trained by Alpine Forest (MADlib) and scores the results. It must be connected to an Alpine Forest (MADlib) operator.

## Ensemble Decision Tree Modeling with Alpine Forest

TIBCO Data Science - Team Studio provides a number of forest modeling operators for Hadoop and database data sources.

**Important:** If you are using the Alpine Forest Classification operator for database sources from version 6.0 or earlier, then you must remove the operator from your workflow and replace it with [Alpine Forest - MADlib](#). You must use [Alpine Forest Predictor - MADlib](#) with it.



Like the Decision Tree operator, [Alpine Forest Classification](#) creates a tree-like branching series of computational steps or logic "tests" that lead to an ultimate decision value. The difference is that the Alpine Forest Classification operator creates multiple decision trees, with each tree differing slightly. Specifically, each tree works on a random subset of the training data and uses a random subset of variables at the decision nodes.

Alpine Forest Classification is, therefore, an "ensemble" method combining a "forest" of individual decision trees that are generated using a random selection of attributes at each node to determine the split. The final classification decision is determined by a "vote count" of the most frequent classification across all of the resulting trees.

The main idea behind Alpine Forests is that, by creating many different decision trees and assuming that each individual tree makes mistakes in different places, the group of trees together should know, on average, the right answer in most of the places. Therefore, the aggregate tree results are hopefully more accurate than a single tree's results.

### **Alpine Forest Classification Modeling Advantages**

Alpine Forest Classification modeling is considered to be one of the most accurate learning algorithms currently available, producing highly accurate categorical classification results. Additional advantages of Alpine Forest Classification modeling include:

- Ability to automatically select variables from a large set of predictors, without the modeler first having to reduce the variables to only strong predictors first.
- Ability to work well "off-the shelf" without major configuration. A modeler can get quick, relatively accurate results within a few minutes.
- Ability to accept thousands of input predictor variables without variable deletion. In other words, it handles "wide" data where there are more predictors than observations and does not need to do some sort of variable reduction process first, like other methods usually have to do.
- Ability to indicate which variables are important for the classification.
- Ability to generate built-in, cross-validation error estimates of model accuracy as the forest building progresses.
- Ability to pick up very non-linear boundaries or interactions between variables.
- Ability to handle large datasets efficiently.

Some disadvantages of the Alpine Forest Classification method include the tendency to overfit for some datasets (that is, if the number of trees is set too high) and that the resulting Alpine Forests are difficult for humans to interpret and visualize. Also, for data including categorical variables with varying numbers of levels, Alpine Forests tend to be biased in favor of those attributes with more levels.

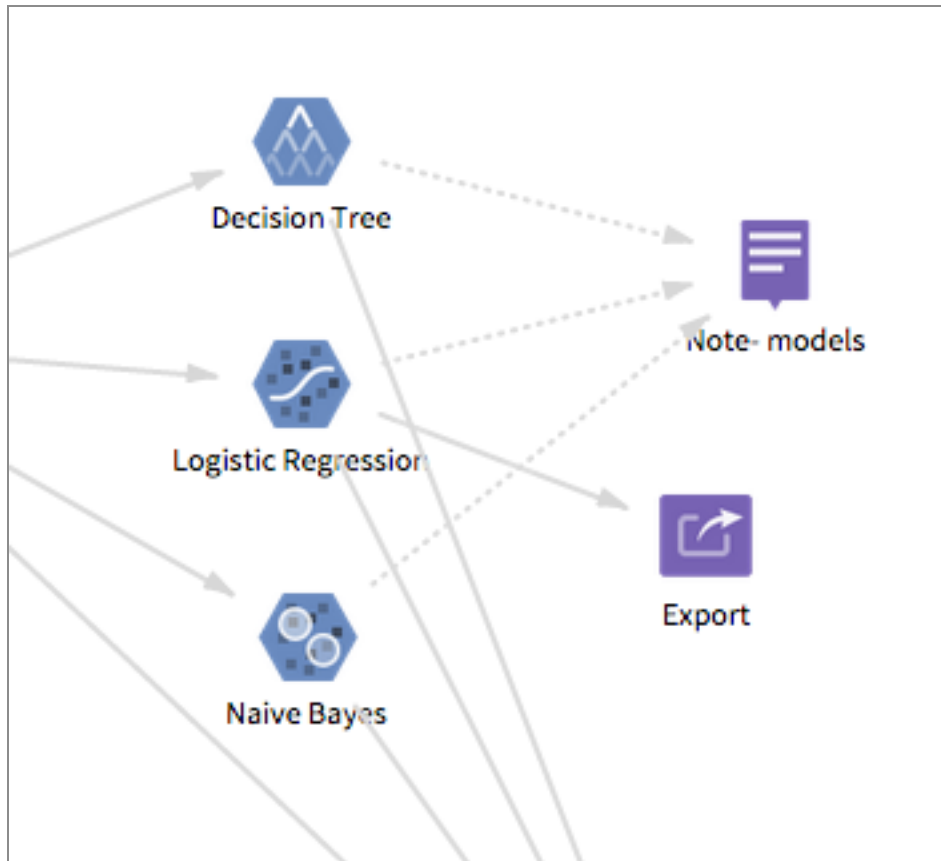
### **Alternative Models**

- [Classification Modeling with Naive Bayes](#)
- [Probability Calculation Using Logistic Regression](#)
- [Support Vector Machine Classification](#)



## Model Export Formats

You can use the Export operator to export a model to a variety of formats.



The model can be stored as a PMML (Predictive Model Markup Language) formatted file, a Team Studio Analytics Model (AM) formatted file, or a Portable Format for Analytics (PFA) file. For the operator configuration properties, input, and output information, see [Export](#).

- Model files can be imported for use in other Team Studio work files via the [Load Model](#) operator.
- PMML files generated using Team Studio can be executed with JPMML for scoring. For example, PMML files can be pushed directly to the TIBCO Streaming Artifact Management Server for scoring in TIBCO® Streaming.
- PFA files can be exported for use in real-time scoring engines and are quickly becoming an industry standard.

**i Note:** The Team Studio administrator must configure an existing TIBCO Streaming Artifact Management Server in the configuration file `alpine.conf` by adding the following information. See your administrator for more information.

```
streambase {
  # streaming login info - defaults work with a locally
  installed instance
  enabled = false
  hostname = localhost
  port = 2185
  https = false
  default_login = ""
  default_pwd = ""
  default_autocommit = true
}
```

The `default_login` and `default_pwd` populates parameters available to all end users. Leave these values empty.

To learn more about TIBCO Streaming and the TIBCO® Artifact Management Server, visit our [website](#).

## PMML Export

The following model operators are supported for PMML export.

Data Source	Supported Model Operators
Hadoop	<ul style="list-style-type: none"> <li>Alpine Forest Classification</li> <li>Alpine Forest Regression</li> <li>Naive Bayes</li> <li>Linear Regression</li> <li>Logistic Regression</li> <li>SVM Classification</li> <li>K-Means</li> </ul>

Data Source	Supported Model Operators
Database	<ul style="list-style-type: none"> <li>• Linear Regression</li> <li>• Logistic Regression</li> <li>• K-Means</li> </ul>

### Analytics Model (AM) Export

The following model operators are supported for AM export.

Data Source	Supported Model Operators
Hadoop	<ul style="list-style-type: none"> <li>• Linear Regression</li> <li>• Logistic Regression</li> <li>• K-Means</li> <li>• PCA</li> <li>• Gradient Boosting Classification</li> <li>• Gradient Boosting Regression</li> <li>• Normalization</li> </ul>
Database	<ul style="list-style-type: none"> <li>• Linear Regression</li> <li>• Logistic Regression</li> <li>• K-Means</li> <li>• Naive Bayes</li> <li>• Variable</li> </ul>

Additionally, any Custom Operators that extend the `ModelWrapper` class can export to AM format. for more information about developing custom operators, see the *TIBCO® Data Science - Team Studio Development Kit* at [www.docs.tibco.com](http://www.docs.tibco.com)

## PFA Export

The following model operators are supported for PFA export.

Data Source	Supported Model Operators
Hadoop	<ul style="list-style-type: none"><li>• Linear Regression</li><li>• Logistic Regression</li><li>• K-Means</li><li>• PCA</li><li>• Normalization</li></ul>
Database	<ul style="list-style-type: none"><li>• Linear Regression</li><li>• Logistic Regression</li><li>• K-Means</li></ul>

## Fitting a Trend Line for Linearly Dependent Data Values

Linear regression is the statistical fitting of a trend line to an observed dataset, in which one of the data values - the dependent variable - is found to be linearly dependent on the value of the other causal data values or variables - the independent variables.

- The dependent variable is sometimes called the prediction, and the independent variables the predictors.
- The TIBCO Data Science - Team Studio Linear Regression operator is the simplest and one of the most frequently used modeling operators. For information about configuring these operators, see [Linear Regression \(DB\)](#) or [Linear Regression \(HD\)](#).
- Typically, a linear regression should be the first method attempted for determining the relationship between a continuous, numeric variable and a set of causal variables before any more complex methods are tried.

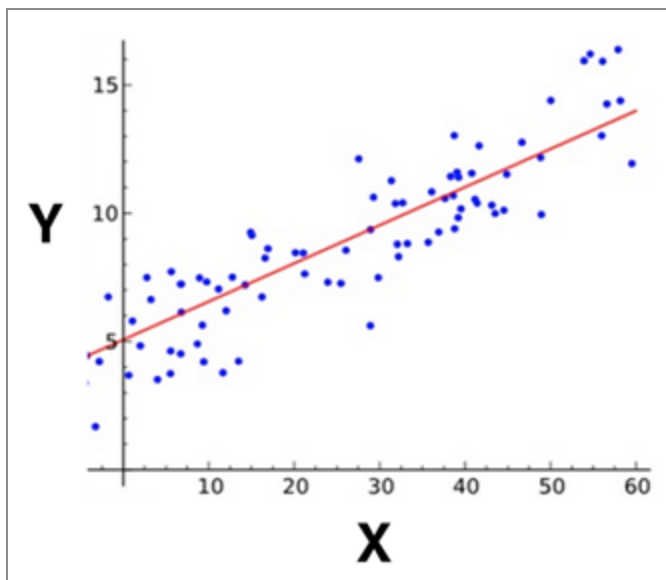
Linear regression is an approach to modeling the relationship between a dependent variable  $Y$  and one or more explanatory, or predictor, variables denoted  $X$ . If there is a

linear association, a change in  $X$  has a corresponding change in  $Y$ . This relationship is analyzed and estimated in the form of a Linear Regression equation, such as

$$Y = \beta_1 X_1 + \beta_2 X_2 + \dots$$

where  $\beta_1, \beta_2, \dots$  act as scaling factors.

In other words, Linear Regression is the statistical fitting of a trend line to an observed dataset, in which one of the data values is found to be dependent on the value of the other data values or variables.



Example of [simple linear regression](#) with only one explanatory variable  $X$ .

Single variable or Simple Linear Regression is easy to understand since it can be represented as trying to best fit a line to an  $XY$  dataset, as illustrated above.

The case where only one explanatory variable  $X$  is involved is called Simple Linear Regression. Single variable or Simple Linear Regression can be represented as trying to best fit a line to a  $XY$  dataset. When a dataset has more than one independent variable involved, it is called Multivariate Linear Regression (MLR). The algebra behind a Multivariate Linear Regression Equation for predicting the Dependent Variable  $Y$  as related to the Independent Variables  $X$  can be generically expressed in the following form:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \dots$$

Component	Description	Function
$\hat{Y}$	Dependent Variable	Predicted dependent variable value $Y$ based on the values of the independent variables $X$ .
$\beta_0$	Intercept	A fixed constant value that is the value of $Y$ when all the $X$ values are zero.  This is sometimes referred to as alpha.
$X_i$	Independent Variables	The values of the independent variables are found to affect the value of the dependent variable $Y$ . For linear regressions, the value of $Y$ changes directly or linearly in relation to the value of $X$ changing.
$\beta_i$	Coefficient	The scaling factor or Coefficient value, beta, which quantifies how strongly the value of $X$ affects the value of $Y$ . Specifically, the interpretation of $\beta_i$ is the expected change in $Y$ for a one-unit change in $X$ when the other covariates are held fixed.

The TIBCO Data Science - Team Studio Linear Regression operator runs an algorithm on the  $XY$  dataset to determine the best fit values for the Intercept constant,  $\beta_0$ , and Coefficient values,  $\beta_i$ .

There are various ways to estimate such a best fit linear equation for a given dataset. One of the most commonly used methods, also used by the TIBCO Data Science - Team Studio Linear Regression Algorithm, is the ordinary least-squares (OLS) approach. This method calculates the best-fitting line for the observed data by minimizing the sum of the squares of the vertical deviations of each data point to the line (if a point lies on the fitted line exactly, then its vertical deviation is 0). Because the deviations are first squared, then summed, there are no cancellations between positive and negative values.

The following diagram depicts this concept of minimizing the squares of the deviations,  $d$ , of the data points from the linear regression.

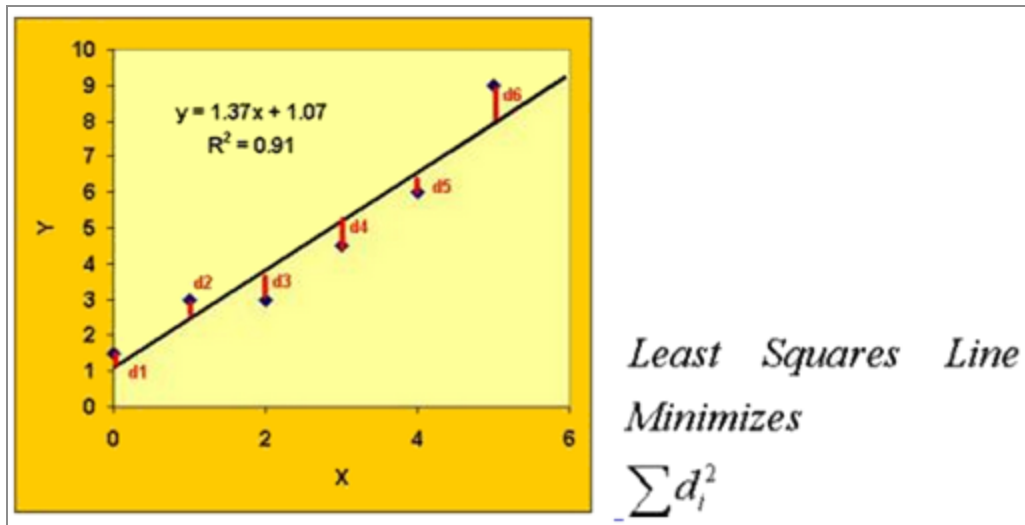


Illustration of the Ordinary Least Squares Method of Linear Regression Estimation<sup>1</sup>  
Also calculated during the least-squares method is a Correlation Coefficient,  $R$ , which varies between -1 and +1.

$$R = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

where  $\bar{x}$  and  $\bar{y}$  are the line data values and  $x_i$  and  $y_i$  are the actual data values.

The square of the correlation coefficient,  $R^2$ , is useful for understanding how well a linear equation fits the analyzed dataset.  $R^2$  represents the fraction of the total variance explained by regression. This statistic is equal to one if the fit is perfect, and to zero when the data shows no linear explanatory power whatsoever.

For example, if the  $R^2$  value is 0.91, 91% of the variance in  $Y$  is explained by the regression equation.

### Regularization of Linear Regression

The Ordinary Least Squares approach sometimes results in highly variable estimates of the regression coefficients, especially when the number of predictors is large relative to the number of observations. To avoid the problem of over-fitting the Regression model, especially when there is not a lot of data available, adding a regularization parameter (or constraint) to the model can help reduce the chances of the coefficients being arbitrarily stretched due to data outliers. Regularization refers to a process of introducing additional information in order to prevent over-fitting, usually in the form of

a penalty or constraint for data complexity.

Three common implementations of Linear Regression Regularization include Ridge, Lasso, and Elastic Net Regularization.

### L2 Regularization (Ridge)

$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^p \beta_j^2$$

Minimizes the quantity above. The coefficients shrink towards zero, although they never become exactly zero. Ridge constrains the sum of squares of the coefficients in the loss function. L2 Regularization results in a large number of non-zero coefficients.

### L1 Regularization (Lasso)

$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^p |\beta_j|$$

Minimizes the quantity above. The coefficients shrink toward zero, with some coefficients becoming exactly zero to help with variable selection. Lasso constrains the sum of the absolute values of the coefficients in the loss function. L1 Regularization gives sparse estimates. Namely, in a high dimensional space, many of the resulting coefficients are zero. The remaining non-zero coefficients weigh the explanatory variable(s) (X) found to have importance in determining the dependent variable, Y.

### Elastic Net Regularization

Combines the effects of both the Ridge and Lasso penalty constraints in the loss function given by:

$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \left[ \frac{(1 - \alpha)}{2} \beta_j^2 + \alpha |\beta_j| \right]$$



If Elastic Parameter  $\alpha = 1$ , the loss function becomes L1 Regularization (Lasso) and if  $\alpha = 0$ , the loss function becomes L2 Regularization (Ridge). When  $\alpha$  is between 0~1, the loss function implements a mix of both L1 (Lasso) and L2 (Ridge) constraints on the coefficients.

With higher lambda, the loss function penalizes the coefficients except the intercept. As a result, with really large lambda in linear regression, the coefficients are all zero, and the intercept is the average of the response. Logistic regression has a similar property, but the intercept is understood as prior-probability.

In general, use regularization to avoid overfitting, so multiple models with different lambda should be trained, and the model with smallest testing error should be chosen. For example, try lambda with values from [0, 0.1, 0.2, 0.3, 0.4, ... 1.0] and examine how the values affect the model.

## Linear Regression Use Case (1)

The following use case analyzes a United Nations dataset with sample Education, Literacy, Life Expectancy and GDP data.



### United Nations dataset on Country Education, Literacy, Life Expectancy, and GDP relationships

The dataset obtained from <http://socserv.socsci.mcmaster.ca/jfox/Books/Applied-Regression-2E/datasets/index.html>.

**Note:** Right-click the DB Table to see the data explorer options. See [Exploration Operators](#) section for more details.

## Datasets

The exact dataset used is [UN.csv](#) (10KB).

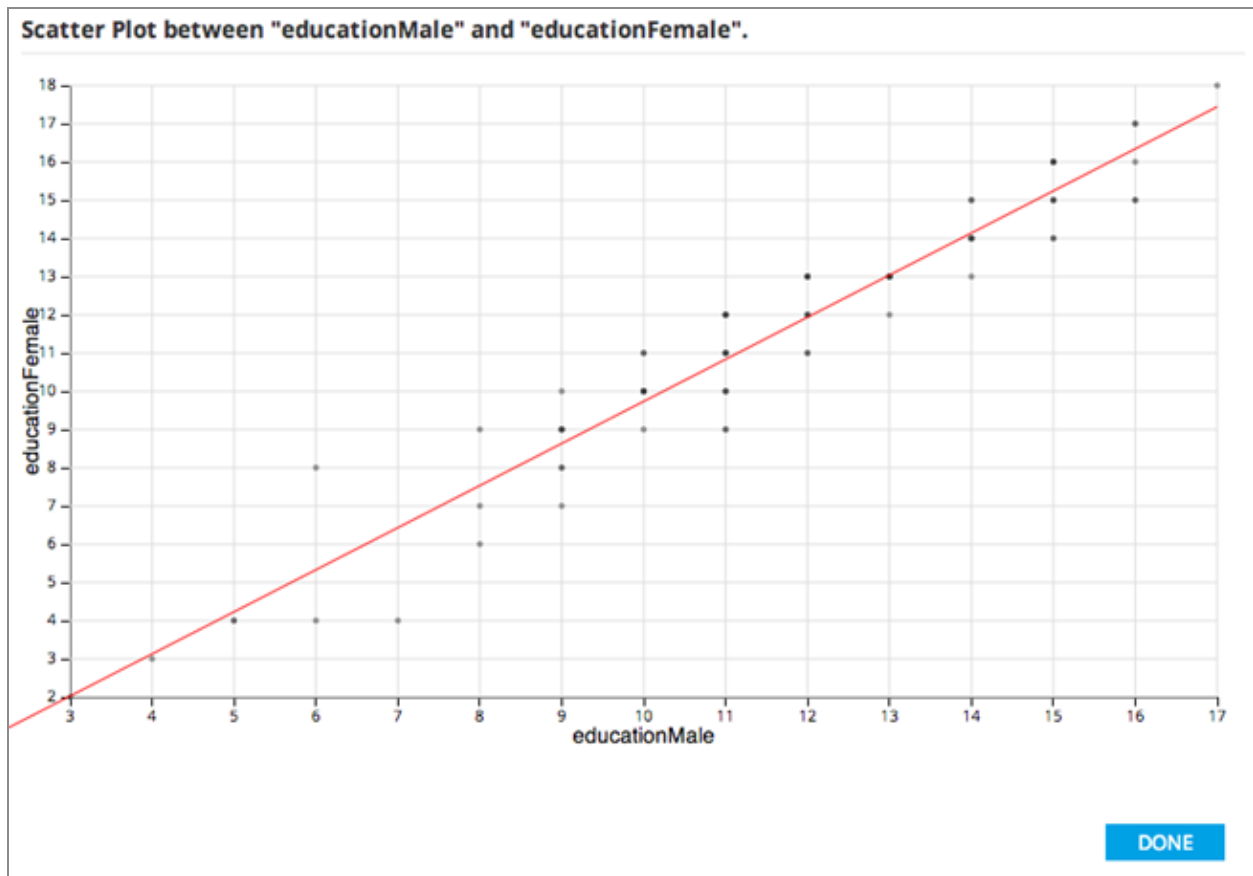
The collected United Nations dataset can be imported into TIBCO Data Science - Team Studio and quickly viewed using the Preview window. The first few rows are shown below:

united_nations													
countries	region	cfr	contraception	educationMale	educationFemale	lifeMale	lifeFemale	infantMortality	GDPperCapita	economicActivity	economicActivity	literacyMale	literacyFemale
Bahamas	America	2	62	12	13	71	77	14	12,545	81	67	2	2
Belize	America	4	47	11	10	73	76	30	2,569	79	34	21	23
Botswana	Africa	4	33	11	11	49	52	56	3,640	75	42	20	40
Burundi	Africa	6	9	5	4	46	49	114	205	90	91	51	78
Cape Verde	Africa	4	N/A	9	9	66	68	41	994	85	41	19	36
Costa Rica	America	3	75	N/A	N/A	75	79	12	2,696	79	33	5	5
Croatia	Europe	2	N/A	12	12	68	77	10	4,014	64	44	1	4
French Guiana	America	N/A	N/A	N/A	N/A	N/A	N/A	22	23,530	75	52	13	15
Germany	Europe	1	75	15	15	73	80	6	29,632	68	47	N/A	N/A
Greece	Europe	1	N/A	14	14	76	81	8	8,684	65	36	2	5
Haiti	America	5	18	N/A	N/A	53	56	82	386	80	49	52	58
Hong Kong	Asia	1	86	13	13	76	82	5	22,898	76	48	4	12
Ivory Coast	Africa	5	11	N/A	N/A	50	52	86	736	89	42	50	70
Korea Dem. Peoples Rep.	Asia	2	62	N/A	N/A	69	75	22	271	N/A	N/A	N/A	N/A
Lebanon	Asia	3	N/A	N/A	N/A	68	72	29	3,114	N/A	N/A	5	10

## Workflow

Prior to selecting and running a Linear Regression Operator, various Data Explorer Operators<sup>2</sup>, such as the Scatter Plot Matrix and Correlation Analysis plots, can be used to quickly assess whether there appears to be any sort of linear relationship between the dependent and independent variables. The following shows part of the Scatter Plot Matrix for some of the UN data:





A Scatter Plot Graph is displayed in the matrix for every pair of selected variables.

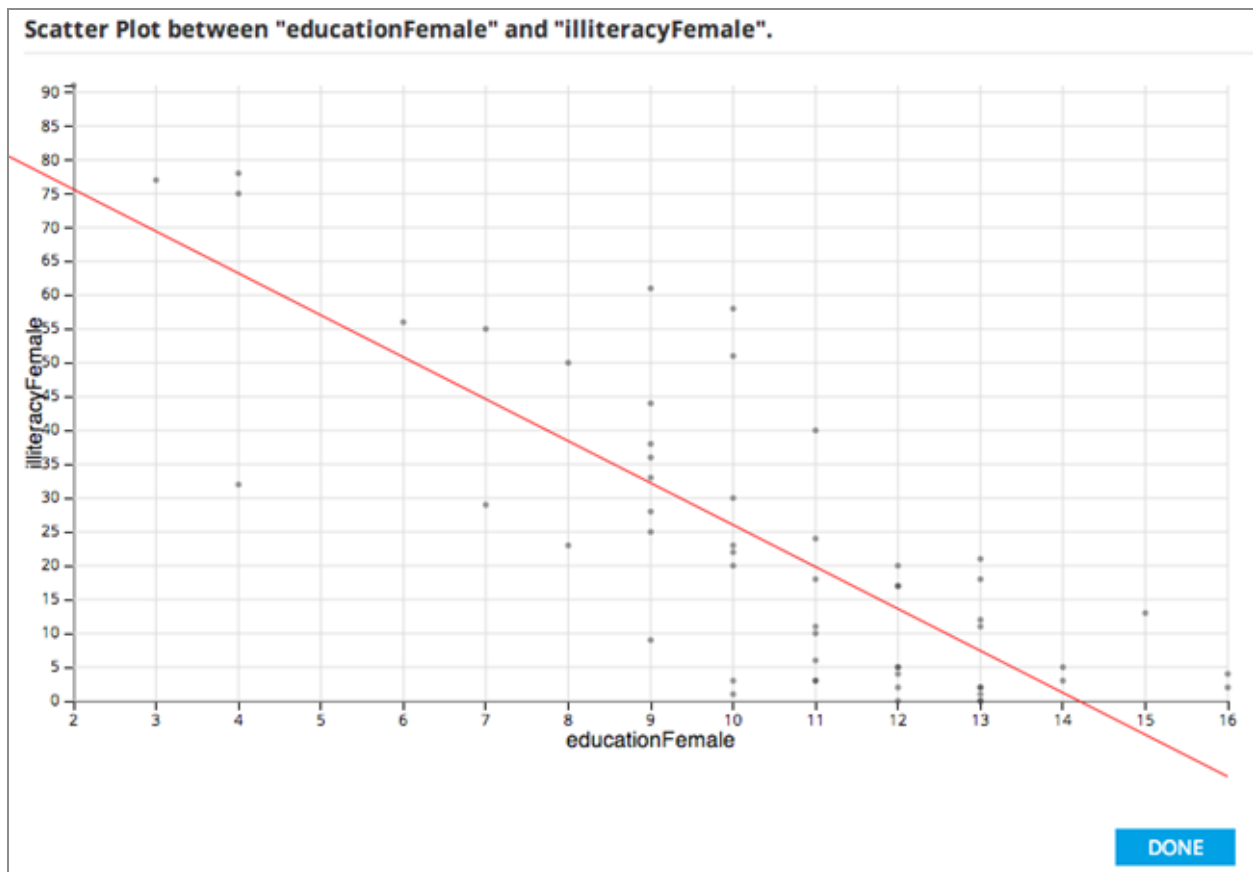
In this example, several evident linear relationships are displayed, such as a strong linear relationship between the *educationMale* and *educationFemale* variables.

This can be investigated further by clicking on the particular Scatter Plot Graph to view one variable (*educationFemale*) as the Y and the other (*educationMale*) as the X-axis.

Note that as the education of males goes up, so does the education of females. Looking at this data shows that the number of years a country has female education starts with a 1.3 year deficit with respect to male education, but it grows with a slope above 1 (meaning that countries that educate men longer have less of a deficit in educating women.)

However, these two variables are so perfectly correlated it is suspicious - perhaps they are collinear variables instead? Collinearity is a linear relationship between two explanatory variables in a model. Two variables are perfectly collinear if there is an exact linear relationship between the two. So *educationMale* and *educationFemale* are almost collinear and the modeler might likely remove one of them from the model.

Looking at the linear relationship between *educationFemale* and *illiteracyFemale* shows the expected result that as education of the female decreases, the illiteracy rate for the females in the country increases.



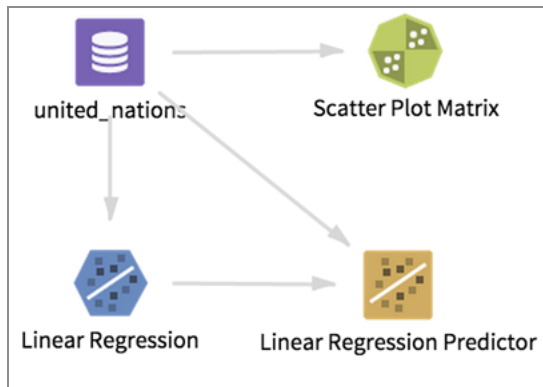
These results might trigger the idea to create a model that predicts a country's female illiteracy rate based on its female education levels.

One interesting model from this data might be life expectancy as a general indicator of a country's well-being. The predictors might be variables like education, contraception, infant mortality rates, and illiteracy levels. Since the male and female variables in the data are so strongly collinear, the modeler would want to eliminate the redundancy. The *educationFemale* variable would likely be picked since it indicates both education levels and the emancipation of women and, therefore, has more information contained in its value.

The step of eliminating redundant, or collinear, independent variables is a very important one in linear regression modeling.

Within TIBCO Data Science - Team Studio, the modeler can use the Linear Regression Operator to create a model using the United Nations data and a Linear Regression

Operator in order to predict the years of *lifeFemale* as the dependent variable and *educationFemale*, *contraception*, *infantMortality*, and *illiteracyFemale* as the independent variables, as follows:



## Results

The Summary results tab shows the overall Linear Regression equation, coefficient values and variables along with an R2 and Standard Error values for the model.

Note: An R2 value of 0.8915 shows a very high predictive capability (89% predictability). The Standard Error is +/- 3.67 years from the real female life expectancy, which seems like a reasonable error amount.

RESULTS - Linear Regression	
<b>Summary</b> Data Residual Plot Q-Q Plot	$\text{lifeFemale} =$ $0.0144 * \text{contraception}$ $+ 0.0141 * \text{educationFemale}$ $- 0.2733 * \text{infantMortality}$ $- 0.0121 * \text{illiteracyFemale}$ $+ 80.1815$ <b>R2: 0.8915</b> <b>Standard Error: 3.6732</b>

The Data Results tab provides the specific  $\beta$  Correlation Coefficients for each of the independent variables along with the related SE, T and P-Value statistics. When analyzing this data, the modeler is looking for independent variables that have both a low P-value ("confidence" level) and a high  $\beta$  Coefficient ("strength" level).

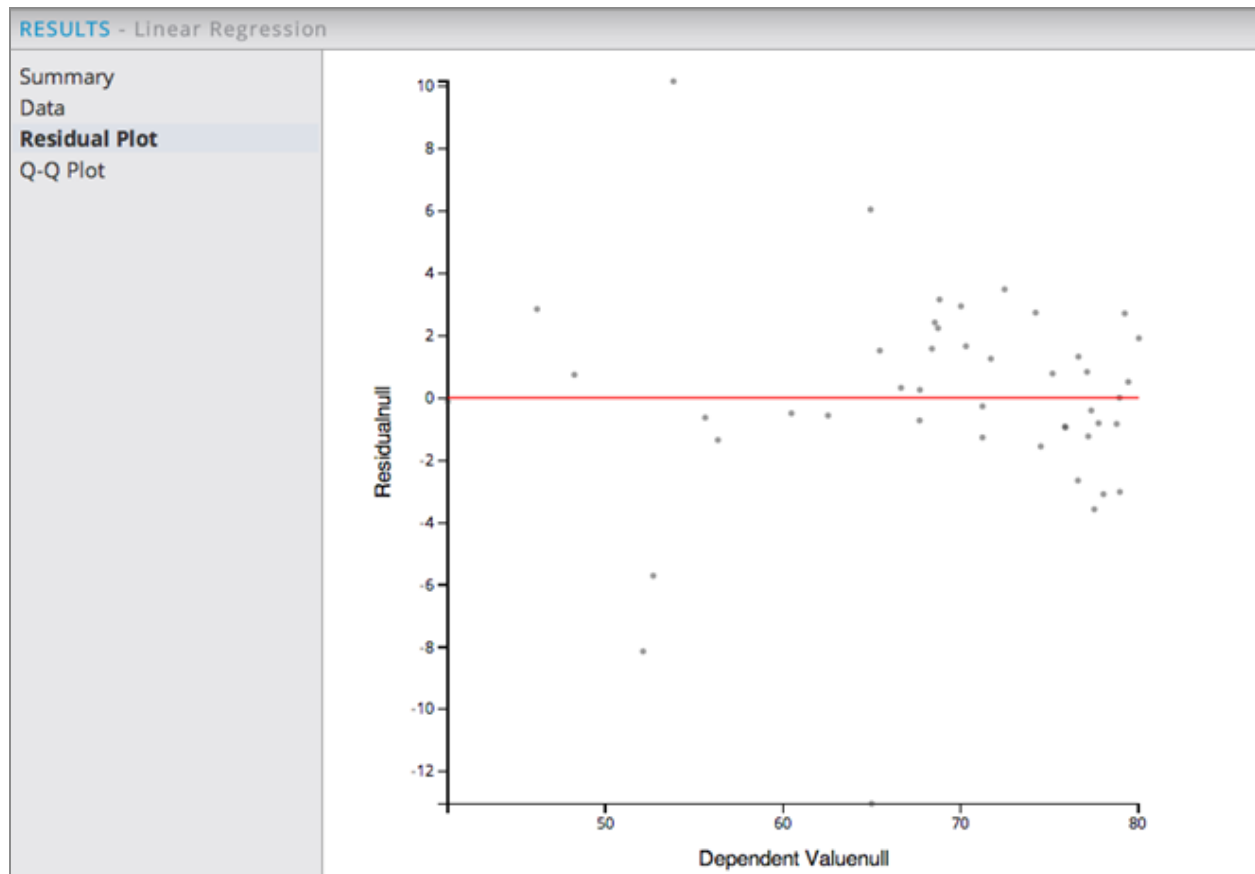
RESULTS - Linear Regression					
Summary	Attribute	Coefficient	SE	T-statistics	P-value
<b>Data</b>	intercept	80.1815	4.4453	18.0375	1.11e-16
Residual Plot	contraception	0.0144	0.0429	0.3355	0.739
Q-Q Plot	educationFemale	0.0141	0.3364	0.0419	0.9668
	infantMortality	-0.2733	0.031	-8.8053	6.628e-11
	illiteracyFemale	-0.0121	0.0443	-0.2736	0.7858

The data can also be sorted by  $\beta$  Coefficient values in order to understand which of the independent variables with low P-value has the greatest strength or correlation effect on the dependent variable. A high  $\beta$  indicates a change in the value of the independent variable has a greater effect on the change in the value of the dependent variable. In this example, *educationFemale* has the strongest correlation effect on female life expectancy.

Sorting the data by the P-value is usually the best way to analyze this data to understand the next modeling steps required. The variables with the lowest P-values are the ones the modeler should have the highest confidence of being true predictors in the model. Variables with P-values over 0.05 might be considered as optional for including in the model as there is a greater chance they might not be true predictors (that is, *illiteracyFemale* might be removed from the model next). In this example, *infantMortality* has a very low P-value, which makes intuitive sense for it to have an inverse correlation with female life expectancy levels.

RESULTS - Linear Regression					
Summary	Attribute	Coefficient	SE	T-statistics	P-value ▲
<b>Data</b>	intercept	80.1815	4.4453	18.0375	1.11e-16
Residual Plot	infantMortality	-0.2733	0.031	-8.8053	6.628e-11
Q-Q Plot	contraception	0.0144	0.0429	0.3355	0.739
	illiteracyFemale	-0.0121	0.0443	-0.2736	0.7858
	educationFemale	0.0141	0.3364	0.0419	0.9668

Looking at the Residual Plot provides a way to analyze the overall fit of a Linear Model on the data.

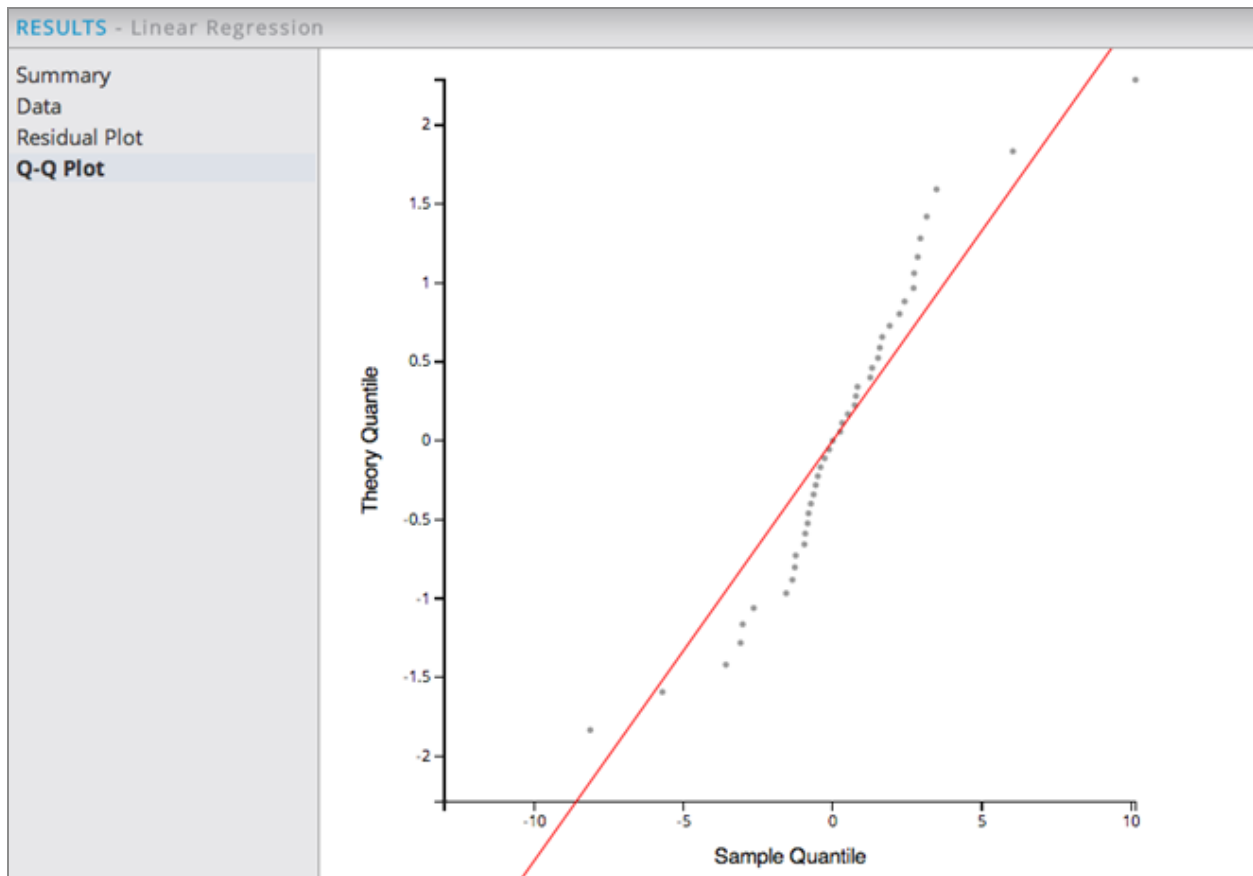


The example Residual Plot above shows a somewhat random distribution of the residuals, with most of the observed data points being between ages 65-80.

The Residual Plot is a graph of (actual Y - predicted Y) on the vertical axis and predicted Y on the horizontal axis. For example, for  $x=66$  (life expectancy age predicted to be 66), the residual is only 1 year older. This indicates the linear regression model does a pretty good job of predicting observed life expectancy rates.

Looking at the Q-Q Plot is an additional check that can be done in order to see if a linear model is appropriate for the data.





When looking at the Q-Q Plot, the distribution of the Residuals should match the theoretical (normal) distribution represented by the straight line. The points are expected to fit along the line  $y=x$ . This example above indicates a pretty good linear fit of the data near and above the average lifetime and less reliable of a prediction out of that range. However, since QQ plots show whether the distribution of the sample data on the X-axis matches the expected normal distribution of the linear model forecast on the Y-axis, they are not as critical to having a good model as a high R2 value.

When there are outliers or if the line is curved at the beginning or the end as in this example, the modeler may decide to exclude the outliers, or maybe add non-linear terms or interactions to improve the model.

As a final step, the Linear Regression Operator is linked to the Prediction Operator, the predicted  $P(\text{lifeFemale})$  values can be seen as well, as follows:

RESULTS - Linear Regression Predictor														
countries	region	tfr	contraception	educationMal	educationFem	lifeMale	lifeFemale	infantMortality	GDPperCapita	economicActi	economicActi	illiteracyMale	illiteracyFem	P(lifeFemale)
Croatia	Europe	2	N/A	12	12	68	77	10	4014	64	44	1	4	N/A
Lebanon	Asia	3	N/A	N/A	N/A	68	72	29	3114	N/A	N/A	5	10	N/A
Qatar	Asia	4	32	11	12	70	75	17	14013	93	28	21	20	75.9215
United.Arab.Emirates	Asia	3	N/A	10	10	74	77	15	17690	93	24	21	20	N/A
Grenada	America	N/A	54	N/A	N/A	N/A	N/A	14	2485	69	56	N/A	N/A	N/A
Puerto.Rico	America	2	64	N/A	N/A	73	81	9	12213	64	35	8	10	N/A
Gabon	Africa	5	N/A	N/A	N/A	54	57	85	5007	N/A	N/A	26	47	N/A
Maldives	Asia	7	N/A	N/A	N/A	66	63	49	1079	77	20	7	7	N/A
Senegal	Africa	6	13	N/A	N/A	50	52	62	572	83	24	57	77	N/A

## Summary

The predicted values are within a useful range of the actual, observed values. Note: the Prediction operator only processes rows that have no missing values for the selected independent variables. So in this example, not all the data rows have a *P\_lifeFemale* prediction value.

In summary, when analyzing a Linear Regression model overall, look for a high R2 value coupled with a randomly distributed Residual Plot and a strong match of the Q-Q Plot. These three things, co-existing together, provide high confidence in the fit of the Linear Model.

## Linear Regression Use Case (2)

This use case overviews the analysis of the linear relationship between the compressive strength of concrete and varying amounts of its components, especially cement.



## Concrete Compressive Strength Analysis

## Datasets

The example data tracks the concrete's compressive strength (MPa) compared to its cement, slag, fly ash, water, super plasticizer, coarse aggregate, and fine aggregate amounts (kg in a m3 mixture) and age (in days).

The dataset comes from the UC Irvine site: <https://archive.ics.uci.edu/ml/machine-learning-databases/concrete/compressive/> and the exact version used for this case is [concrete\\_data.csv](#) (41KB). It contains 1030 observations. The first few rows are shown below:

	A	B	C	D	E	F	G	H	I
1	cement_component	blast_furnac	fly_ash_com	water_comp	superplastici	coarse_aggre	fine_aggrega	age_day	concrete_co
2	540	0	0	162	2.5	1040	676	28	79.99
3	540	0	0	162	2.5	1055	676	28	61.89
4	332.5	142.5	0	228	0	932	594	270	40.27
5	332.5	142.5	0	228	0	932	594	365	41.05
6	198.6	132.4	0	192	0	978.4	825.5	360	44.3
7	266	114	0	228	0	932	670	90	47.03
8	380	95	0	228	0	932	594	365	43.7
9	380	95	0	228	0	932	594	28	36.45
10	266	114	0	228	0	932	670	28	45.85
11	475	0	0	228	0	932	594	28	39.29
12	198.6	132.4	0	192	0	978.4	825.5	90	38.07
13	198.6	132.4	0	192	0	978.4	825.5	28	28.02
14	427.5	47.5	0	228	0	932	594	270	43.01
15	190	190	0	228	0	932	670	90	42.33
16	304	76	0	228	0	932	670	28	47.81
17	380	0	0	228	0	932	670	90	52.91
18	139.6	209.4	0	192	0	1047	806.9	90	39.36
19	342	38	0	228	0	932	670	365	56.14
20	380	95	0	228	0	932	594	90	40.56
21	475	0	0	228	0	932	594	180	42.62

## Workflow

After importing this CSV data into TIBCO Data Science - Team Studio, the modeler can begin by analyzing the correlation between the compressive strength of concrete and the amount of cement used in the mixture.

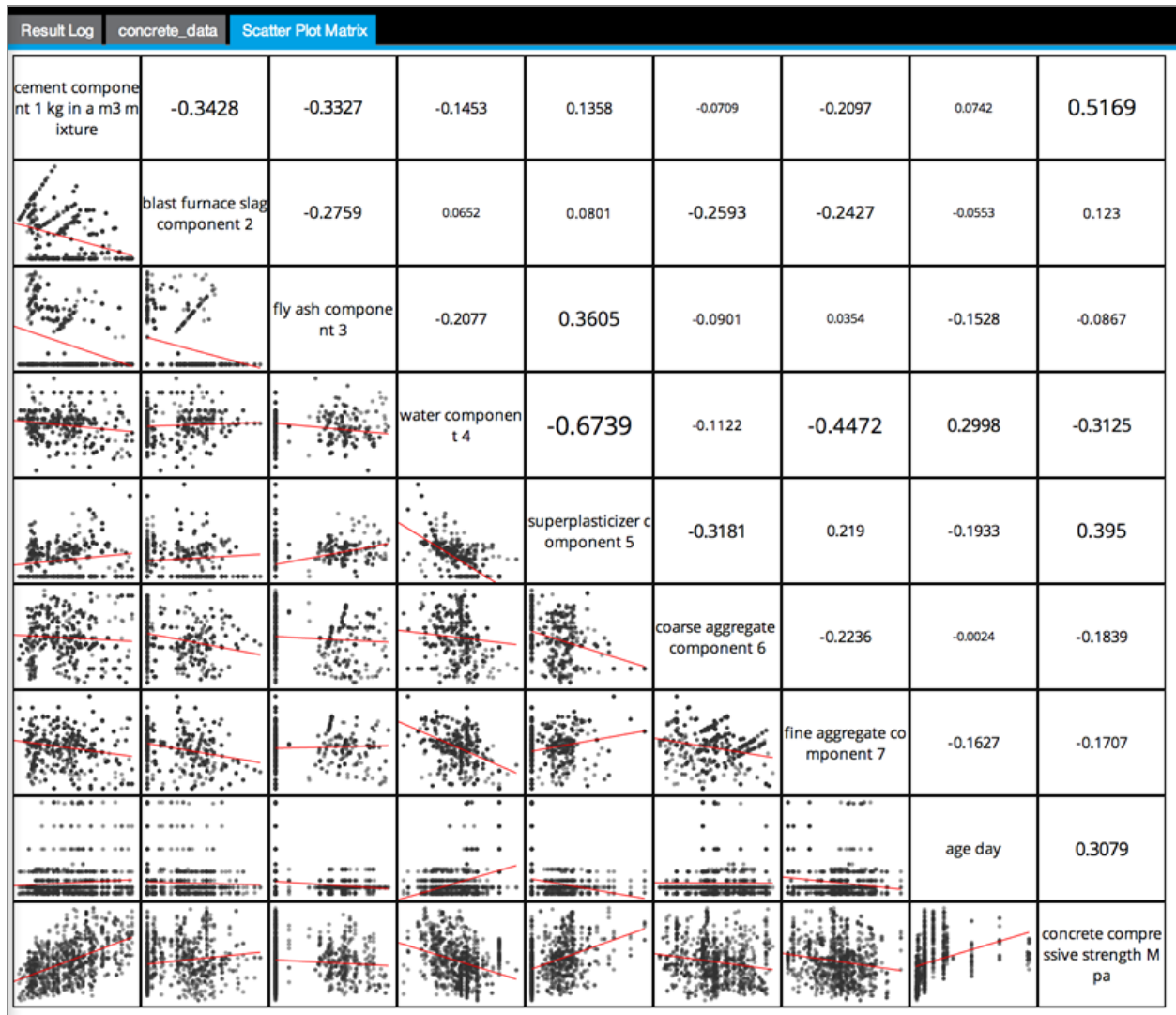
The modeler can create a new Data Flow with a DB Table element and then use the Preview option to view the data loaded within TIBCO Data Science - Team Studio, as shown below.

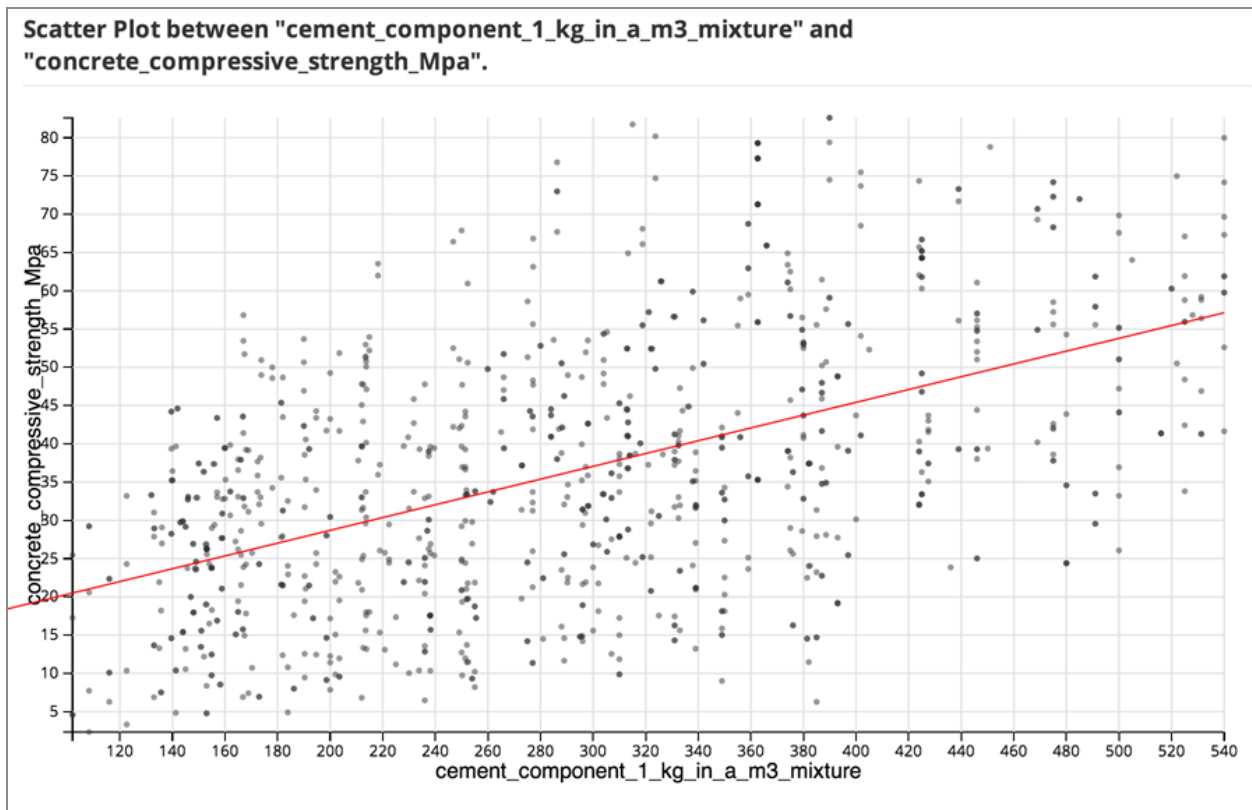
concrete_data								
cement_componen	blast_furnace_sla	fly_ash_componen	water_componen	superplasticizer_	coarse_aggregate	fine_aggregate_cc	age_day	concrete_compre
380	95	0	228	0	932	594	365	43.7
380	95	0	228	0	932	594	28	36.45
380	0	0	228	0	932	670	90	52.91
139.6	209.4	0	192	0	1,047	806.9	90	39.36
380	95	0	228	0	932	594	90	40.56
139.6	209.4	0	192	0	1,047	806.9	28	28.24
139.6	209.4	0	192	0	1,047	806.9	3	8.06
139.6	209.4	0	192	0	1,047	806.9	180	44.21
380	0	0	228	0	932	670	365	52.52
380	0	0	228	0	932	670	270	53.3
380	95	0	228	0	932	594	270	41.15
380	0	0	228	0	932	670	180	53.1
380	95	0	228	0	932	594	180	40.76
380	95	0	228	0	932	594	7	32.82
139.6	209.4	0	192	0	1,047	806.9	7	14.59
139.6	209.4	0	192	0	1,047	806.9	360	44.7
516	0	0	162	8.2	801	802	28	41.37
200	200	0	190	0	1,145	660	28	49.25
516	0	0	162	8.3	801	802	28	41.37
250	180	95	159	9.5	860	800	28	67.87
387	20	94	157	14.3	938	845	28	50.24

The data can be analyzed for linear dependencies using a Scatter Plot Matrix Operator with all columns selected, as follows:



Looking at a ScatterPlotGraph of the Concrete Strength related to the CementAmt reveals a general linear trend, scattered randomly around the trendline. As the amount of cement increases, there is a general overall increase in the strength of the concrete. This is a good indication that a linear regression analysis is worthwhile.





The modeler can then connect the Linear Regression Operator to the DB Table and perform an initial linear regression analysis using only the Concrete Strength(MPa) column as the suspected Dependent Variable and the Cement Amount column as the Independent Variable.

## Results

The results for this example show the following:

The **Summary** results tab shows an R2 of only 0.27, indicating that there are likely other factors than just the amount of cement in the mixture that affects the strength of the concrete. However, it does not necessarily mean that a linear model is not a good fit - the rest of the results should be analyzed.

RESULTS - Linear Regression	
<b>Summary</b>	
Data	concrete_compressive_strength_Mpa = 0.0821 * cement_component_1_kg_in_a_m3_mixture + 12.2359
Residual Plot	
Q-Q Plot	R2: 0.2672 Standard Error: 14.6792

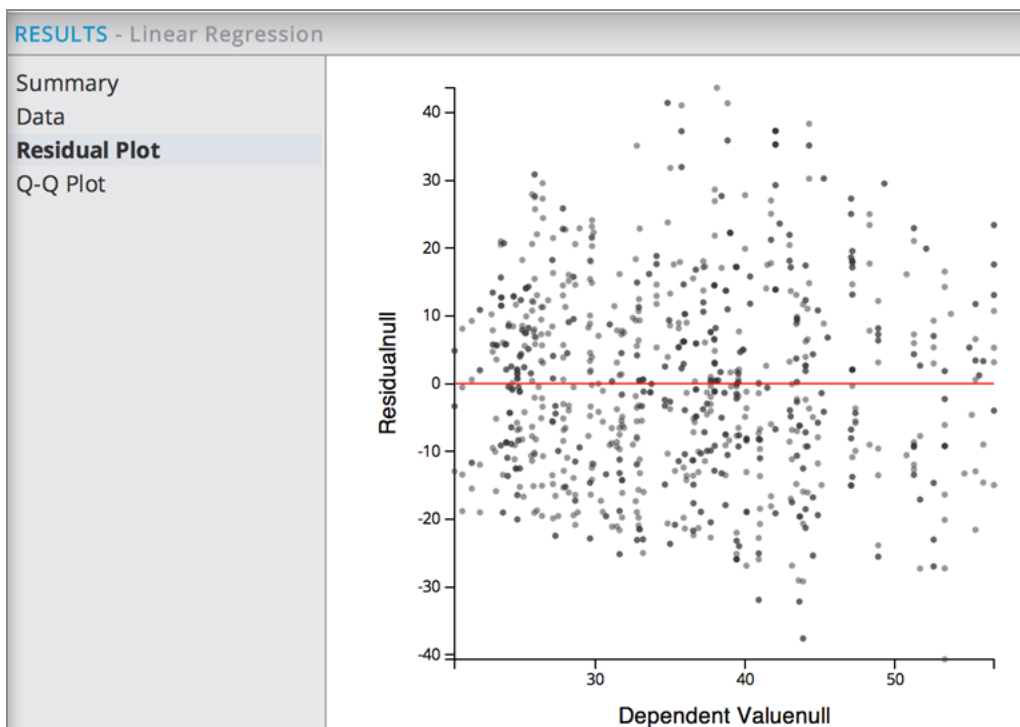
The **Data** results tab displays:

RESULTS - Linear Regression					
Summary	Attribute	Coefficient	SE	T-statistics	P-value
Data	Intercept	12.2359	1.0072	12.1488	8.882e-16
Residual Plot	cement_component_1_kg_in_a_m3_mixture	0.0821	0.0032	25.3529	0
Q-Q Plot					

This linear regression analysis shows a very low p-value, well under the 0.05 threshold. This indicates a high level of confidence that the amount of cement in the mixture linearly affects the compressive strength of the concrete.

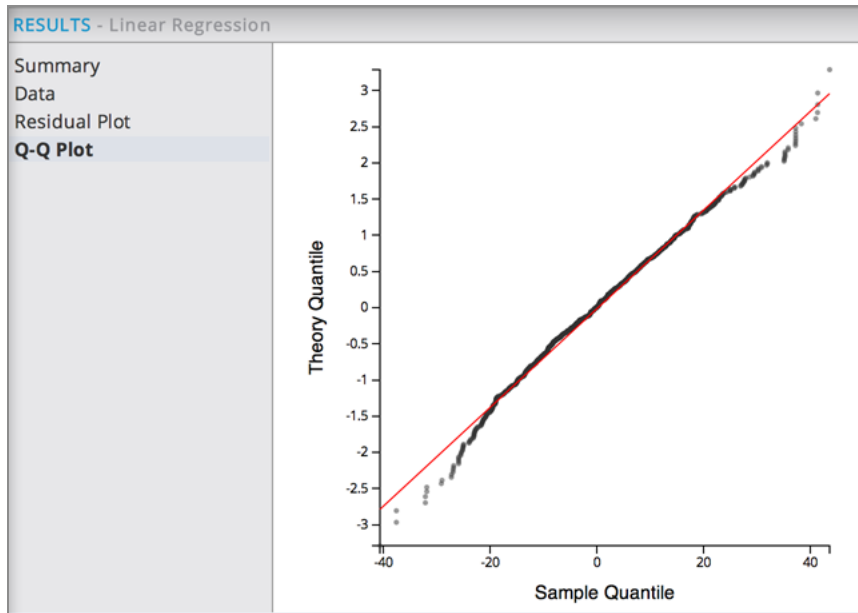
The strength of the linear relationship is represented by the Coefficient value of 0.0821. The bigger the Coefficient, the stronger the effect of Cement Amount on Concrete Strength. It represents the steepness of the Linear Regression line for forecasting Concrete Strength based on Cement column.

The **Residual Plot** displays the following graph:



This shows a random pattern of Concrete Strength residuals around the Concrete Strength fit value horizontal line, suggesting that a Linear Regression model is a good fit for the data. There are no structure anomalies seen.

The **Q-Q Plot** shows the following graph:



This also indicates a good fit for there being a Linear relationship between CementAmt and Concrete Strength - that is, the prediction errors are normally distributed as expected.

Overall, the Linear Regression model seems like a good start, but because the R2 for the model is low when only analyzing one independent variable, the modeler might next want to try to add all of the available variables for the next linear regression analysis. The results for all of the variables being included in the regression show the following:

The **Summary** results tab shows the following information:

RESULTS - Linear Regression	
Summary	concrete_compressive_strength_Mpa =
Data	0.1112 * cement_component_1_kg_in_a_m3_mixture
Residual Plot	+ 0.0927 * blast_furnace_slag_component_2
Q-Q Plot	+ 0.0729 * fly_ash_component_3
	- 0.1881 * water_component_4
	+ 0.2469 * superplasticizer_component_5
	+ 0.0053 * coarse_aggregate_component_6
	+ 0.0065 * fine_aggregate_component_7
	+ 0.1065 * age_day
	+ 10.9675
	R2: 0.6251
	Standard Error: 10.5205

The R2, or accuracy of the model, has improved to 0.62 and the Standard Error has decreased.

The **Data** results tab shows the following table:

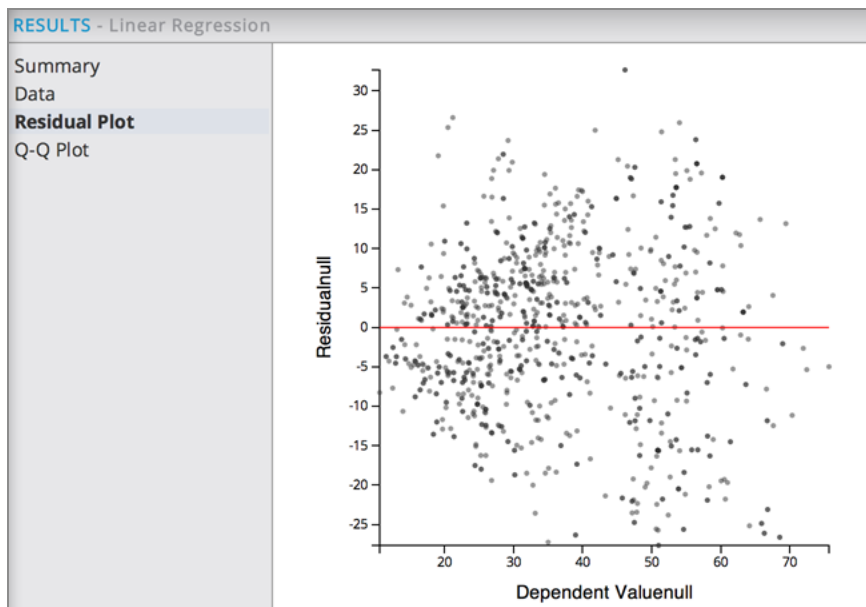


RESULTS - Linear Regression					
Summary	Attribute	Coefficient	SE	T-statistics	P-value
Data	Intercept	10.9675	20.7002	0.5298	0.5963
Residual Plot	cement_component_1_kg_in_a_m3_mixture	0.1112	0.0064	17.3223	1.443e-15
Q-Q Plot	blast_furnace_slag_component_2	0.0927	0.0076	12.1716	2.22e-16
	fly_ash_component_3	0.0729	0.0096	7.5864	5.318e-14
	water_component_4	-0.1881	0.0319	-5.896	4.457e-9
	superplasticizer_component_5	0.2469	0.0719	3.435	0.0006
	coarse_aggregate_component_6	0.0053	0.0073	0.7234	0.4696
	fine_aggregate_component_7	0.0065	0.0083	0.7897	0.4298
	age_day	0.1065	0.004	26.6558	2.22e-16

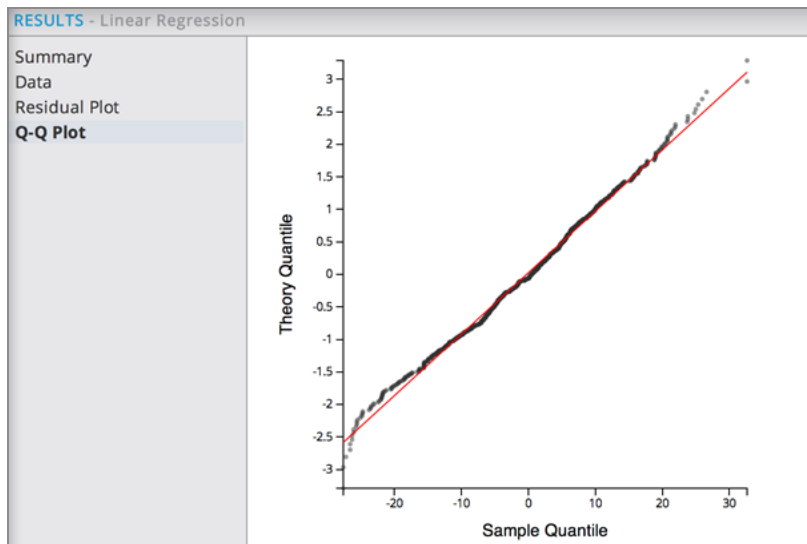
Sorting them by Coefficient value (clicking on the column header 'Coefficient') shows the Independent Variables, Superplasticizer, and Cement have the strongest linear correlation effect on the Concrete Strength.

Sorting them by P-value (clicking on the column header 'P-value') shows the Independent Variables, Age, Blast Furnace Slag, and Cement, have the highest confidence level in their significance in the model.

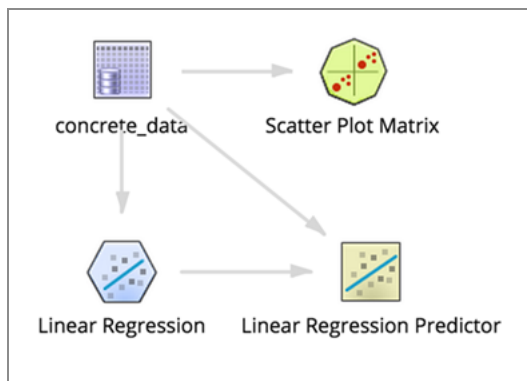
The **Residual Plot** again displays a random distribution of the Residuals around the Concrete Strength fit value horizontal line.



The **Q-Q Plot** again displays a normal distribution of the model's prediction errors.



To obtain the Linear Regression model's predictions for the Concrete Strength based on the component amounts in the mixture, the modeler should add in the Linear Regression Prediction Operator and link it to both the DB Table source and the Linear Regression Operator, as follows:



The results add an extra column to the data that provides the PredictedConcrete Strength value, P(MPa), in addition to the actual observed Concrete Strength value, MPa.

concrete_con	P(concrete_cc
43.01	57.5586
47.81	21.1976
41.84	47.9748
37.43	31.7888
55.26	57.0837
43.7	67.6749
41.54	38.3910
35.08	29.5526
49.19	27.7998
50.95	37.3836
54.38	46.9675
29	54.6234
40.2	66.3148
28.6	52.4062
45.7	55.0494

## Summary

The lesson here is that although the obtained  $R^2$  is below the general 0.8 rule of thumb, an  $R^2$  of 0.62 is certainly an improvement over the initial model fit and the results overall still provide a nice linear regression model with all of the residual plots looking good.

Note: Either the un-modeled portion of the  $R^2$  is a factor orthogonal to the known data and, if found, could improve the model quality or it is just un-correlated noise and would not help enhance the model.

The modeler has to decide if the predictions provided by the model are "good enough" for the business goals at hand.

## Probability Calculation Using Logistic Regression

Logistic Regression is the statistical fitting of an s-curve logistic or logit function to a dataset in order to calculate the probability of the occurrence of a specific categorical event based on the values of a set of independent variables.

Logistic Regression is an easily interpretable classification technique that gives the probability of an event occurring, not just the predicted classification. It also provides a measure of the significance of the effect of each individual input variable, together with a measure of certainty of the variable's effect. An example use case is determining the probability of loan default for an individual based on personal financial data.

TIBCO Data Science - Team Studio supports the following two common forms of Logistic Regression:

- The most common and widely used form, Binomial Logistic Regression, is used to predict a single category or binary decision, such as "Yes" or "No." A classic use case is determining the probability of loan default for an individual based on personal financial data. Specifically, Binomial Logistic Regression is the statistical fitting of an s-curve logistic or logit function to a dataset in order to calculate the probability of the occurrence of a specific event, or Value to Predict, based on the values of a set of independent variables.
- The more general form is Multinomial Logistic Regression (MLOR)\* which handles the case in which there are multiple categories to predict, not just two. It handles categorical data and predicts the probabilities of the different possible outcomes of a categorically distributed dependent variable. Example use cases might include weather predictions (sunny, cloudy, rain, snow), election predictions, or medical issue classification. Specifically, Multinomial Logistic Regression is the statistical fitting of a multinomial logit function to a dataset in order to calculate the probability of the occurrence of a multi-category dependent variable which allows two or more discrete outcomes.

## General Principles

Logistic regression analysis predicts the odds of an outcome of a categorical variable based on one or more predictor variables. A categorical variable is one that can take on a limited number of values, levels, or categories, such as "valid" or "invalid". A major advantage of Logistic Regression is its predictions are always between 0 and 1, unlike Linear Regression.

For example, a logistic model might predict the likelihood of a given person going to the beach as a function of temperature. A reasonable model might predict, for example, that a change in 10 degrees makes a person two times more or less likely to go to the beach. The term "twice as likely" for probability refers to the odds doubling (as opposed to the probability doubling). Rather, it is the **odds** that are doubling: from 2:1 odds, to 4:1 odds, to 8:1 odds, etc. Such a logistic model is called a log-odds model.

Hence, in statistics, Logistic Regression is sometimes called the logistic model or logit model. It is used for predicting the probability of the occurrence of a specific event by fitting data to a logit Logistic Function curve.

A Logistics Function is represented by an s-curve which was introduced by Pierre Verhulst in 1844, studied in relation to population growth. A [generalized logistic curve](#) can model the "S-shaped" behavior (abbreviated S-curve) of growth of some population  $P$ . The initial stage of growth is approximately [exponential](#); then, as saturation begins, the growth slows, and at maturity, growth stops. This simple logistic function may be defined by the formula

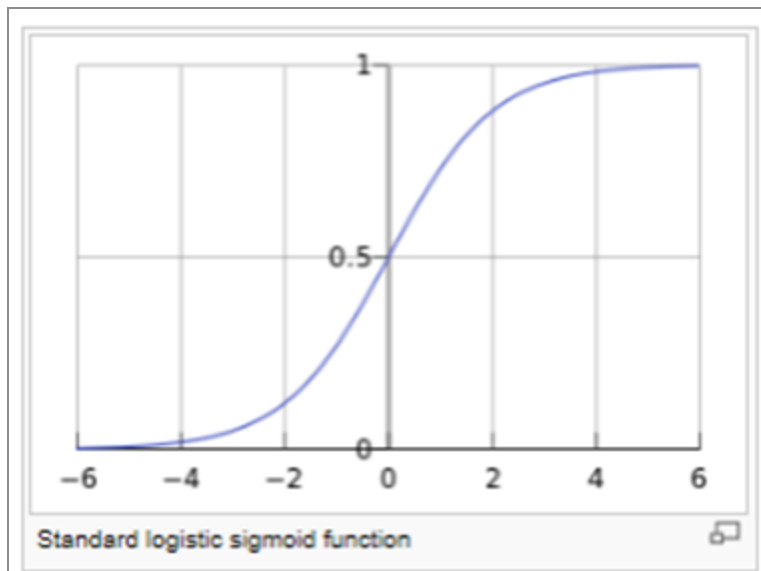
$$P(t) = \frac{1}{1 + e^{-t}}$$

where the variable  $P$  denotes a population,  $e$  is [Euler's number](#) (2.72), and the variable  $t$  might be thought of as time.

[http://en.wikipedia.org/wiki/Logistic\\_function#cite\\_note-0](http://en.wikipedia.org/wiki/Logistic_function#cite_note-0)

For values of  $X$  in the range of [real numbers](#) from  $-\infty$  to  $+\infty$ , the S-curve shown is obtained. In practice, due to the nature of the [exponential function](#)<sup>-t</sup>, it is sufficient to compute  $t$  over a small range of real numbers such as  $[-6, +6]$ .

The following is an example of an S-Curve or Logistic Function.



The Logistic Function can be applied to more generalized models that attempt to predict the probability of a specific event. In this case, there may be several factors or variables that contribute to whether the event happens. This Logistic Regression formula can be written generally in a linear equation form as:

$$\ln(P/(1 - P)) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots$$

Where  $P$  = Probability of Event, and are the regression coefficients and  $X_1, X_2, \dots$  are the independent variable values. Solving for the Probability equation results in:

$$\text{Probability of event} = P = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots)}}$$

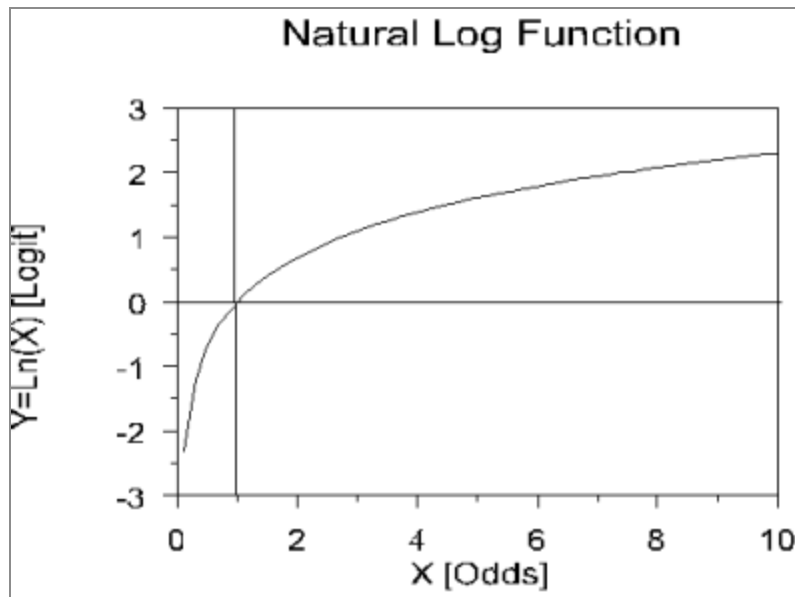
### Logistic Regression Odds Ratio

The odds of an event occurring are defined as the probability of a case divided by the probability of a non-case given the value of the independent variable. The odds ratio is the primary measure of effect size in logistic regression and is computed to compare the odds that membership in one group leads to a case outcome with the odds that membership in some other group leads to a case outcome. The odds ratio (denoted OR) is simply calculated by the odds of being a case for one group divided by the odds of being a case for another group. This calculates how much a change in the independent variable affects the value of the dependent.

As an example, suppose we only know a person's height and we want to predict whether that person is male or female. We can talk about the probability of being male or female, or we can talk about the odds of being male or female. Let's say that the probability of being male at a given height is .90. Then the odds of being male would be:

$$\text{odds} = \frac{P}{1 - P} = .9/.1 = 9 \text{ to } 1 \text{ odds}$$

Logistic Regression takes the natural logarithm of the odds (referred to as the logit or log-odds) to create a continuous criterion. The natural log function curve might look like the following.



The logit of success is then fit to the predictors using linear regression analysis. The results of the logit, however, are not intuitive, so the logit is converted back to the odds using the exponential function or the inverse of the natural logarithm. Therefore, although the observed variables in logistic regression are categorical, the predicted scores are actually modeled as a continuous variable (the logit).

### Binomial vs. Multinomial Regression

Logistic regression can be binomial or multinomial.

- Binomial or binary logistic regression refers to the instance in which the criterion can take on only two possible outcomes (e.g., "dead" vs. "alive", "success" vs. "failure", or "yes" vs. "no").
  - Note: The Alpine Logistics Regression Operator applies a binomial regression by assuming the dependent variable is either the Value to Predict or Not (Value to Predict).
- Multinomial Logistic Regression (MLOR) refers to the instance in which the criterion can take on three or more possible outcomes (for example, "better" vs. "no change" vs. "worse"). Generally, the criterion is coded as "0" and "1" in binary logistic regression as it leads to the most straightforward interpretation.

## Iteratively Reweighted Least Squares (IRLS)

The Alpine Logistic Regression Operator utilizes the method of Iteratively Reweighted Least Squares (IRLS) for calculating the best fitting, etc. Coefficient values. IRLS is a modeling fit optimization method that calculates quantities of statistical interest using weighted least squares calculations iteratively. The process starts off by finding the value of coefficients using the input, observed dataset with a standard least squares estimating approach, just as in Linear Regression modeling. It then takes the first estimate of coefficients and uses them to weight and recalculate the input data (using a mathematical weighting expression). This iterative, re-weighting of the input data continues until convergence is obtained, as defined by the Epsilon.

This IRLS method outputs both coefficients for each variable in the model as well as weights for each observation that help the modeler understand behavior of the algorithm during the iterations. These weights are useful diagnostics for identifying unusual data once convergence has been reached.

The Logistic Regression algorithm uses the Maximum Likelihood (ML) method for finding the smallest possible deviance between the observed and predicted values using calculus derivative calculations. After several iterations, it gets to the smallest possible deviance or best fit. Once it has found the best solution, it provides the final chi-square value for the deviance which is also referred to as the -2LL.

## Alternative Models

- [Classification Modeling with Naive Bayes](#)
- [Classification Modeling with Decision Tree](#)
- [Alpine Forest Classification](#)
- [Support Vector Machine Classification](#)

## Logistic Regression Use Case (1)

Binomial logistic regression is used extensively in the medical and social sciences fields, and in marketing applications that predict a customer's propensity to purchase a product.

### Credit Delinquency

The following model demonstrates using a logistic regression model to assess chances of credit delinquency. In this case, the Value to Predict might be defined as the probability of

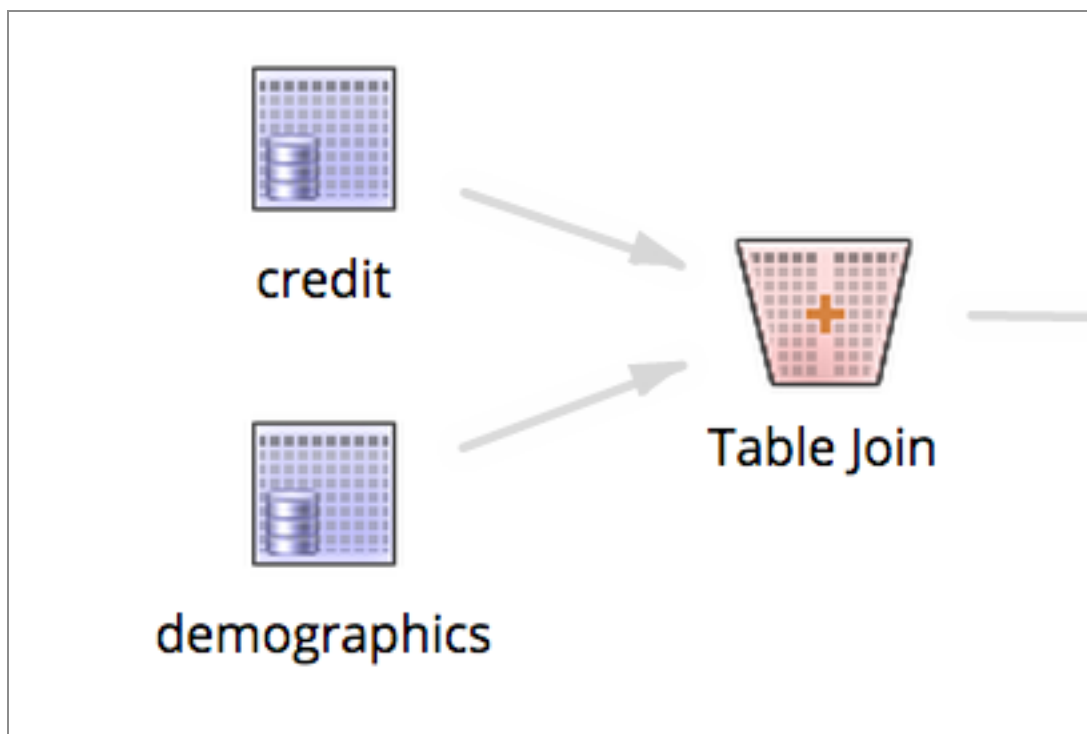


a "1" for an individual having a seriously delinquent credit card balance. A logistic regression could predict the chances of a Serious Delinquency value of 1 given the individual's revolving utilization, debt ratio, credit lines, monthly income, age, education, number of dependents, and number of times 30/90 days late.

## Datasets

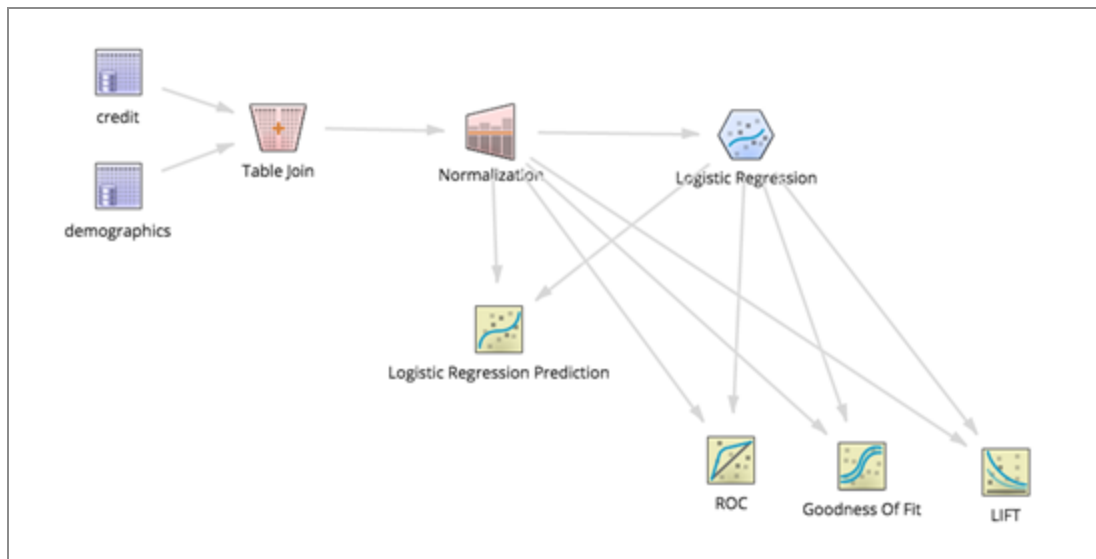
This example dataset was created by joining two source datasets: one with credit history ([credit.csv](#) (1.8MB)), and one with demographic profile data for each client ([demographics.csv](#) (1.4MB)).

The Table Join operator configuration and the resulting dataset are shown below.



## Workflow

Build a logistic regression model from the joined dataset by adding the Logistic Regression and Logistic Regression Prediction operators, as well as the ROC (Receiver Operating Characteristic), Confusion Matrix, Goodness of Fit, and LIFT scoring operators, as follows.



## Results

The **Summary** section displays the following.

RESULTS - Logistic Regression	
<b>Summary</b>	<b>Number of iterations: 6</b>
<b>Data</b>	<b>Residual Deviance: 14365.3747</b>
	<b>Null deviance: 18528.9167</b>
	<b>Chi-squared value: 4163.5420</b>
	<b>Fraction of variance explained: 0.2247</b>

The overall logistic regression model converged quite quickly with only six iterations that indicate there is correlation determined.

- The Residual Deviance is better than the Null Deviance, meaning that the model is more predictive than a random guess.
- The Chi-squared value is the difference between the two.

Always check the ratio of the Chi-Squared value to Null Deviance. This ratio is the portion of the Dependent Variable's variance explained by the logistic regression model: an

analogue of R2 statistic for linear regression. In this example, the percent of variance explained is 4163.5420/18528.9167, or 22.4%. (Not very high.)

The **Data** results section displays the following:

RESULTS - Logistic Regression							
Summary	Attribute	beta	Odds Ratio	SE	Z-value	P-value	Wald
<b>Data</b>	Bias (offset)	-4.5651	N/A	0.1152	-39.6152	2.424e-71	1,569.3649
	90dayslate	2.1257	8.3789	0.0573	37.1248	9.733e-69	1,378.2502
	revolve_util	-0.1245	0.883	0.1474	-0.8443	0.3985	0.7128
	debt_ratio	2.394	10.9568	0.0667	35.8985	2.149e-67	1,288.699
	credit_lines	-0.0393	0.9614	0.0257	-1.532	0.1255	2.3472
	income	0	1	0	-3.477	0.0005	12.0896
	30dayslate	-0.0098	0.9902	0.0477	-0.2058	0.8369	0.0424
	age	0.0014	1.0014	0.002	0.7327	0.4638	0.5368
	num_dep	-0.004	0.9961	0.0226	-0.1747	0.8614	0.0305
	edu	-0.0093	0.9908	0.0151	-0.6166	0.5375	0.3801

For analysis, sort by **P-value** by clicking the column header. This provides a quick way to assess the confidence of the variable being significant in the model. The smaller the P-value the better.

In this example, 90dayslate, debt\_ratio, and income all have P-values less than 5%. You can iterate on the model, removing the less significant parameters such as credit\_lines, revolve\_util, age, edu, 30dayslate, and num\_dep. Rerun the model and recheck the significance.

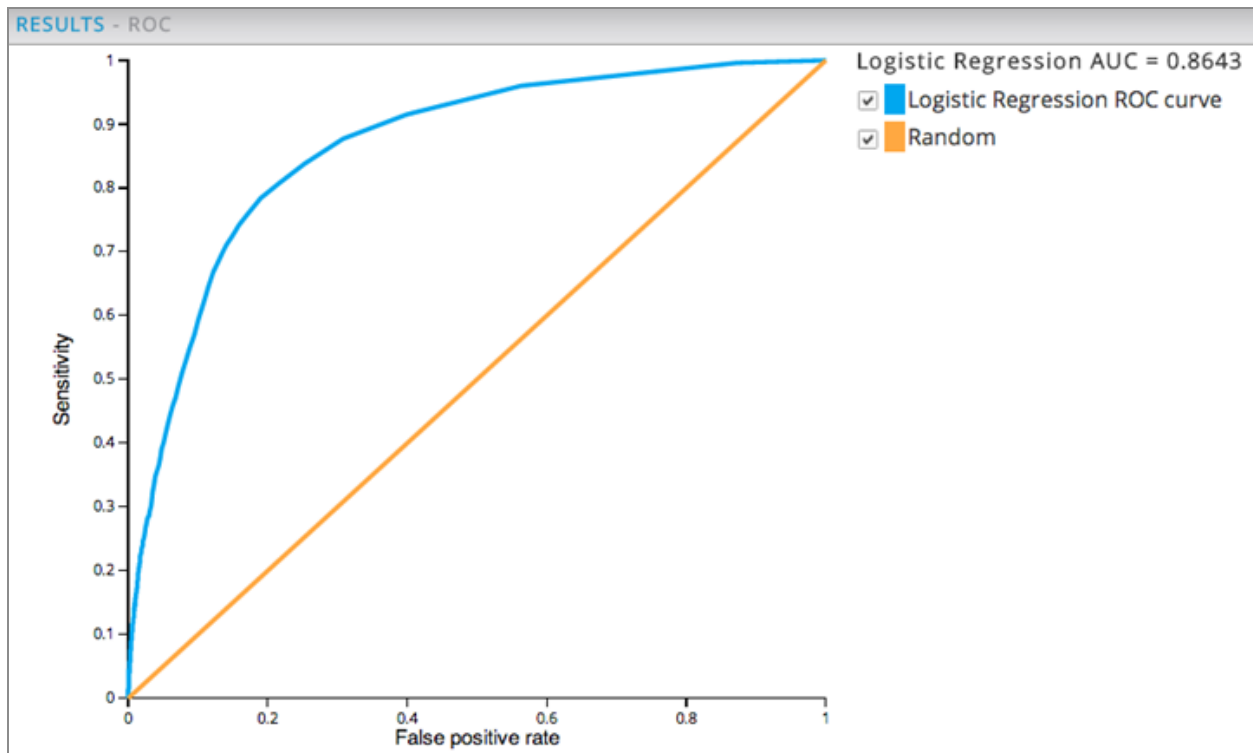
Including the Logistic Regression Prediction operator provides additional model results:

- The prediction value (0 or 1) uses a threshold assumption of less than 50% confidence that the prediction happens.
- The C(1) column indicates the confidence that the Dependent value is 1.
- The C(0) column indicates the confidence that the Dependent value is 0.
- The modeler can decide to ignore the prediction column and use a different confidence level to group the individuals likely to have credit delinquency, using a lower threshold such as 0.25, for example.

	P(delinquent)	C(1)	C(0)
	0	0.0129	0.9871
	0	0.0769	0.9231
	0	0.0576	0.9424
	0	0.0171	0.9829
	0	0.0195	0.9805
	0	0.0114	0.9886
	0	0.0316	0.9684
	0	0.2655	0.7345
	0	0.0208	0.9792
	0	0.0128	0.9872
	0	0.0123	0.9877
	0	0.0174	0.9826
	0	0.0356	0.9644

Hooking in additional logistic regression model diagnostic tools, such as the ROC graph, LIFT, and Goodness Of Fit operators, provides additional feedback on the quality of the model.

The ROC Curve plots the sensitivity (percent of correctly-classified results) versus false positives, as shown below.



The larger the AUC value, the more predictive the model is. Any AUC value over .80 is typically considered a "good" model. A value of .5 just means the model is no better than a "dumb" model that can guess the right answer half of the time.

## Logistic Regression Use Case (2)

Binomial logistic regression is used extensively in the medical and social sciences fields as well as in marketing applications that predict a customer's propensity to purchase a product.

### Predicting Malignant Breast Cancer Cells based on Cell Measurements

From a UCI dataset dataset obtained from UCI website at:

<https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/> of patient cells with and without breast cancer, it was noticed that various characteristics and measurements of cells can provide significant modeling capabilities for predicting whether the cells are benign or malignant.

## Datasets

The exact dataset version used for this case is [breast\\_cancer\\_data.csv](#) (122KB).

It contains 569 observations, a diagnosis column, and 30 attributes related to cell nucleus properties (texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry and fractal dimension). The mean, standard error, and worst (largest) of these features were computed, resulting in 30 predictive features.

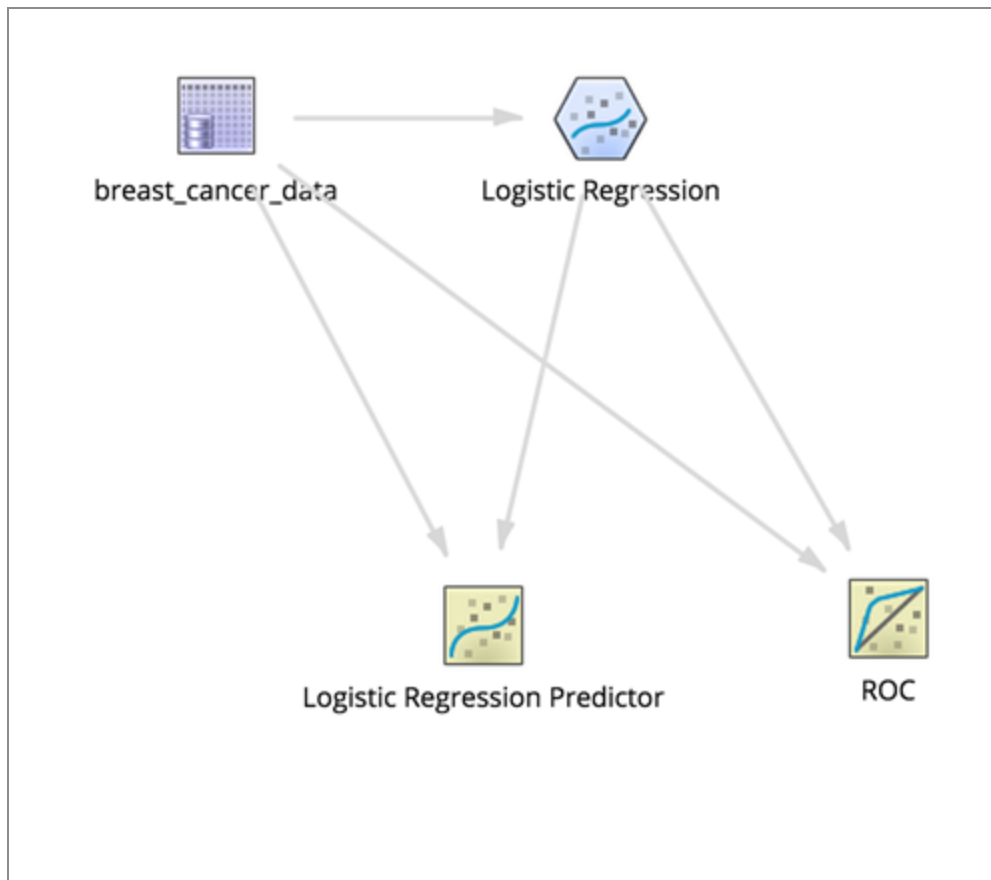
By loading the dataset into TIBCO Data Science - Team Studio, you can view the data using Data Explorer view (right-click the DB Table operator). The first few rows and columns are shown below:

breast_cancer_data														
id	diagnosis	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	mean_compactness	mean_concavity	mean_concave_points	mean_symmetry	mean_fractal_dimension	stddev_radius	stddev_texture	stddev_perimeter
851,509	M	21.16	23.04	137.2	1,404	0.0943	0.1022	0.1097	0.0863	0.1769	0.0528	0.6917	1.127	4.303
852,763	M	14.58	21.53	97.41	644.8	0.1054	0.1868	0.1425	0.0878	0.2252	0.0692	0.2545	0.9832	2.11
853,401	M	18.63	25.11	124.8	1,088	0.1064	0.1887	0.2319	0.1244	0.2183	0.062	0.8307	1.466	5.574
853,612	M	11.84	18.7	77.93	440.6	0.1109	0.1516	0.1218	0.0518	0.2301	0.078	0.4825	1.03	3.475
854,039	M	16.13	17.88	107	807.2	0.104	0.1559	0.1354	0.0775	0.1998	0.0652	0.334	0.6857	2.183
857,010	M	18.65	17.6	123.7	1,076	0.1099	0.1686	0.1974	0.1009	0.1907	0.0605	0.6289	0.6633	4.293
857,373	B	13.64	16.34	87.21	571.8	0.0768	0.0606	0.0186	0.0172	0.1353	0.0595	0.1872	0.9234	1.449
857,810	B	13.05	19.31	82.61	527.2	0.0806	0.0379	0.0007	0.0042	0.1819	0.055	0.404	1.214	2.595
859,283	M	14.78	23.94	97.4	668.3	0.1172	0.1479	0.1267	0.0903	0.1953	0.0665	0.3577	1.281	2.45
859,464	B	9.465	21.01	60.11	269.4	0.1044	0.0777	0.0217	0.015	0.1717	0.069	0.2351	2.011	1.66
859,983	M	13.8	15.79	90.43	584.1	0.1007	0.128	0.0779	0.0507	0.1662	0.0657	0.2787	0.6205	1.957
8,610,629	B	13.53	10.94	87.91	559.2	0.1291	0.1047	0.0688	0.0656	0.2403	0.0664	0.4101	1.014	2.652
8,610,637	M	18.05	16.15	120.2	1,006	0.1065	0.2146	0.1684	0.108	0.2152	0.0667	0.9806	0.5505	6.311
8,611,161	B	13.34	15.86	86.49	520	0.1078	0.1535	0.1169	0.0699	0.1942	0.069	0.286	1.016	1.535
86,135,502	M	19.02	24.59	122	1,076	0.0903	0.1206	0.1468	0.0827	0.1953	0.0563	0.5495	0.6636	3.055

## Workflow

The id is the patient ID and can be ignored. The diagnosis is the dependent variable to predict and can have a value of "M" for malignant or "B" for benign. The rest of the columns are numerical quantities obtained from a microscopic cell survey.

Because this model predicts a binary malignant or benign diagnosis, it makes sense to try the Logistic Regression operator.



Initially, the modeler might take the aggressive approach and select all the non-ID variables to be in the model.

The **Summary** results tab shows:

RESULTS - Logistic Regression	
<b>Summary</b>	<b>The algorithm did not converge</b>
<b>Data</b>	<b>Number of iterations: 25</b>
	<b>Residual Deviance: 0.0004</b>
	<b>Null deviance: 751.4400</b>
	<b>Chi-squared value: 751.4396</b>
	<b>Fraction of variance explained: 1.0000</b>

A number of bad diagnostic results are displayed.

The algorithm did not converge, as indicated by the Iteration amount being the maximum setting. The Residual deviance is essentially zero, which is suspicious because it makes the Chi square/ Null deviance indicate a near-perfect model fit (not likely).

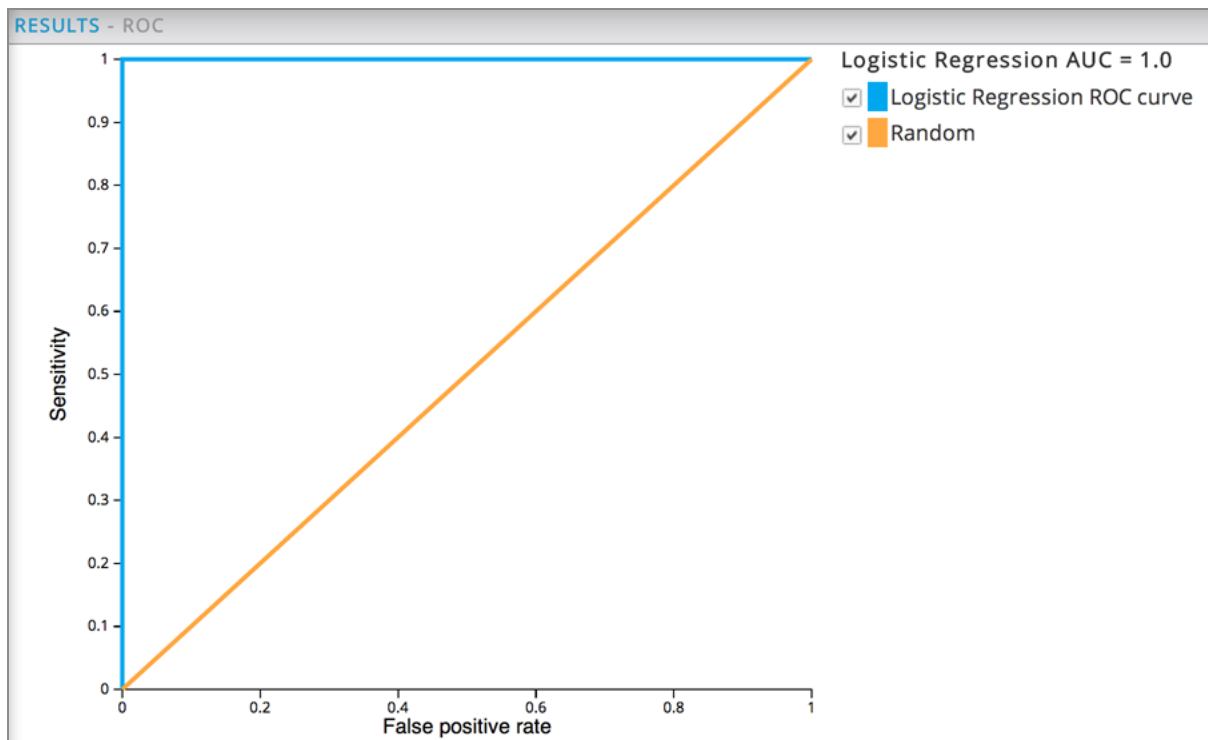
The **Data** results tab shows:

RESULTS - Logistic Regression							
Summary	Attribute	beta	Odds Ratio	SE	Z-value	P-value	Wald
<b>Data</b>	Bias (offset)	-203.0064	N/A	5,678,351.39	0	1	1.278e-9
	mean_radius	-2,159.3709	0	2,012,643.8184	-0.0011	0.9991	0
	mean_texture	39.349	1.228e+17	187,885.3648	0.0002	0.9998	0
	mean_perimeter	59.0245	4.305e+25	268,690.5448	0.0002	0.9998	0
	mean_area	16.7407	18,640,000	25,039.7324	0.0007	0.9995	4.5e-7
	mean_smoothness	17,645.5624	Infinity	27,560,000	0.0006	0.9995	4.1e-7
	mean_compactness	-20,355.9987	0	7,119,929.9737	-0.0029	0.9977	0
	mean_concavity	11,025.8677	Infinity	14,320,000	0.0008	0.9994	5.9e-7
	mean_concave_points	11,220.587	Infinity	4,161,294.8612	0.0027	0.9978	0
	mean_symmetry	-7,826.4775	0	7,448,116.9481	-0.0011	0.9992	0
	mean_fractal_dimension	25,069.9877	Infinity	12,140,000	0.0021	0.9984	0
	stddev_radius	1,016.2413	Infinity	5,916,767.7369	0.0002	0.9999	0
	stddev_texture	-200.258	0	2,396,305.8455	-0.0001	0.9999	6.984e-9
	stddev_perimeter	-412.6294	0	587,700.6666	-0.0007	0.9994	4.9e-7
	stddev_area	42.3035	2.356e+18	114,868.587	0.0004	0.9997	1.4e-7
	stddev_smoothness	-45,656.8411	0	209,100,000	-0.0002	0.9998	0
	stddev_compactness	35,651.4717	Infinity	36,590,000	0.001	0.9992	9.5e-7

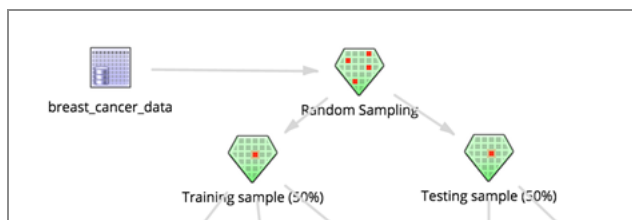
There are a number of large beta coefficient values, which is evidence the model was over fitting at least a subset of data. That would cause coefficients to run to infinity in logistic regression. In this case, the modeler would suspect all of the data is overfit because the model's deviance is zero. Overfitting means a very good fit on the training data that does not show the same utility on the new data.



The ROC curve has an AUC of 1.0, which is definitely an indication of overfitting.



If the modeler suspects overfitting, a way to diagnose this is to run a cross-validation or "hold out" training where some data is restricted from the model and used to simulate new data. Inserting a Random Sampling operator to split the data into two 50% groups of sample data would accomplish this, as follows:

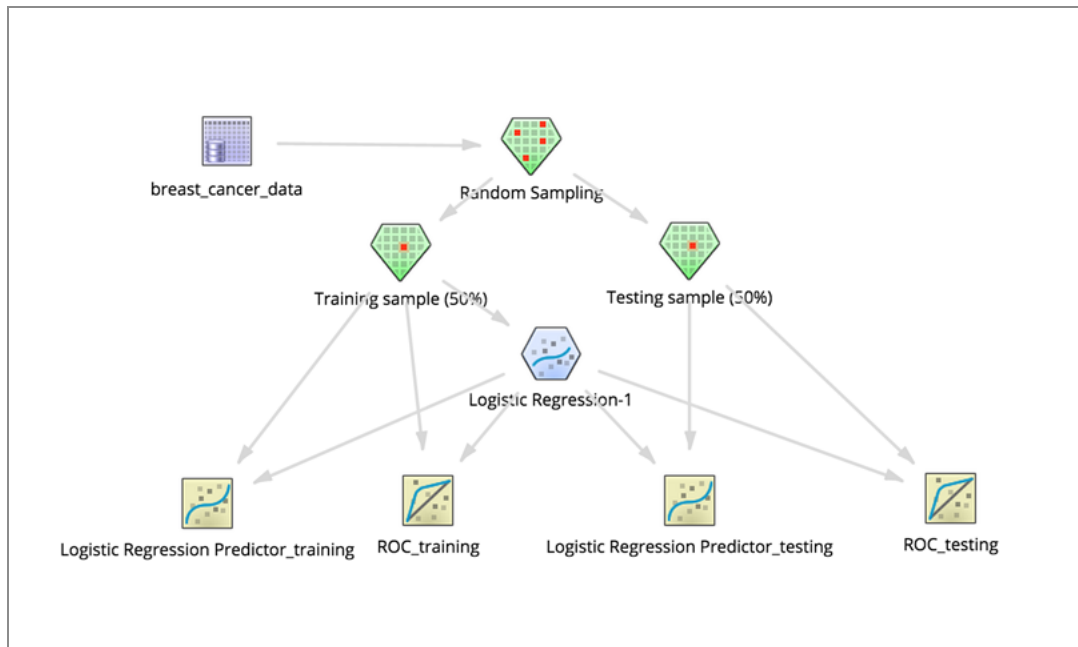


There should be two Sample Selector operators added to collect the split datasets.

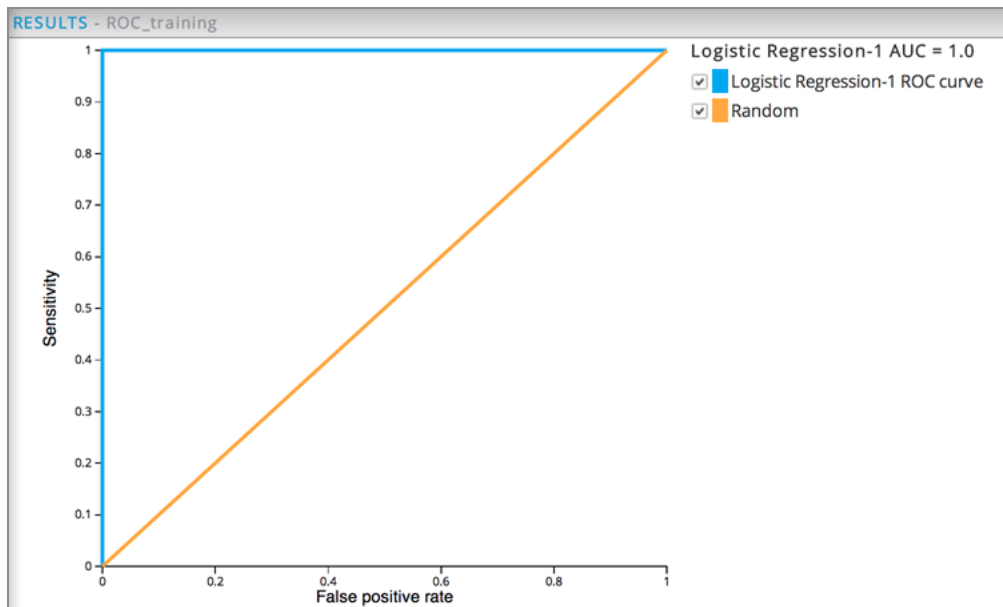
Then adding two Logistic Regression Prediction operators to predict on the data used to build the model and on the non-training data allows the modeler to confirm whether the model only does well predicting using the training dataset.

Also adding ROC operators shows the quality of the model on both datasets. Note: the ROC operators hook up directly with the source dataset and the Logistic Regression operator.

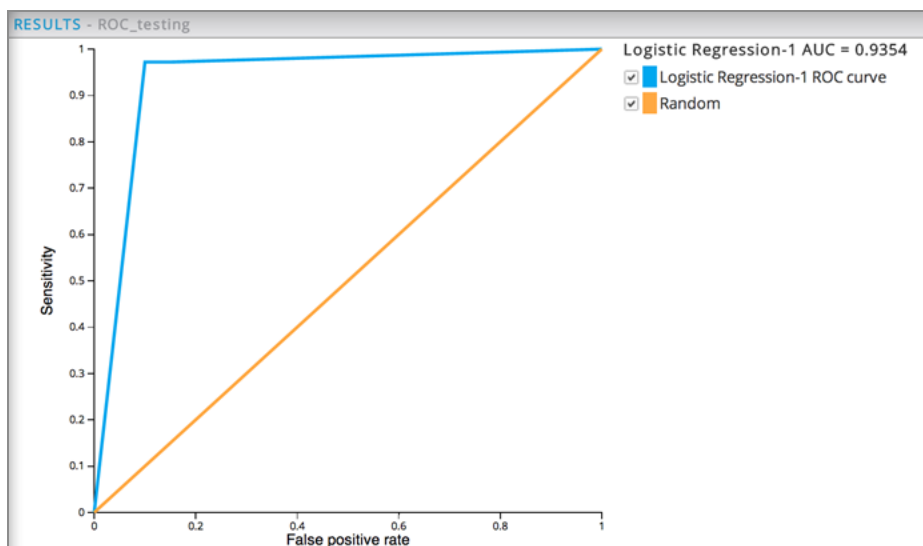
The final cross-validation setup might look as follows:



The resulting ROC graph for the training data shows a perfect model. The green not green in Illuminator line is always the diagonal and useless model. The red line shows the model performance. In the top right corner, it is 100% sensitive (that is, catches all the malignant cases) but has a high false positive rate (nearly always wrong). In the bottom left corner, it has zero false positives (never predicts malignant incorrectly) but has no sensitivity (no one is malignant). The top left corner shows pure model perfection with no false positives (everything predicted correctly) and complete sensitivity (everyone with malignancy was predicted).



The resulting ROC graph for the cross-validation data has the Area Under the Curve (AUC) of 0.9354 instead of 1. This is still excellent but is radically different from the training performance. For cross-validation purposes, the modeler wants the performance to be close - this much drop off is additional evidence of over fitting.



The ideal fix for over fitting in this case would be to obtain more training data in order to avoid the over-memorization of the data. However, the alternate, more realistic fix is to eliminate some of the variables, even though they might be useful in the model. With less training data, it is safer to have fewer training variables.

One approach for eliminating variables might be to turn on the stepwise regression methodology for the Logistic Regression operator, using stepwise mode using "AIC" criterion type.

In this example, the stepwise regression reduces the number of independent variables down to 7 from the original 30 variables. However, the model still has issues in this example with the odds ratio reaching Infinity for *mean\_concave\_points*, *stddev\_radius* and *worst\_symmetry*.

RESULTS - Logistic Regression-1							
Summary	Attribute	beta	Odds Ratio	SE	Z-value	P-value	Wald
Data	Bias (offset)	-4,132.2895	N/A	239,308.2883	-0.0173	0.9862	0.0003
	mean_area	-3.0347	0.0481	178.0579	-0.017	0.9864	0.0003
	mean_concave_points	20,352.8161	Infinity	1,232,714.952	0.0165	0.9868	0.0003
	stddev_radius	1,567.4613	Infinity	90,777.5599	0.0173	0.9862	0.0003
	stddev_fractal_dimension	-199,514.3263	0	12,510,000	-0.0159	0.9873	0.0003
	worst_texture	45.9558	9.085e+19	2,692.6163	0.0171	0.9864	0.0003
	worst_area	3.1741	23.9052	185.2723	0.0171	0.9863	0.0003
	worst_symmetry	4,791.1753	Infinity	279,870.0778	0.0171	0.9863	0.0003

As another approach, the modeler might decide to manually remove the standard deviation columns in the data to begin with and rerun the regression without stepwise turned on. This might avoid using variables that duplicate each other's meanings. For example, the modeler might deselect the standard deviation columns from the model. This results in a converged model with reasonable p-values and no run-away coefficient values.

RESULTS - Logistic Regression-1	
Summary	Number of iterations: 12
Data	Residual Deviance: 24.0876
	Null deviance: 366.8306
	Chi-squared value: 342.7430
	Fraction of variance explained: 0.9343

The Number of iteration times did not meet the maximum, which indicates the logistic regression converged appropriately.

The **Data** tab shows reasonable beta and Odds Ratio values.

Summary	Data					
Attribute	beta	Odds Ratio	SE	Z-value	P-value	Wald
Bias (offset)	-10.8259	N/A	51.5416	-0.21	0.8336	0.0441
mean_radius	-0.8066	0.4464	17.7888	-0.0453	0.9638	0.0021
mean_texture	0.3192	1.376	0.4415	0.723	0.4697	0.5227
mean_perimeter	0.4282	1.5345	2.7738	0.1544	0.8773	0.0238
mean_area	-0.043	0.9579	0.1011	-0.4258	0.6702	0.1813
mean_smoothness	-65.1169	0	157.6651	-0.413	0.6796	0.1706
mean_compactness	-377.7114	0	297.0703	-1.2715	0.2036	1.6166
mean_concavity	60.4305	1.757e+26	119.4588	0.5059	0.6129	0.2559
mean_concave_points	494.6076	6.388e+214	421.8872	1.1724	0.241	1.3744
mean_symmetry	22.8976	8.797e+9	62.7898	0.3647	0.7154	0.133
mean_fractal_dimension	323.4186	2.877e+140	578.585	0.559	0.5762	0.3125
worst_radius	-6.2789	0.0019	10.2193	-0.6144	0.5389	0.3775
worst_texture	0.2624	1.3001	0.3455	0.7596	0.4475	0.577
worst_perimeter	0.0574	1.0591	0.6572	0.0874	0.9304	0.0076
worst_area	0.0836	1.0872	0.088	0.9504	0.3419	0.9032
worst_smoothness	23.83	2.235e+10	85.7659	0.2778	0.7811	0.0772
worst_compactness	52.0442	4.004e+22	51.2414	1.0157	0.3098	1.0316
worst_concavity	-14.1953	0	34.8449	-0.4074	0.6837	0.166
worst_concave_points	19.1648	210,400,000	75.9137	0.2525	0.8007	0.0637
worst_symmetry	22.2819	4.752e+9	26.5212	0.8402	0.4008	0.7059
worst_fractal_dimension	-68.6557	0	193.8836	-0.3541	0.7233	0.1254

The modeler can experiment iteratively by sorting the remaining variables by p-value and removing the ones with the highest p-values (that is, least confident of their significance). Note: the variables should not be taken out all at once because a variable may be bad because of another variable duplicating it. So it should be done in steps either manually or automatically, using stepwise.

This might leave the model down to just 3-4 variables, which is less likely to overfit. In the following example, only the variables *mean.concavepoints*, *worst.area*, *mean.texture*, and *mean.radius* have been left in the model. The results are:

RESULTS - Logistic Regression	
Summary	Number of iterations: 10
Data	Residual Deviance: 84.6983
	Null deviance: 751.4400
	Chi-squared value: 666.7417
	Fraction of variance explained: 0.8873

RESULTS - Logistic Regression							
Summary	Attribute	beta	Odds Ratio	SE	Z-value	P-value	Wald
Data	Bias (offset)	-0.7118	N/A	3.8548	-0.1846	0.8535	0.0341
	mean_radius	-3.1109	0.0446	0.6882	-4.5203	0	20.433
	mean_texture	0.4264	1.5317	0.0906	4.7066	0	22.1521
	mean_concave_points	123.1718	3.111e+53	21.4921	5.731	0	32.8447
	worst_area	0.0377	1.0385	0.0072	5.2503	1.6e-7	27.5661

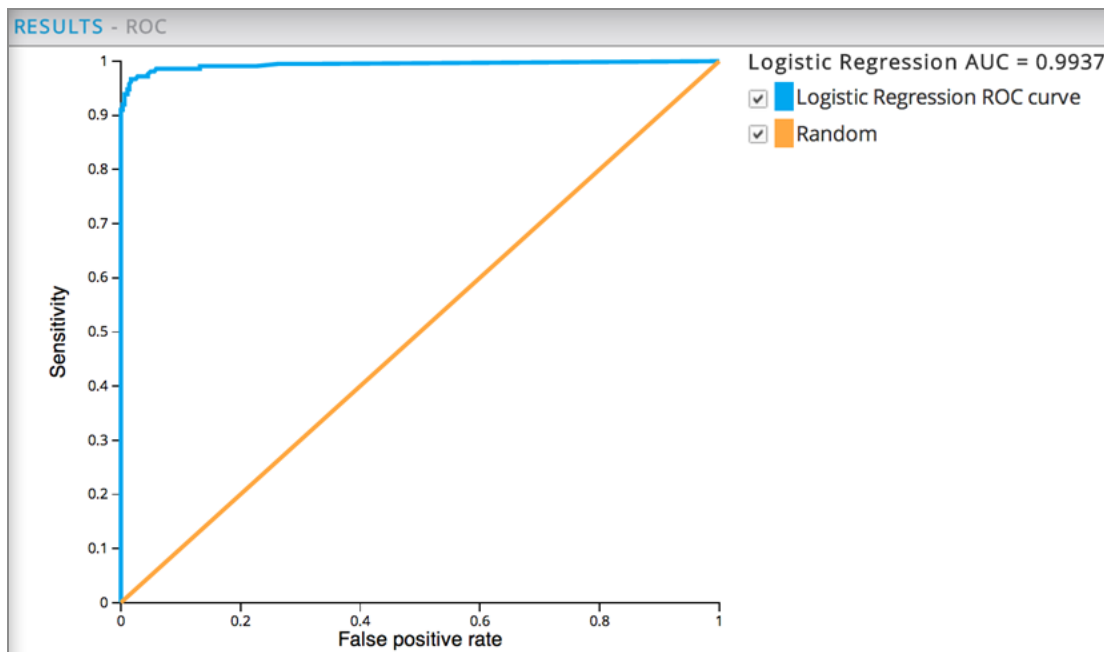
The p-values being very small give high confidence in their significance. Because the deviance of 84 is so low and the chi square/null deviance ratio is approximately .89, the modeler knows that just these 4 variables alone (*mean.concavepoints*, *mean.texture*, *mean.radius*, and *worst.area*) are predicting 89% of the diagnosis variation.

Because the p-values are within acceptable limits (under 0.05), then the modeler would sort by the coefficient values, looking for the variables with the strongest correlation effects.

Attribute	beta
mean_concave_points	123.1718
mean_texture	0.4264
worst_area	0.0377
Bias (offset)	-0.7118
mean_radius	-3.1109

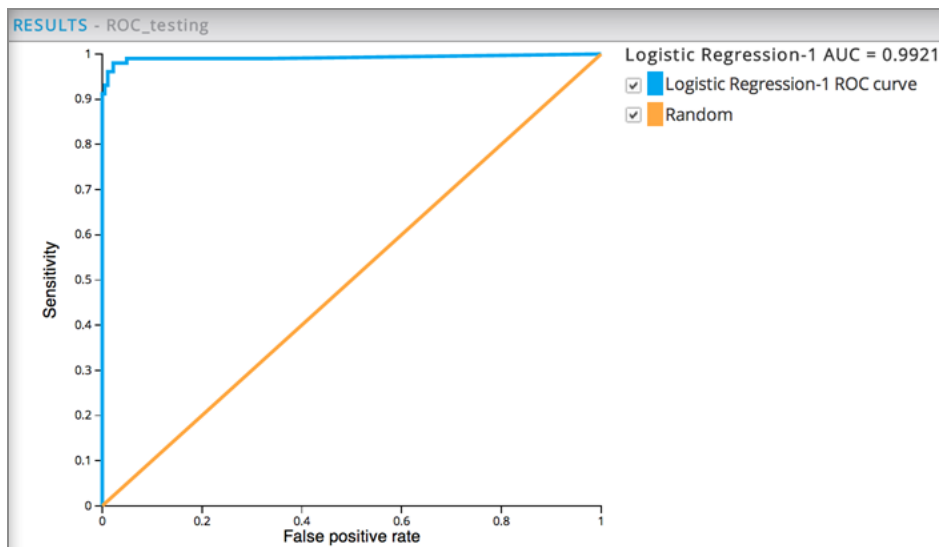
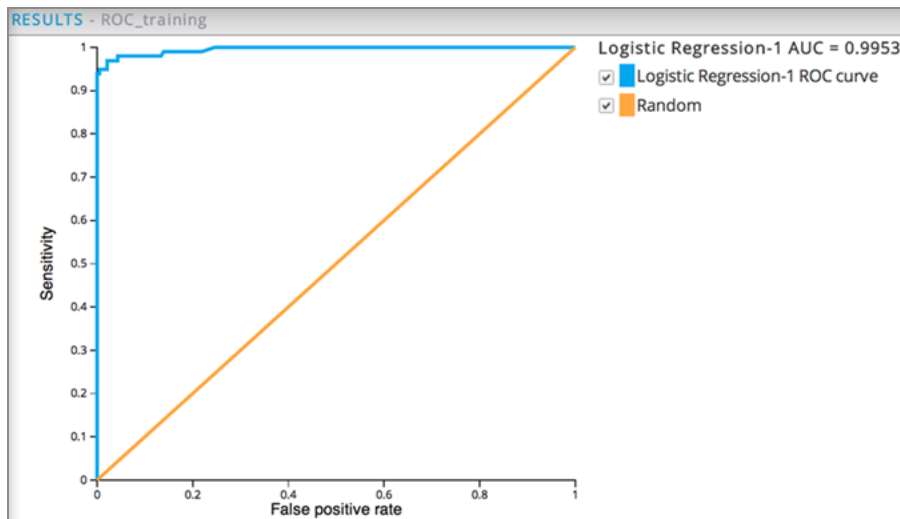
In this example, just looking at the *mean.concavepoints* variable's high coefficient value, the modeler knows it would be quite informative by itself if the model was run only with that independent variable. This model seems to be confirming the intuitive belief that the more non-round or abnormal cell shape/texture, the more chance of cancer.

Adding the ROC Graph operator helps analyze the quality of the model now. The AUC value of .9937 is very high but more believable than 1.0 earlier.



The modeler could rerun the cross-validation study with this reduced variable model to check for more consistent results and confirm there is no longer an over fitting issue.

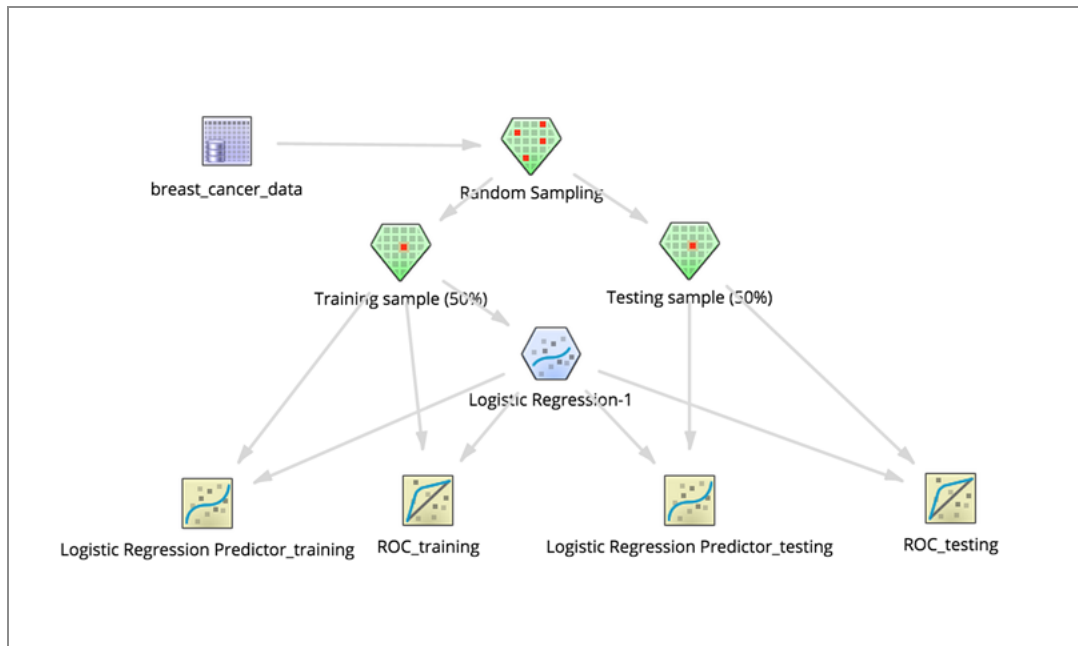
**Note:** The jagged line of the ROC graph is a mild diagnostic sign that the predicted diagnosis sensitivity and variables are acting as thresholds. The modeler might also investigate using the Decision Tree operator, too.



In this example, the two ROC graphs show nearly identical performance when the run against both the training dataset (AUC of .995) and non-training dataset (AUC of .992), which means the model is now a reliable predictor for cancer malignancy.

The modeler's hope is to find a simple model with fewer variables that shows good performance and that cross-validates and performs just as well on non-training data.

As a final step, the modeler would add the Logistic Regression Prediction operator for viewing the prediction column as well as the confidence level columns for both the training data and non-training data.



The prediction confidence levels are high for both the training data predictions and the non-training data predictions. The modeler can be confident in the quality of the logistic regression model. For example, in the first row shown below the predicted diagnosis is benign with more than 99% confidence in the prediction.



P(diagnosis)	C(M)	C(B)
B	0.0004	0.9996
M	1.0000	0.0000
B	0.0000	1.0000
B	0.2079	0.7921
B	0.0007	0.9993
B	0.0036	0.9964
M	1.0000	0.0000
B	0.0018	0.9982
B	0.0003	0.9997
B	0.1215	0.8785
M	1.0000	0.0000
B	0.0208	0.9792

## Classification Modeling with Decision Tree

The Decision Tree operator applies a classification modeling algorithm to a set of input data. This operator is most suitable for predicting or describing data sets with binary or limited number of discrete output categories.

Team Studio provides several implementations of Decision Tree. Select the operator that is the best match for your use case.

Decision Tree implementation and link	Description
<a href="#">Decision Tree</a>	The core Decision Tree offering. It operates on Hadoop datasets.
<a href="#">Decision Tree - MADlib</a>	This Decision Tree operator can be used on databases with MADlib 1.8 or below installed. It uses the C4.5 algorithm in MADlib for its calculations.
<a href="#">Decision Tree Regression - CART</a>	This Decision Tree operator also runs on databases with MADlib installed. For this operator, we support MADlib 1.8 and above. It does

Decision Tree implementation and link	Description
	not work on MADlib 1.7.1, for example. You can use the other Decision Tree - MADlib operator if you have an older version of MADlib. This operator runs a decision tree regression using the CART algorithm.
<a href="#">Decision Tree Classification - CART</a>	This Decision Tree operator also runs on databases with MADlib installed. Like Decision Tree Regression, we support MADlib 1.8 and above. If you have an older version of MADlib, consider using our other Decision Tree MADlib operator. This operator runs a decision tree classification using the CART algorithm.

Like the CART Tree Operator, the Decision Tree Operator creates a branching series of computational steps or logic "tests" that lead to an ultimate decision value.

**i Note:** The CART Operator handles datasets with both discrete and continuous dependent variables while the Decision Tree is only for classification of discrete dependent variables.

The resulting Decision Tree can be used as a predictive model which maps observations about an item (independent variables) to conclusions about the item's target value (dependent variable).

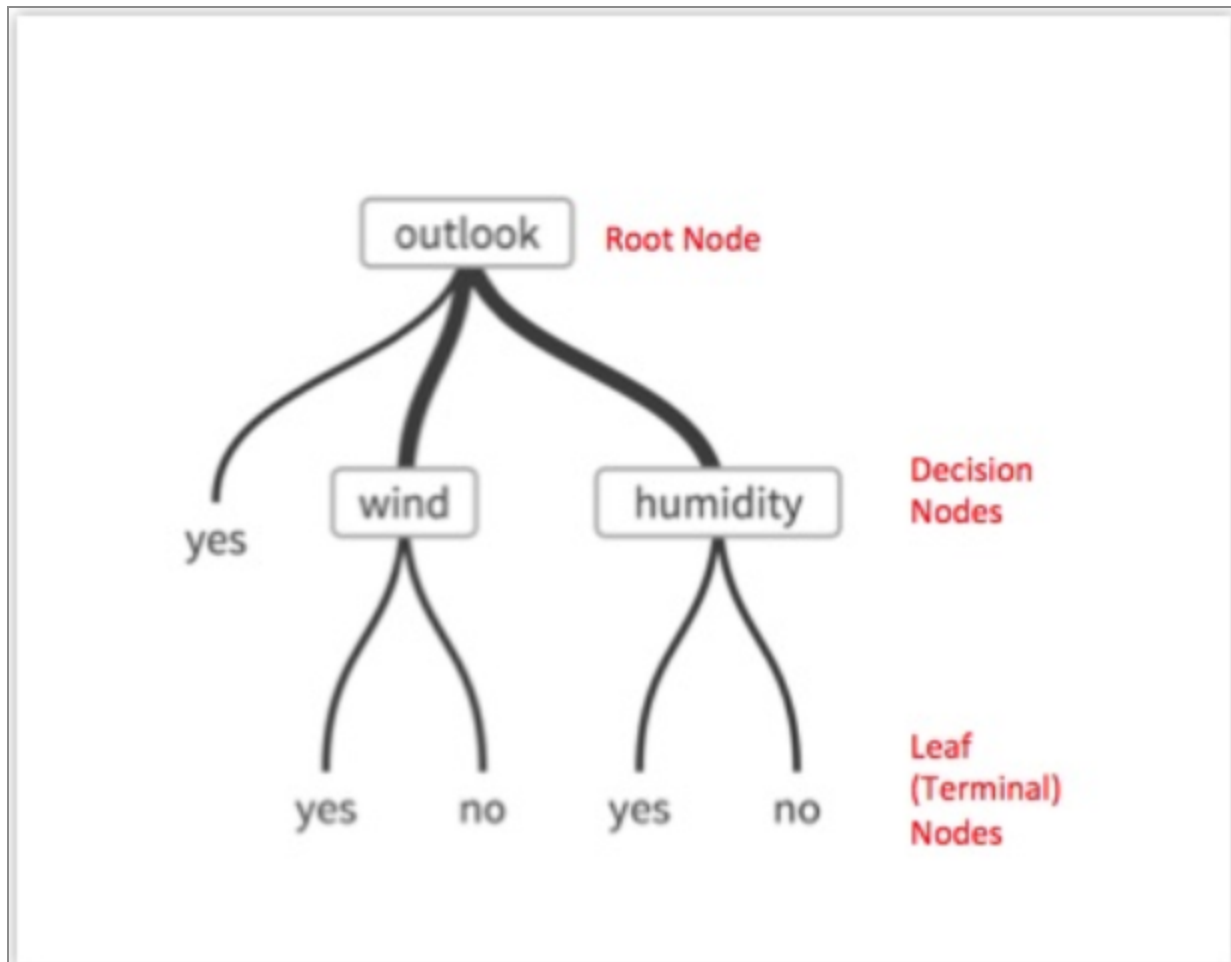
Decision Trees have the following structural properties:

- The hierarchical classification structure called a tree with each decision point, or node, being a point at which the data is segmented into sub-groups.
- The original top node represents the entire dataset and is referred to as the root node.
- The nodes in the tree represent the various decisions, computations or logic tests made in the classification model.
- The bottom leaves (terminal nodes) represent the classification labels where a final decision or classification is made and applied to all the observations in the leaf.
- The branches represent conjunctions of features that lead to the classification labels.

Decision Tree models are attractive because they show both modelers and business users clearly how to reach a decision. They are structured as a sequence of simple questions and

the answers to these questions trace a path down the tree. Therefore, it is easy for a person to manually apply the classification logic path on new data.

The following illustration depicts a simplified decision tree for determining whether to play tennis based on the weather. This shows how visually intuitive the Decision Tree Operator output is in terms of understanding the steps involved in the classification logic.



For a description of the algorithm, see the help for the Decision Tree operator you are using.

After a Decision Tree has been trained within TIBCO Data Science - Team Studio, it can be used in a variety of ways.

- By making inferences that help the user understand the "big picture" of the model. One of the great advantages of decision trees is that they are easy to interpret even by non-technical people. For example, if a decision tree models product sales, a quick glance might tell you that men in the South buy more of your product than

women in the North. Another example might be a model of health risks for insurance policies, and a quick glance at the decision tree might illustrate that smoking and age are important predictors of health.

- By identifying target groups. For example, if a business is looking for the best potential customers for a product, the user could identify the terminal nodes in the tree that have the highest percentage of sales, and then focus the business' sales efforts on the individuals described by those nodes.
- By predicting the target value for specific cases where only the predictor variable values are known. This is known as "scoring." For example, when predicting the outcome of an election, a modeler may look at census results to predict who wins.

Optionally, you can use one of the alternative models to Decision Tree.

- [Alpine Forest Classification](#)
- [Probability Calculation Using Logistic Regression](#)
- [Classification Modeling with Naive Bayes](#)
- [Support Vector Machine Classification](#)

## Decision Tree and CART Operator General Principles

Decision trees are rule-based classifiers that consist of a hierarchy of decision points (the nodes of the tree).

A decision tree can model how the value of a target, dependent variable can be predicted by using the values of a set of predictor, independent variables.

A decision tree could be either a classification tree (labels with a range of discrete values) or a regression tree (labels with a range of continuous (numeric) values).

Classification decision tree analysis is characterized by the predicted outcome being the class to which the data belongs. The Decision Tree operator in TIBCO Data Science - Team Studio performs classification decision tree analysis using the C4.5 (Quinlan 1993) algorithm. In decision tree classification analysis, each dependent variable can have only a discrete list of possible values.

Regression decision tree analysis is characterized by the predicted outcome being considered a real number, such as the price of a house, or a patient's length of stay in a hospital. The TIBCO Data Science - Team Studio CART operators perform regression decision tree analysis using the CART (Breiman et al. 1984) algorithm. In CART decision tree

analysis, each dependent variable can have either a discrete or a continuous list of possible values.

Decision Trees are trained by recursive partitioning: splitting the data set into different groups, and then analyzing each group to perform additional splitting of the data into sub-groups. The algorithm stops splitting based on various stop conditions so it does not over-fit the model.

When you use a decision tree to classify an unknown instance, a single feature is examined at each node of the tree. Based on the value of that feature, the next node is selected. Each node represents a set of records (rows) from the original dataset. Nodes that have child nodes are called "interior" nodes. Nodes that do not have child nodes are called "terminal" or "leaf" nodes. The topmost node is called the "root" node. Unlike a real tree, decision trees are drawn with their root at the top. The root node represents all of the rows in the dataset.

## Decision Tree Concept of Purity

In decision tree construction, concept of purity is based on the fraction of the data elements in the group that belong to the subset.

A decision tree is constructed by a split that divides the rows into child nodes. If a tree is considered "binary," its nodes can only have two children. The same procedure is used to split the child groups. This process is called "recursive partitioning." The split is selected to construct a tree that can be used to predict the value of the target variable. The primary algorithm for deriving a decision tree from a training set employs a greedy approach, which means that it strives for the "purest" of subsets or clearest division of the branch nodes as possible.

This concept of purity is based on the fraction of the data elements in the group that belong to the subset. One way the purity of a set can be defined is as the frequency of its most common constituent. For example, if a set consists of 60% of items in class A, 30% in class B, and 10% in class C, then its purity is 60%.



**Note:** There are other acceptable ways of defining purity that all reach a maximum when all the elements of a set belong to the same class.

Information gain is considered one of the better quantitative measures of increases in purity.

## Information Gain

In decision tree analyses, an important concept is the gain ratio from a parent node to its children.

The gain ratio,  $\Delta$ , measures the gain in purity from parent to children, weighted by the relative size of the subsets, as follows.

$$\text{Gain Ratio } \Delta = I(\text{parent}) - \sum_{\text{children } j} (N_j/N) \log_2(N_j/N)$$

where  $I$  is the purity (or impurity) of a node;

$N_j$  is the number of elements assigned to child node  $j$ ;

$N$  is the total number of elements at the parent node.

The decision tree algorithm tries to perform a splitting that maximizes this gain ratio.

## Pruning or Pre-Pruning

A key concept related to decision trees is pruning or pre-pruning, whereby branches are eliminated from the tree because they do not add enough informational relevance to the model.

Pruning and pre-pruning helps with avoiding over-fitting of the decision tree and makes the tree more compact and easier to read. For a typical dataset, both should be used, unless the algorithm takes too long to run.

The process of pruning involves going through each non-leaf node and determining, based on a confidence value, whether to turn the node into a leaf. In other words, it decides if the sub-tree adds enough extra value to the model. For pruning, the whole tree is built out, and then sub-branches are cut out if they are found to not be good predictors.

The process of pre-pruning limits the decision tree as it is being built based on increases in purity (that is, increase in information gain for the Decision Tree operator and improvement in Gini coefficient for the CART operator). This is a faster process than post pruning, but sometimes can result in too small of a decision tree.

**i Note:** It is standard practice to have pre-pruning and post-pruning interleaved for a combined approach. Post-pruning requires more computation than pre-pruning, yet generally leads to a more reliable tree. No single pruning method has been found to be superior above all others (*Data Mining: Concepts & Techniques*, by Jiawei Han, Micheline Kamber, Jian Pei, page 346).

Pre-pruning is the cheapest way to keep the model small (and to help prevent over-fitting).

## Differences in Decision Tree Algorithms

Various decision tree algorithms such as CART, C4.5, and ID3 differ in certain details.

- How the purity/impurity measure is calculated and at what level of purity the splitting procedure stops. Note that continuing to split a training set until all the leaf nodes are entirely pure usually results in over-fitting the model. The Decision Tree operator uses information gain to measure increase in purity. The CART operators use improvements in the Gini Coefficient.
- Whether the tree is binary or can have a node with more than two children.
- How non-categorical attributes are treated.

## Decision Tree Output Troubleshooting

When you analyze the results of a Decision Tree Model, keep in mind the most important assessment factors.

### Assessing a Decision Tree

When analyzing the results of a Decision Tree Model, the most important assessment factors are as follows.

- It scores well outside of the training data (indicated by the ROC, LIFT, and Goodness of Fit Scoring Operators).
- It avoids the single-node, repetition, and unbalanced tree scenarios.
- It is easily interpretable.

- It makes sense intuitively.

There are a few Decision Tree Operator output results that would make a modeler question the validity of the Decision Tree model, including single-tree nodes, repetition, and replication.

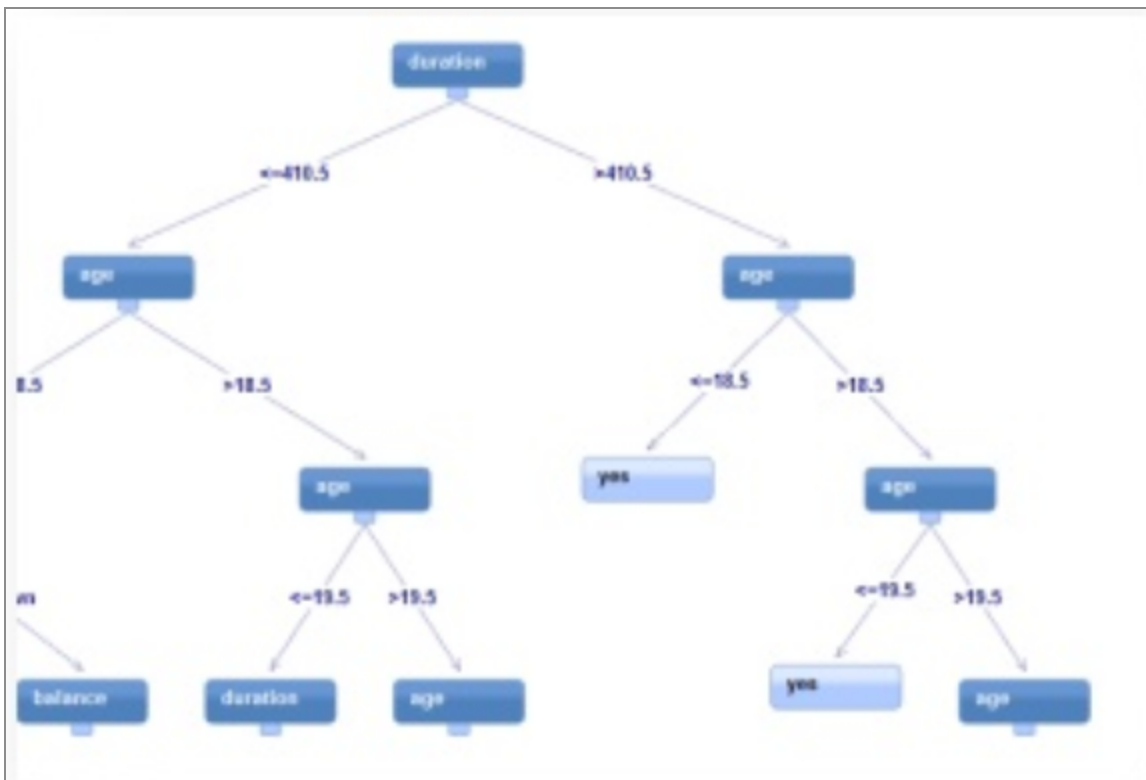
Single-node trees, repetition, or replication conditions might indicate to the modeler that a different form of modeling should be used instead of Decision Trees.

### Single Tree Nodes

For the extreme case of data that does not have any sort of classification structure to it, the Decision Tree output would simply be a single-node tree that assigns elements the label of the most frequently occurring class.

### Repetition

Another anomaly that might be seen for Decision Tree output is a tree with repetition, whereby an attribute is repeatedly tested along a given branch of a tree. For example, the following tree shows the attribute age repeatedly being compared to different threshold values down the tree.





## Replication

Another problem to watch out for is when replication or duplicate sub-trees exist within a tree. This issue could be potentially solved by creating combination variables, or it could be an acceptable model with the replication.

## Decision Tree Use Case

This use case model demonstrates using a decision tree model to assess the characteristics of the client that leads to the purchase of a new product targeted in a direct marketing campaign.

### Direct Marketing Campaign Success Classification

#### Datasets

In this case, the dependent variable is *y* column of whether the client that was called during a direct marketing campaign subscribed to a term deposit (*yes* or *no*). This dataset comes from the UC Irvine site (<http://archive.ics.uci.edu/ml/machine-learning-databases/00222/>), and the exact version used for this case is [bank\\_marketing\\_data.csv](#) (3.7MB).

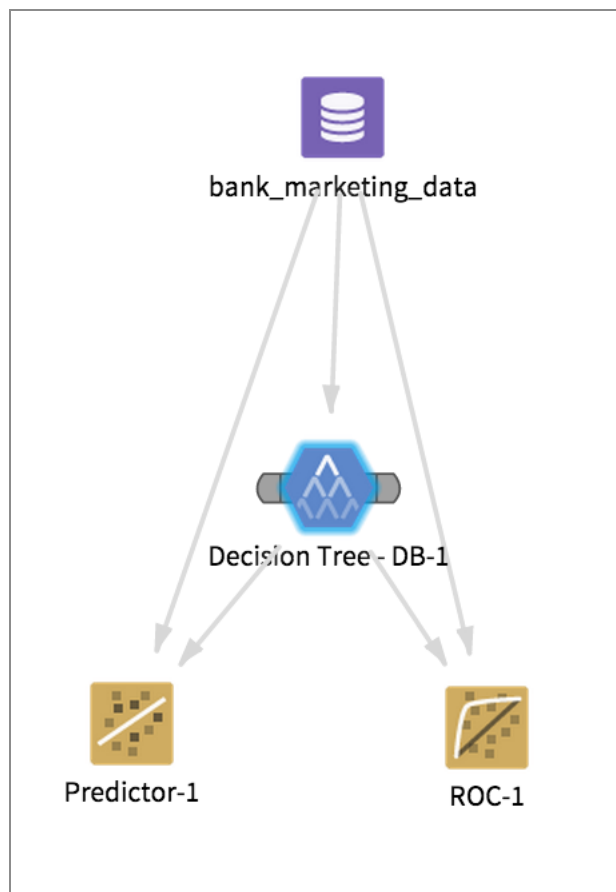
The dataset contains 45,211 observations and 17 columns (16 attributes and 1 dependent value). The attributes analyzed includes the client's age, job type, marital status, education, account default status, housing loan status, personal loan status, contact type, last contact day, last contact month, last contact duration (seconds), # of contacts, # of days since contacted, # of days passed since last contact, previous number of contacts, outcome of previous campaign.

The first few rows are shown below, with *y* as the dependent variable:

bank_marketing_data																	
age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	outcome	y	
38	technician	married	secondary	no	581	yes	no	unknown	6	may	79	1	-1	0	unknown	no	
54	services	married	secondary	yes	0	yes	no	unknown	6	may	138	3	-1	0	unknown	no	
30	blue-collar	married	secondary	no	450	no	no	unknown	6	may	526	2	-1	0	unknown	no	
38	blue-collar	married	secondary	no	384	yes	no	unknown	6	may	1,906	3	-1	0	unknown	no	
30	technician	married	secondary	no	484	yes	no	unknown	6	may	703	1	-1	0	unknown	yes	
54	technician	divorced	secondary	no	21	yes	no	unknown	6	may	381	2	-1	0	unknown	no	
30	unemployed	divorced	secondary	no	1,144	yes	no	unknown	6	may	252	1	-1	0	unknown	no	
38	entrepreneur	married	tertiary	no	898	yes	no	unknown	6	may	132	2	-1	0	unknown	no	
38	admin.	single	secondary	no	221	yes	no	unknown	6	may	211	2	-1	0	unknown	no	
38	management	married	unknown	no	1,349	yes	no	unknown	6	may	100	1	-1	0	unknown	no	
46	blue-collar	married	primary	no	530	yes	no	unknown	6	may	739	3	-1	0	unknown	no	
38	management	married	tertiary	no	389	yes	no	unknown	6	may	262	1	-1	0	unknown	no	
46	services	married	unknown	no	80	yes	no	unknown	6	may	245	2	-1	0	unknown	no	
54	services	married	secondary	no	510	yes	no	unknown	6	may	196	2	-1	0	unknown	no	
30	technician	single	unknown	no	-48	yes	no	unknown	6	may	152	2	-1	0	unknown	no	
46	blue-collar	married	unknown	no	401	yes	no	unknown	6	may	306	2	-1	0	unknown	no	
38	admin.	married	secondary	no	3,047	yes	no	unknown	6	may	285	2	-1	0	unknown	no	
54	technician	divorced	secondary	no	83	yes	no	unknown	6	may	190	3	-1	0	unknown	no	
30	management	married	tertiary	no	32	yes	no	unknown	6	may	122	3	-1	0	unknown	no	
30	admin.	single	secondary	no	115	yes	no	unknown	6	may	66	3	-1	0	unknown	no	
54	blue-collar	married	secondary	no	254	yes	no	unknown	6	may	66	2	-1	0	unknown	no	
38	housemaid	married	secondary	no	0	yes	no	unknown	6	may	59	3	-1	0	unknown	no	
30	self-employed	single	tertiary	no	800	no	no	unknown	6	may	95	2	-1	0	unknown	no	
46	technician	married	secondary	yes	289	no	no	unknown	7	may	51	3	-1	0	unknown	no	
46	services	married	secondary	no	89	yes	no	unknown	7	may	479	2	-1	0	unknown	no	

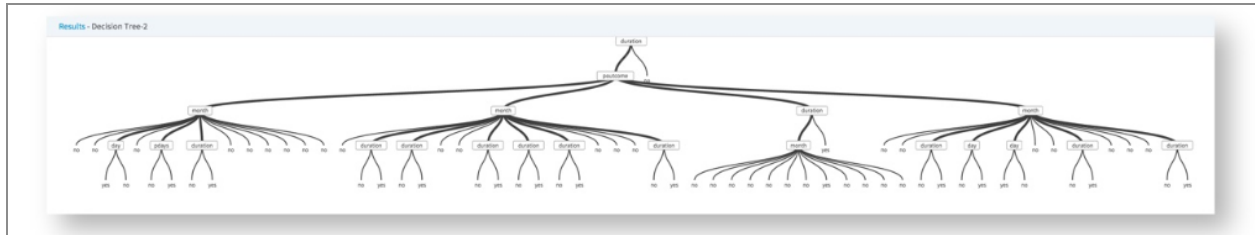
## Workflow

The overall configuration for this Analytic Flow might be something like the following:



For purposes of this example, assume the modeler is trying to understand the question, "Which sub-group of clients should be targeted for the highest subscription rate of a term deposit product?"

If the modeler selects all the variables as independent variables, the results are quite complicated, repetitive and do not seem to reveal any immediate business insights.



In this particular example, the variable month is exploding the Decision Tree and not seeming to add much business decision value.

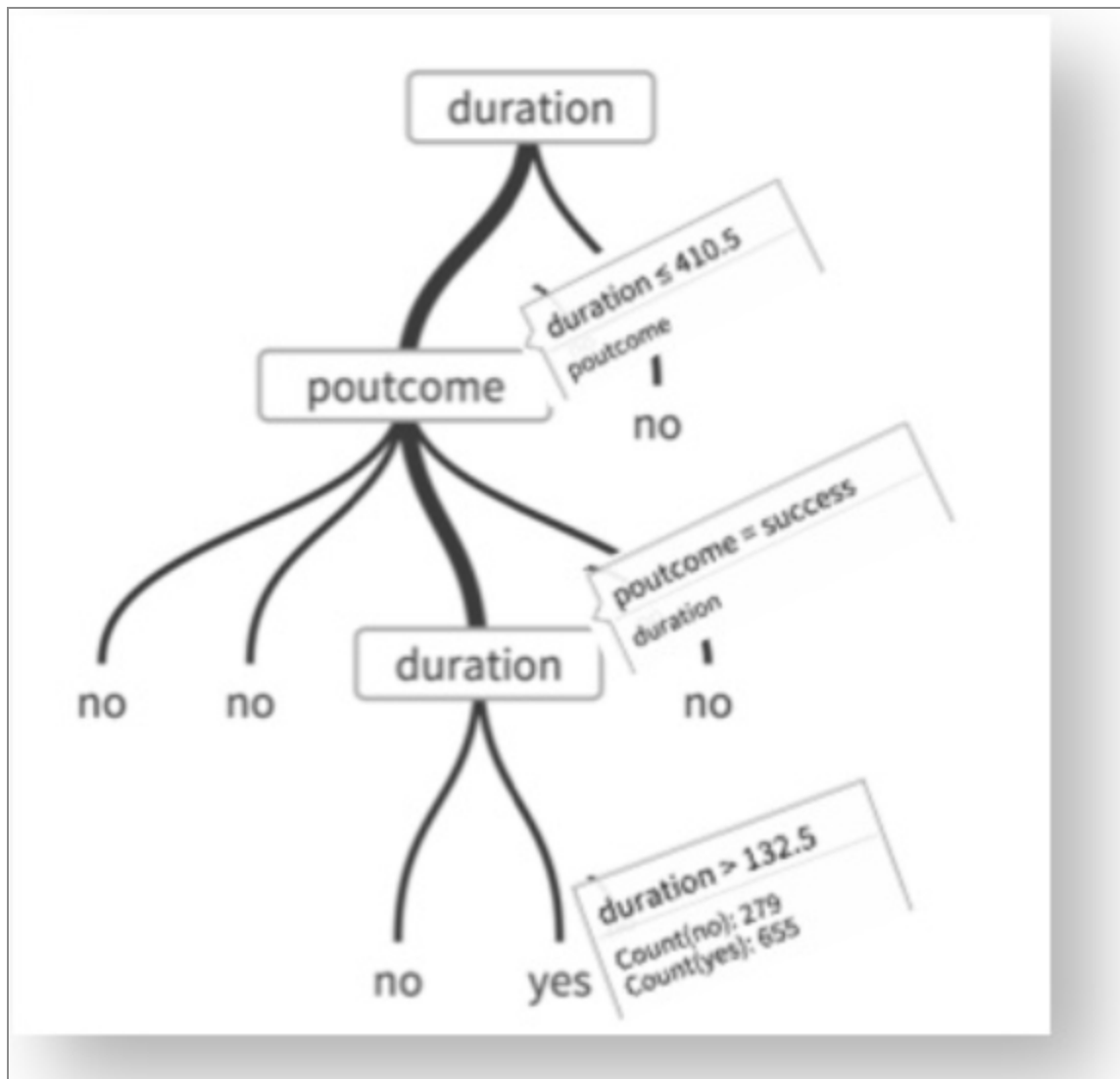
One approach for avoiding such over-fitting might be to test some initial assumptions the modeler might have regarding the independent variables.

For example, perhaps different clients are more likely to subscribe to a term deposit if they have been well informed on the product during the campaign (got a long enough contact), or already subscribed in the past after a successful marketing campaign.

With this hypothesis in mind, the modeler might reduce the potential independent variables analyzed and select only the *contact*, *duration*, *campaign*, *pdays*, *previous*, *outcome* variables to include in the analysis. The overall Decision Tree Operator configuration properties can be kept at their default values.

## Results

The results in this case are more understandable, revealing the insight that clients for whom the last campaign was successful AND for whom the last call was between 132 and 410 seconds seem to be more likely to subscribe to the term deposit product.



Including the Predictor Operator provides additional model results:

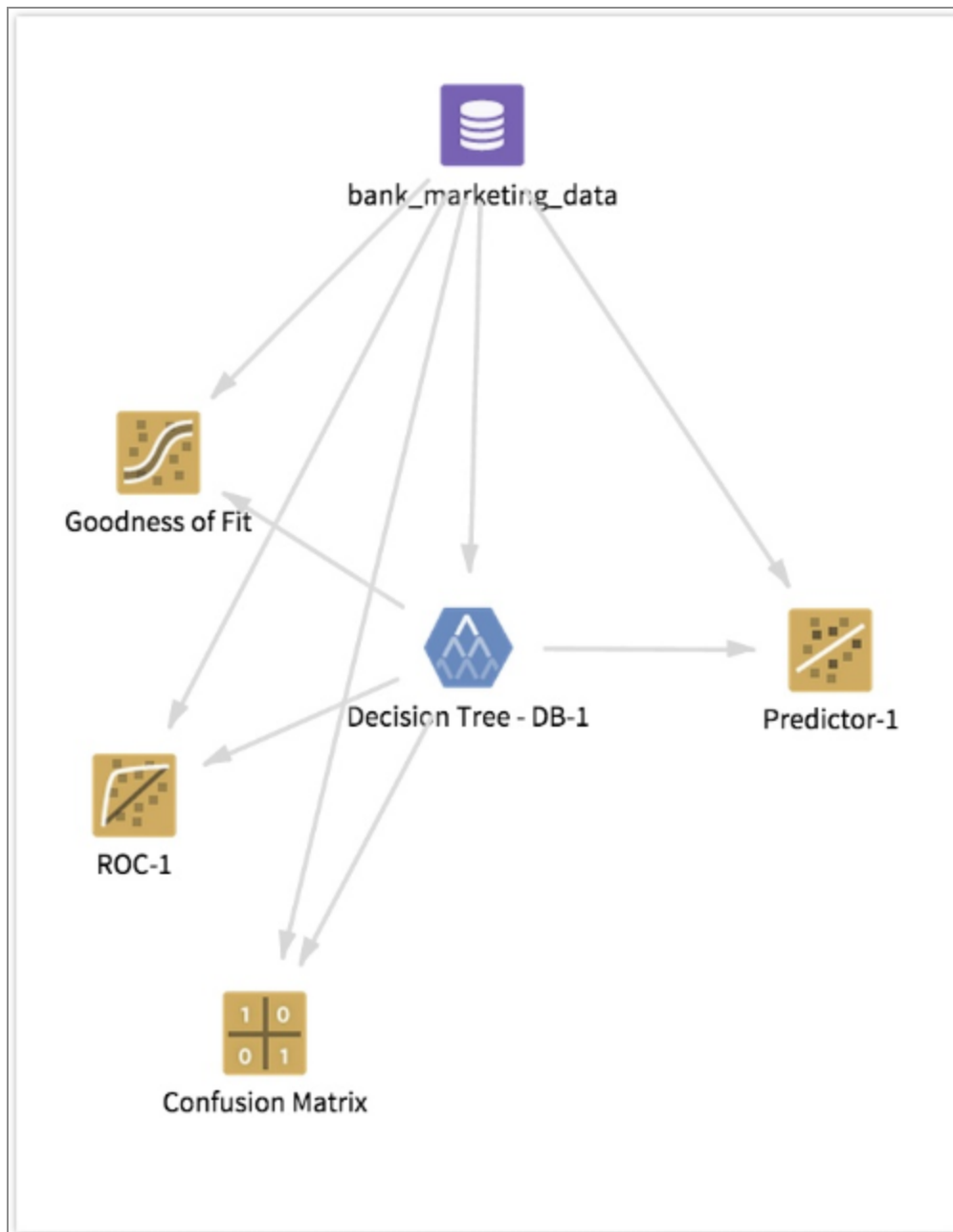
DOC Use Case Decision Tree 1 Alpine Workflow Result

Result Log | bank\_marketing\_data | Decision Tree-2 | Predictor-1

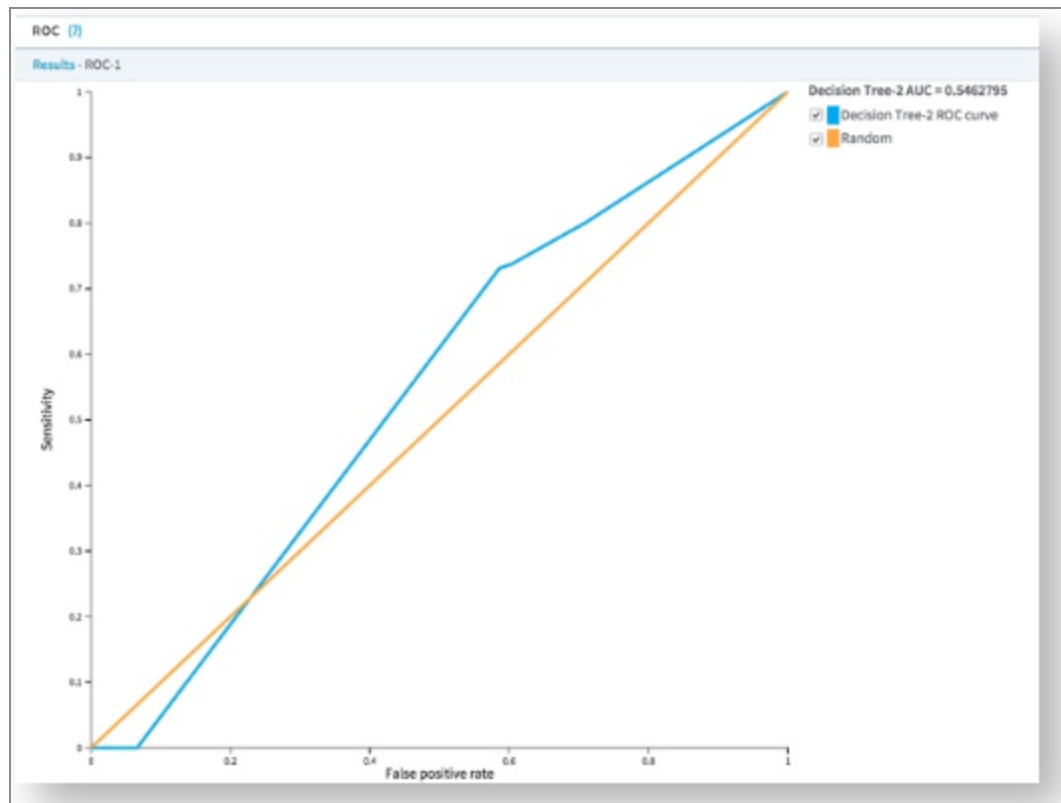
age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	postcome	y	P(y)
55	blue-collar	married	primary	no	23	yes	no	unknown	5	may	251	1	-1	0	unknown	no	no
55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	0	unknown	yes	no
31	management	single	tertiary	no	0	yes	no	unknown	5	may	562	1	-1	0	unknown	no	no
31	services	married	secondary	no	388	yes	no	unknown	5	may	730	2	-1	0	unknown	no	no
55	blue-collar	married	secondary	no	89	yes	no	unknown	5	may	106	3	-1	0	unknown	no	no
23	student	single	secondary	no	157	yes	no	unknown	6	may	54	1	-1	0	unknown	no	no
55	admin.	married	secondary	no	183	yes	no	unknown	6	may	150	1	-1	0	unknown	no	no
39	management	single	secondary	no	1877	yes	no	unknown	6	may	185	1	-1	0	unknown	no	no
47	self-employed	married	unknown	no	935	yes	no	unknown	6	may	225	1	-1	0	unknown	no	no
55	services	divorced	secondary	no	141	yes	no	unknown	6	may	262	2	-1	0	unknown	no	no
31	management	single	tertiary	no	1852	yes	no	unknown	6	may	170	3	-1	0	unknown	no	no
39	services	married	secondary	no	1205	yes	no	unknown	6	may	118	2	-1	0	unknown	no	no
55	unemployed	married	tertiary	no	383	no	no	unknown	6	may	343	3	-1	0	unknown	no	no
47	services	married	primary	no	222	yes	no	unknown	7	may	68	1	-1	0	unknown	no	no
31	technician	single	secondary	no	147	yes	no	unknown	7	may	374	1	-1	0	unknown	no	no
31	services	married	tertiary	no	309	yes	yes	unknown	7	may	294	1	-1	0	unknown	no	no
31	services	single	secondary	no	-45	no	no	unknown	7	may	246	2	-1	0	unknown	no	no
39	management	single	tertiary	no	406	yes	no	unknown	7	may	68	1	-1	0	unknown	no	no
39	management	single	unknown	no	2887	yes	no	unknown	7	may	42	2	-1	0	unknown	no	no
47	technician	married	primary	no	145	yes	no	unknown	7	may	157	6	-1	0	unknown	no	no
39	admin.	married	secondary	no	419	yes	yes	unknown	8	may	35	1	-1	0	unknown	no	no
31	technician	single	tertiary	no	1798	yes	no	unknown	8	may	393	1	-1	0	unknown	no	no
31	blue-collar	married	secondary	no	335	yes	no	unknown	8	may	226	1	-1	0	unknown	no	no
39	management	single	tertiary	no	375	yes	no	unknown	8	may	66	2	-1	0	unknown	no	no
31	blue-collar	married	primary	no	68	yes	no	unknown	8	may	226	1	-1	0	unknown	no	no

The prediction  $P(y)$  value (yes or no) uses a threshold assumption of greater than 50% confidence that the prediction happens.

Hooking in additional model diagnostic tools, such as the ROC graph, Confusion Matrix, and Goodness Of Fit Operators, provides additional feedback on the quality of the model.



The ROC (Receiver Operating Characteristic) Curve plots the sensitivity (% of correctly classified results) vs. false positives. The larger the AUC value, the more predictive the model is. Any AUC value over .80 is typically considered a "good" model. A value of .5 just means the model is no better than a "dumb" model that can guess the right answer half of the time.

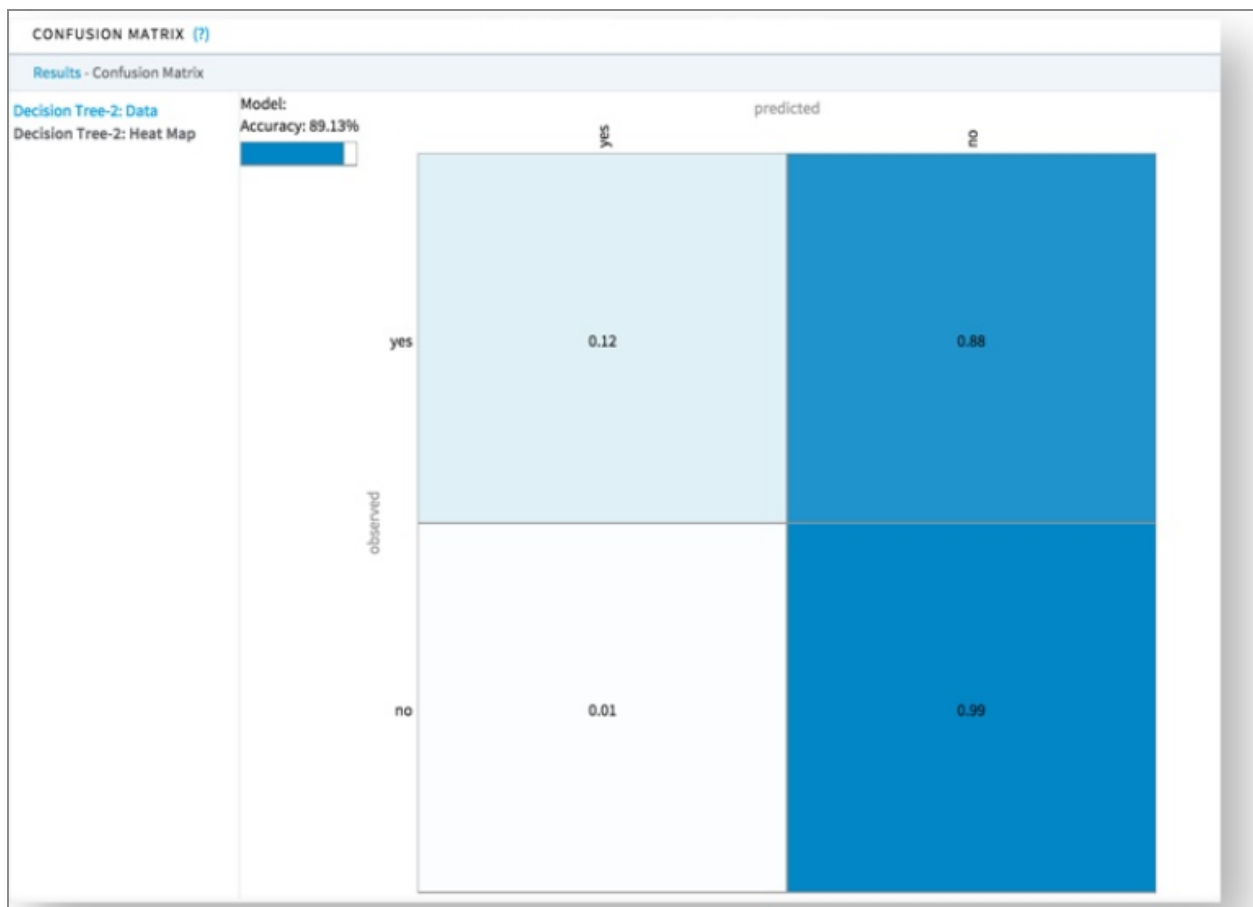


The Goodness of Fit Operator provides model accuracy, error and other validation data.

Node	Accuracy	Error	Stats	Recall	Precision	F1	Specificity	Sensitivity
Decision Tree-2	0.8913318	0.1086682	yes	0.1238419	0.7012848	0.2105094	0.9930114	0.1238419
Decision Tree-2	0.8913318	0.1086682	no	0.9930114	0.8953407	0.9416501	0.1238419	0.9930114

The following confusion matrix table summarizes model accuracy data, and the heat map visual summarizes the actual versus predicted counts of a classification model. These two summaries help assess the model's accuracy for each of the possible class values in a visually intuitive manner.

CONFUSION MATRIX (?)			
Results - Confusion Matrix			
Decision Tree-2: Data Decision Tree-2: Heat Map		Predicted (yes)	Predicted (no)
	Observed (yes)	655	4634
	Observed (no)	279	39643
	Class Precision	0.7012848	0.8953407
		Accuracy: 0.8913317	



## Classification Modeling with Naive Bayes

Naive Bayes is a classification modeling method, like Logistic Regression and Decision Tree models.



**!** **Important:** The database version of this operator has been deprecated and was removed in version 6.1. However, a new and improved Naive Bayes operator was created to replace it. To use this new operator, you must remove the old Naive Bayes (database) operator from your workflow and replace it with the new [Naive Bayes \(DB\)](#) operator. Support for Naive Bayes on Hadoop remains unchanged; see [Naive Bayes \(HD\)](#) for information.

A Naive Bayes model predicts a categorical or binary outcome, such as "yes" or "no." Specifically, the TIBCO Data Science - Team Studio Naive Bayes operator calculates the probability of a particular event occurring. It can be used to predict the probability of a certain data point being in a particular classification.

For an example use case, see [Naive Bayes Use Case](#).

The Naive Bayes theorem assumes that the predictors or variables are all independently related to the outcome. For example, it can reflect the probability of a customer buying a computer based on the age of the customer (independently from the income, sex, or other attributes of the customer).

- The Naive Bayes is a surprisingly accurate classifier given that the assumption of independence is rarely true.
- This assumption of independence gives the Naive Bayes classifier the extra benefit of being computationally lightweight, requiring only small training sets for the calculation of means and variances rather than a more complex covariance matrix.
- Naive Bayes classification methodology is also particularly helpful over other classification methods when faced with the 'curse of dimensionality'; that is, when the number of predictors, or independent variables, is very high.
- Some typical examples of Naive Bayes modeling include spam detection, biological classifications, and financial loan forecasting.

In summary, the TIBCO Data Science - Team Studio Naive Bayes Operator implements a fast, effective classification tool with results that are easy to interpret.

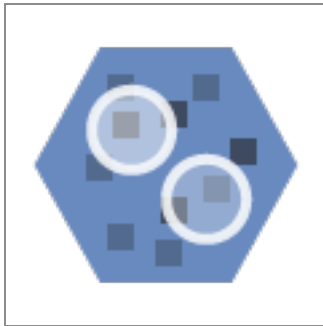
## Alternative Models

- Logistic Regression ([Logistic Regression \(DB\)](#)) or ([Logistic Regression \(HD\)](#))
- [Decision Tree](#)
- [Alpine Forest Classification](#)

- [Support Vector Machine Classification](#)

## Naive Bayes Use Case

Naive Bayes modeling is useful for a variety of different applications in the biological and medical fields, in online document and spam classification areas, for supply chain stock management, and for financial predictions. Here is a specific use case example from the biological field.



## Biological Leukemia Classification

### Datasets

The source dataset for this use case is [LeukemiaTrainingData.csv](#) (9KB). Training biological leukemia gene dataset used in this use case was obtained from: <http://research.cs.queensu.ca/home/xiao/dm.html>.

This example dataset was created from biological study measurements - the data was cleaned, normalized and filtered down to the top 31 genes associated with leukemia. Then the data was transposed so that each column represents an Attribute (Gene name) and each row represents a biological sample record. The **Class** column was added to indicate whether the type of leukemia related to the sample was either "ALL" or "AML".

The goal of applying the Naive Bayes model on the dataset is to find those genes, or attributes, that have high discriminating ability to distinguish between different types of leukemia. Therefore, in this case, the Dependent Value represents the Type of Leukemia, which is a categorical variable with possible values of either "ALL" or "AML".


The first rows of this dataset are shown below:

LeukemiaTrainingData																
id_at	U46751_at	U50736_ma1_at	U82759_at	X04085_ma1_at	X14008_ma1_f_at	X15949_at	X17042_at	X52142_at	X59417_at	X74262_at	X85116_ma1_s_at	X95735_at	Y00787_s_at	Y12670_at	Z15115_at	Target
1.894	928	119	2.000	2.917	416	7.972	261	5.216	1.370	334	713	93	524	3.425	ALL	
1.489	706	87	1.220	93	588	2.408	269	7.743	1.624	182	724	116	201	6.216	ALL	
1.962	383	149	1.781	560	201	1.912	122	5.746	768	675	68	118	371	9.114	ALL	
6.903	2.539	805	2.887	13.087	20	8.479	20	528	130	3.397	2.612	16.000	1.064	802	AML	
3.717	2.545	1.137	6.000	4.063	35	8.859	20	1.561	309	2.195	3.348	3.104	1.296	967	AML	
1.379	1.062	118	1.395	2.716	104	3.460	20	3.424	1.184	169	307	5.345	337	1.920	ALL	
1.939	1.175	409	463	1.781	97	1.094	96	5.204	1.058	176	492	522	167	1.329	ALL	
2.111	883	102	220	1.631	515	1.232	20	1.273	882	203	390	136	314	1.616	ALL	
1.437	805	59	773	441	169	1.750	105	3.262	716	342	433	183	190	3.074	ALL	
4.719	2.402	652	5.878	4.192	35	8.518	20	782	279	727	2.947	8.054	1.807	2.271	AML	
1.514	1.668	248	775	1.683	314	140	179	2.073	655	571	20	455	689	6.153	ALL	
944	825	67	1.143	635	393	1.232	260	6.084	1.625	540	848	38	330	5.617	ALL	
1.571	899	770	3.970	108	598	1.687	277	3.840	1.409	300	222	185	655	6.110	ALL	
3.088	2.535	1.083	2.027	5.778	59	3.259	20	1.214	311	1.017	3.297	12.715	1.467	1.843	AML	
1.496	1.398	667	823	1.716	91	416	20	7.724	2.221	20	309	301	574	3.118	ALL	
587	626	306	628	833	27	1.343	69	6.541	953	139	20	1.278	174	2.525	ALL	
1.983	1.309	432	1.271	846	339	1.519	184	6.482	1.459	1.074	615	20	146	4.692	ALL	
990	1.286	20	115	1.976	104	291	109	1.810	500	20	20	150	273	5.374	ALL	
1.106	2.137	262	2.140	2.327	79	3.628	20	490	336	1.639	1.050	216	1.038	1.082	AML	
1.747	1.066	735	6.133	830	267	403	191	5.528	1.339	378	52	159	463	3.810	ALL	
2.286	515	99	1.990	637	747	2.133	499	7.288	1.995	304	938	29	625	5.831	ALL	
585	731	366	431	1.525	20	178	142	2.443	862	417	20	146	780	1.601	ALL	
1.262	2.943	1.157	7.260	7.821	23	5.849	20	1.784	275	983	1.548	2.096	1.085	1.568	AML	
1.401	942	410	1.655	1.143	403	392	190	3.821	1.051	199	693	241	716	4.261	ALL	
1.347	747	20	294	4.009	20	5.252	20	846	209	530	408	3.684	258	1.277	ALL	
2.731	817	359	4.004	883	748	180	326	7.629	4.253	1.071	827	119	636	11.241	ALL	
5.868	3.965	618	5.162	15.761	71	10.603	20	1.648	194	1.096	4.863	16.000	1.051	2.007	AML	
7.113	1.641	822	3.077	1.953	64	7.858	20	1.203	596	2.274	2.224	15.202	733	1.238	AML	
528	842	147	537	3.300	46	597	26	4.194	1.306	117	247	215	270	1.790	ALL	
1.784	864	188	662	3.576	307	2.525	159	1.756	2.050	452	360	94	180	3.382	ALL	
1.739	825	160	350	886	48	354	20	699	176	177	122	206	180	2.572	ALL	
7.061	7.396	671	4.681	1.661	64	1.385	20	99	51	4.465	1.631	10.041	7.538	916	AML	

## Workflow

For this use case, a Naive Bayes model can be built from the training Naive Bayes Leukemia dataset by adding the Naive Bayes and Naive Bayes Prediction Operators in a new flow:

The Naive Bayes Operator is configured with the **Target** column as the dependent variable, as follows:



### Naive Bayes - Leukemia Type Probability

---

**Dependent Column** Target

**Force Retrain** ☒ Yes ☐ No

**Calculate Deviance** ☐ true ☒ false

**Columns** Select Columns

All the datasets gene columns are selected (except for the first **ID** column that represents the sample number).

## Results

Saving and running the model produces the following results. The **Summary** results tab displays:

RESULTS - Naive Bayes	
Summary	Class Priors
Data	Prior (AML): 0.2895
	Prior (ALL): 0.7105

The training dataset has 28.95% occurrence of AML class of leukemia and 71.05% occurrence of ALL class of leukemia. This provides information on the historical training dataset.

**i Note:** It is important that the priors being used to train the model are similar to the class distribution in the overall population.

The **Data** results tab displays each of the top 31 predicting gene attributes along with their normal distribution curve Mean and Standard Deviation calculations.

RESULTS - Naive Bayes				
Summary	Attribute	"Class"	Mean	Standard Deviation
Data				
	D26156_s_at	AML	578.8182	214.6937
	D26156_s_at	ALL	1,306	520.5034
	D49950_at	AML	243.9091	93.181
	D49950_at	ALL	76.963	52.7888
	L08246_at	AML	3,767.4545	1,941.5221
	L08246_at	ALL	1,066.5556	668.5876
	L13278_at	AML	25.0909	12.3082
	L13278_at	ALL	131.1111	89.4356
	L47738_at	AML	203.6364	203.8157
	L47738_at	ALL	1,251.7407	897.8516
	L49229_f_at	AML	41.3636	28.5527
	L49229_f_at	ALL	226.2963	172.2984
	M11147_at	AML	14,140.8182	2,828.1452
	M11147_at	ALL	7,835.4444	3,838.8782
	M16038_at	AML	1,811.6364	953.6882
	M16038_at	ALL	375.7778	239.6609
	M19045_f_at	AML	7,160.4545	4,784.6615
	M19045_f_at	ALL	1,559.8519	1,185.6473
	M21551_rna1_at	AML	464	115.3265
	M21551_rna1_at	ALL	234.963	127.8176

This data is helpful for understanding the measurement values that indicate the different classes of leukemia and for selecting the genes that are the most predictive of the type of Leukemia. For example, the D26156 gene attribute indicates AML class of leukemia when it has a value of 578 +/- 215, whereas it indicates ALL class of leukemia when it has a value of 1306 +/- 520.

Attribute	"Class"	Mean	Standard Deviation
D26156_s_at	AML	578.8182	214.6937
D26156_s_at	ALL	1,306	520.5034

In this use case, most of the normal distribution curves for each class of leukemia are different from each other (that is, different Means and small Standard Deviations). This is because the starting dataset already reduced over 7,000 different genes down to these top 31 most predictive leukemia genes. So we expect that most of the genes are significant to the model.

Including the Naive Bayes Predictor Operator provides additional model results:

	U32944_at	U46751_at	U50136_rna1_u	U82759_at	X04085_rna1_u	X14008_rna1_u	X15949_at	X17042_at	X52142_at	X59417_at	X74262_at	X85116_rna1_u	X95735_at	Y00787_s_at	Y12670_at	Z15115_at	Target	P(Target)	C(AML)	C(ALL)
51	277	1347	747	20	294	4009	20	5252	20	846	209	530	408	3684	258	1277	ALL	ALL	0.0000	1.0000
55	2591	1888	1837	20	2474	817	822	1150	214	2455	3139	1263	926	282	784	3714	ALL	ALL	0.0000	1.0000
95	651	11618	3818	1050	5065	6612	20	11003	20	1628	313	478	6218	6506	647	1365	AML	AML	1.0000	0.0000
326	3110	944	825	67	1143	635	393	1232	260	6084	1625	540	848	38	330	5617	ALL	ALL	0.0000	1.0000
469	3502	1894	928	119	2000	2917	416	7972	361	5216	1370	334	713	93	524	3425	ALL	ALL	0.0000	1.0000
336	1530	1747	1066	735	6133	830	267	403	191	5528	1339	378	52	159	463	3810	ALL	ALL	0.0000	1.0000
105	3349	1298	1124	393	1057	557	277	177	107	3016	1372	162	298	333	600	8444	ALL	ALL	0.0000	1.0000
336	108	1739	825	160	350	886	48	354	20	699	176	177	122	206	180	2572	ALL	ALL	0.0000	1.0000
43	2089	1496	1398	667	823	1716	91	416	20	7724	2221	20	309	301	574	3118	ALL	ALL	0.0000	1.0000
247	328	3088	2555	1083	2027	5778	59	3259	20	1214	311	1017	3297	12715	1467	1843	AML	AML	1.0000	0.0000
77	1143	1962	383	149	1781	560	201	1912	122	5746	768	675	68	118	371	9114	ALL	ALL	0.0000	1.0000
311	2952	2286	515	99	1990	637	747	2133	499	7288	1995	304	938	29	625	5831	ALL	ALL	0.0000	1.0000
377	1283	990	1286	20	115	1976	104	291	109	1810	500	20	20	150	273	5374	ALL	ALL	0.0000	1.0000
118	1002	1379	1062	118	1395	2716	104	3460	20	3424	1184	169	307	5345	337	1920	ALL	ALL	0.0000	1.0000
73	295	5868	3965	618	5162	15761	71	10603	20	1648	194	1096	4883	16000	1051	2007	AML	AML	1.0000	0.0000
95	133	5749	1373	1099	3072	13076	33	8855	20	1583	20	1999	3482	16000	1513	936	AML	AML	1.0000	0.0000
90	288	1106	2137	262	2140	2327	79	3628	20	490	336	1639	1050	216	1038	1082	AML	AML	1.0000	0.0000
265	1480	1437	805	59	773	441	169	1750	105	3262	716	342	433	183	190	3074	ALL	ALL	0.0000	1.0000
38	469	1262	2943	1157	7260	7821	23	5849	20	1784	275	983	1548	2096	1085	1568	AML	AML	1.0000	0.0000

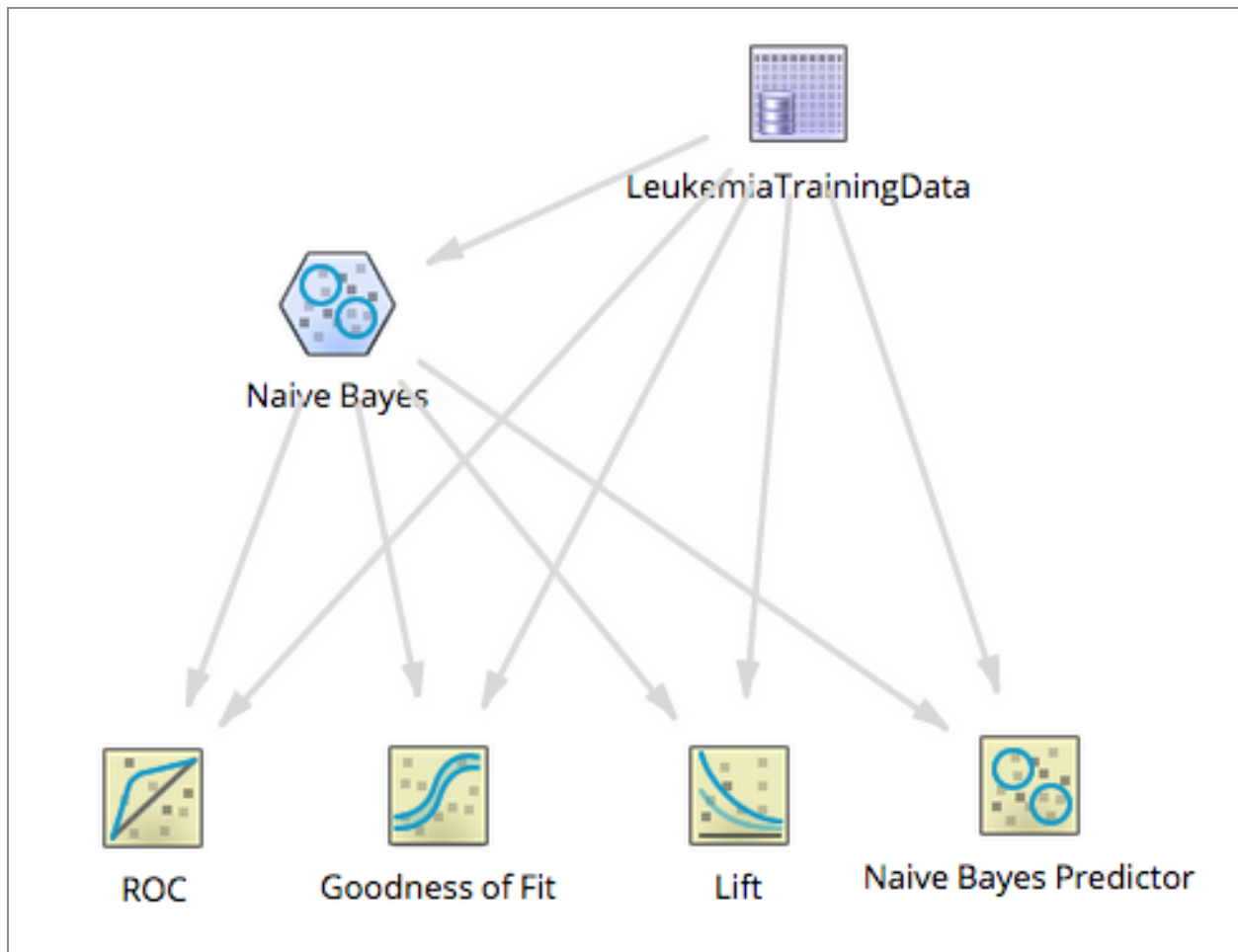
The prediction value (0 or 1) uses a threshold assumption of > than 50% confidence that the prediction happens.

The **C(AML)** column indicates the confidence that the dependent value is AML.

The **C(ALL)** column indicates the confidence that the Dependent value is ALL.

**i Note:** This dataset has perfect prediction results. This might be explained due to the fact that it has been reduced to the top 31 most predictive leukemia attribute genes.

The ROC, Goodness of Fit and LiFT scoring operators should also be added to assess the model.



## Summary

These results indicate the Naive Bayes model is highly predictive.

The modeler can then test the model on a different dataset than the training dataset to illustrate how the model can be used to predict leukemia types given new samples.

- Add the Leukemia Test dataset to the flow, with the same column headings but where the class (target value) of leukemia is not yet known.
- Run the following flow configuration with the test dataset connected to the Naive Bayes Predictor instead of the training dataset.
- Running the model now provides predictions for which type of leukemia each sample row contains with high confidence, illustrating the predictive power of a Naive Bayes model.

# Computed Metrics and Use Case for the Regression Evaluator

For model validation, the Regression Evaluator operator uses the MLlib regression evaluator. You can use it with.

## Metrics

Accuracy	Description	Equation
<b>Mean Squared Error (MSE)</b>	<p>The sum of the squared difference between actual and predicted columns, divided by the number of observations in the dataset.</p> <p>A value of 0 indicates that the predicted and actual values are the same for each observation. A very high value indicates that on average, the difference between actual and predicted values is very large.</p>	$MSE = \frac{1}{N} \sum_{i=1}^N (Y_i - \hat{Y}_i)^2$
<b>Mean Squared Error (MSE)</b>	The square root of the MSE metric.	$RMSE = \sqrt{\frac{\sum_{i=1}^N (Y_i - \hat{Y}_i)^2}{N}}$
<b>Mean Absolute Error (MAE)</b>	<p>The average of the absolute difference between the predicted and actual columns for each observation. A value of 0 indicates that the predicted and actual values are the same for each observation. A very high value indicates that on average, the difference between actual and predicted values is very large in both directions.</p>	$MAE = \frac{1}{N} \sum_{i=1}^N  Y_i - \hat{Y}_i $
<b>Coefficient of Determination (<math>R^2</math>)</b>	<p>The proportion of the variance in the dependent variable that is predictable from the independent variables. See <a href="#">Coefficient of Determination</a> for more details.</p> <p>An <math>R^2</math> of 1 indicates that the regression line perfectly fits the data, while a value of 0 indicates that it doesn't fit the data at all. When a value is negative, it indicates that the regression model is worse than the horizontal line, and does not capture the trend of the data.</p>	$R^2 = 1 - \frac{RSS}{TSS}$ <p><b>RSS</b> = sum of squares of residuals</p> <p><b>TSS</b> = total sum of squares</p>



Accuracy	Description	Equation
<b>Mean Absolute Percentage Error (MAPE)</b>	<p>A measure of prediction accuracy. It expresses accuracy as a percentage. However, it cannot be used if there are zero values, because there would be a division by zero. If a row contains a zero value, the row is skipped.</p> <p>See <a href="#">Mean Absolute Percentage Error</a> for more details.</p>	$MAPE = \frac{100}{n} \sum_{t=1}^n \left  \frac{A_t - F_t}{A_t} \right $

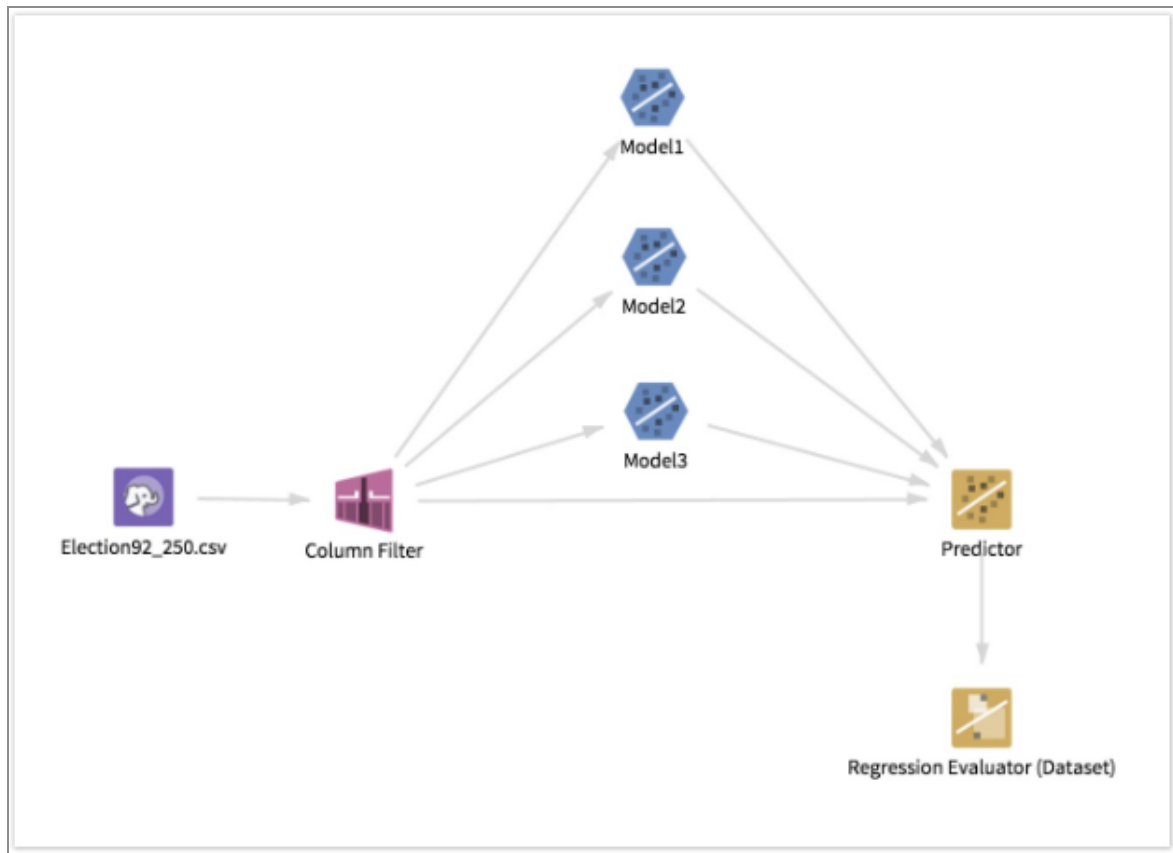
See the MLlib information at [the Spark site](#) for more information.

The Regression Evaluator operator (for either [Regression Evaluator \(DB\)](#) or [Regression Evaluator \(HD\)](#)) handles null values by eliminating them from the input calculation. If you want a different behavior, use the Null Value Replacement operator (for either [Null Value Replacement \(DB\)](#) or [Null Value Replacement \(HD\)](#)) on the initial training data to replace bad or missing values. All of the TIBCO Data Science - Team Studio MapReduce operators replace bad data with null values in a format suitable for the Regression Evaluator, so this operation does not fail on output of a MapReduce operator such as a Column Filter.

### Use with Team Studio Predictors

One likely use case for this operator is as an evaluator for a Linear Regression operator (either [Linear Regression \(DB\)](#) or [Linear Regression \(HD\)](#)). It can be used to compare different regressions. To do this, the user should connect each of the model operators and the dataset used to train them to one TIBCO Data Science - Team Studio Predictor, then connect the Predictor to this operator. To configure the Regression Evaluator, select the original dependent variable column passed through the Predictor and the columns generated by the Predictor (one for each model). The last few columns passed in through the Predictor are the predictions made for each of the models that it predicted on.

## Example Workflow



## Collaborative Filtering

Collaborative filtering is commonly used for recommender systems. This approach collects information about a user's preferences and uses that to make predictions on what they may like based on their similarity to other users who have rated similar products.

A well-known example of collaborative filtering is Amazon's "Users who bought this item also bought..." recommendation system. ([source](#)) We use the ALS (alternating least squares) method from MLlib to compute the latent factors. ([source](#))

You can see a visual example of how this works in the [Collaborative filtering Wikipedia article](#).

Three operators are used for collaborative filtering:

- [Collaborative Filter Trainer](#)

- [Collaborative Filter Recommender](#)
- [Collaborative Filter Predictor](#)

By combining these in a workflow, you can build a customized recommender system.

## Prediction Threshold

You can use the Classification Threshold Metrics operator to determine the prediction threshold that determines what the predicted class is, based on the probabilities that the model outputs.

Many classification models actually output a "score" (often a probability) for each class, where a higher score indicates higher likelihood. If we take the example of a binary classification, the model can output a probability for each class:  $P(Y=1|X)$  and  $P(Y=0|X)$ .

Instead of simply taking the higher probability, there can be some cases where the model might need to be tuned so that it predicts only a class when the probability is very high (for example, only block a credit card transaction if the model predicts fraud with >90% probability). Therefore, there is a prediction threshold that determines what the predicted class is, based on the probabilities that the model outputs.

The following table provides the metrics that are available with the [Classification Threshold Metrics](#) operator.

TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative

Metric	Definition
<b>Recall</b> (proportion of positive cases that were correctly classified)	$\frac{TP}{TP+FN}$
<b>Precision</b> (proportion of positive predicted cases that were correct)	$\frac{TP}{TP+FP}$
<b>F-Measure</b> (for specified $\beta$ , measures the effectiveness of retrieval with respect to a user who attaches $\beta$ times as much importance to recall as precision)	$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta \cdot \text{precision}) + \text{recall}}$

Metric	Definition
<b>False Positive Rate</b> (proportion of negative cases that were incorrectly classified as positive)	$\frac{FP}{FP+TN}$
<b>Cumulative Count</b> (number of cases that were predicted positive for the corresponding confidence threshold)	
<b>Cumulative Count (Percentage)</b> (percentage of cases that were predicted positive for the corresponding confidence threshold)	
<b>Accuracy</b> (proportion of cases (positive and negative) that were correctly classified)	$\frac{TP+TN}{TP+TN+FP+FN}$
<b>Lift</b> (measures how a model performs compared to random guessing for a specific segment. For example, suppose a population has an average response rate of 1%, but a certain model has identified a segment (here the population predicted positive, with <code>confidence_score &gt; confidence_threshold</code> ) with a response rate of 10%. Then that segment has a lift of 10.0. (10%/1%))	$\frac{\text{precision} \cdot \text{total cases}}{TP + FN}$
<b>Kolmogorov-Smirnov (KS)</b> (evaluates the performance of a classification model by measuring the degree of separation between the positive and negative distributions. KS falls between 0 and 1, and the higher the value, the better the model is at separating the positive from negative cases.  If $KS = 1$ , the confidence threshold partitions the population into two separate groups (one group contains all the positives and the other all the negatives)  If KS is close to 0, the model cannot differentiate between positives and negatives.	$ \text{recall} - \text{false positive rate} $

# Principal Component Analysis

PCA, or Principal Component Analysis, is a multivariate technique for examining relationships among several quantitative variables. It uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of uncorrelated variables (principal components).

See [PCA \(DB\)](#) for information about using PCA with a database data source. See [PCA \(HD\)](#) for information about using PCA with a Hadoop data source. See [PCA](#) for information about using PCA with a TIBCO Data Virtualization data source.

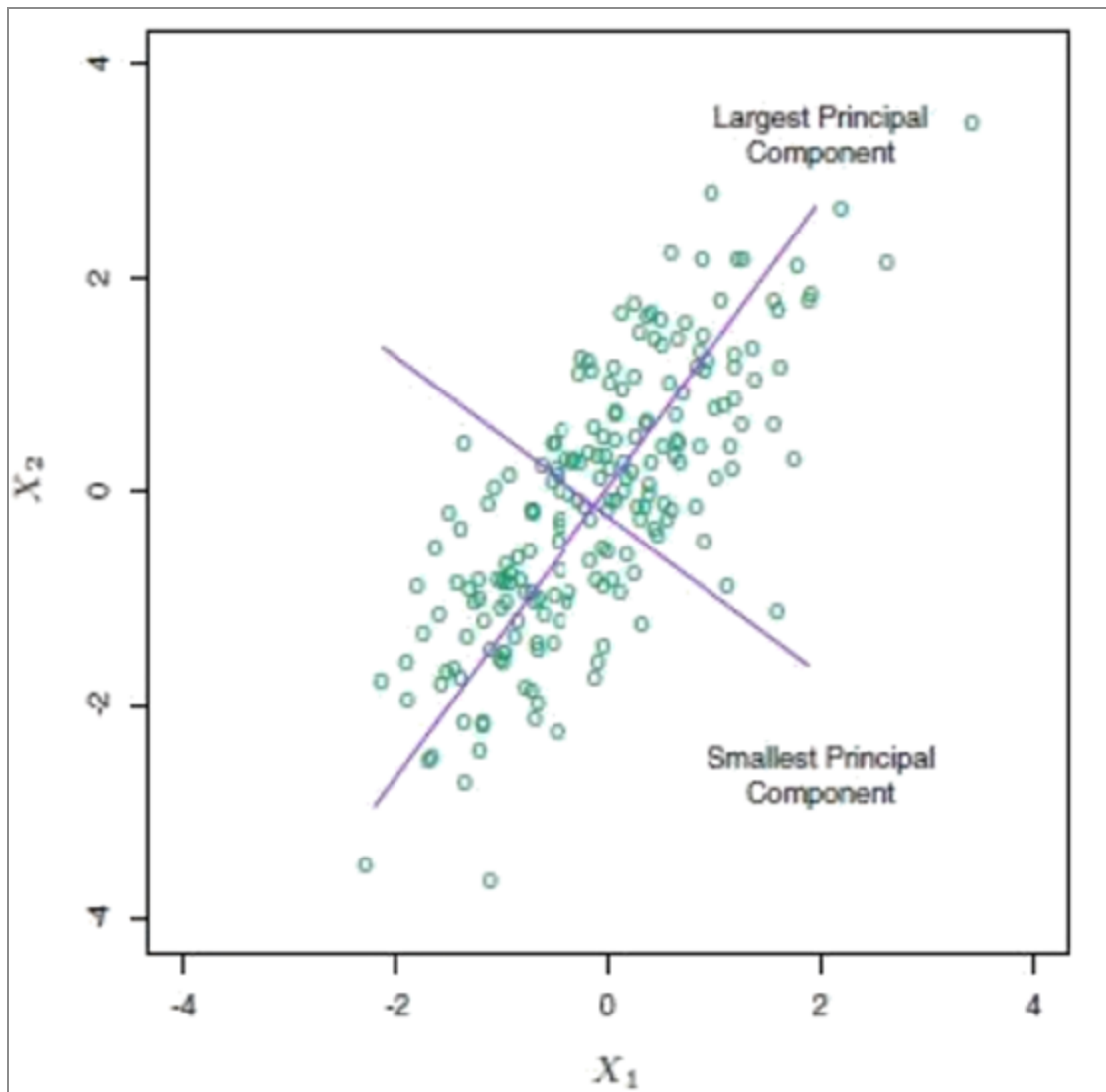
The main reason for using the PCA modeling technique is to create a mapping from the original  $N$  dimensions of a large data set to a new set of a smaller,  $n$  dimensions (where  $n$  is usually a significantly smaller number than  $N$ ).

- The principal components represent the axes (or unit vectors along the axes) of the new dimensions.
- The principal components are ordered, such that the first component contains the most variation from the original data set.
- In other words, if the original data set is mapped onto just the first component, the least information is lost compared to mapping it onto any other component.

Unlike attribute subset selection, which reduces the attribute set size by retaining a subset of the initial set of attributes, PCA "combines" the essence of attributes by creating an alternative, smaller set of variables. The initial data set is therefore projected onto this smaller set.

PCA uses fewer principal components to replace original variables in analysis which often reveals relationships that were not previously suspected and thereby allowing interpretations that would not ordinarily result.

- PCA is then often the first step of the data analysis (as a method for dimensionality reduction), followed by discriminant analysis, cluster analysis, or other multivariate techniques.
- It is thus important to find those principal components that contain most of the information.
- The following figure shows the principal components of some data points in two dimensions. The largest principal component runs along an axis that shows the greatest variation of the data and would represent the first principal component. The smallest principal component shows the least data variation.



Creating PCA workflows varies depending if the analysis is being done against a database or a Hadoop data source, as follows:

- The PCA Operator for database both analyzes the data for principal components, defines the data mapping and also transforms the data (by applying the mapping) to be passed into any other modeling Operator directly.
- The PCA Operator for Hadoop analyzes the data for determining the principal components and defines the data mapping but needs the PCA Apply Operator to transform the data (by applying the mapping) before passing the reduced variable set into any other modeling Operator.

- The PCA Apply Operator for Hadoop applies the data mapping. It can be applied to either new, updated data sets (with the same column names) or to the source training data set (whereas the PCA Operator for database can only be applied to the training data set).

## Support Vector Machine Classification

Support Vector Machine (SVM) is an advanced supervised modeling technique for classifying both linear and nonlinear data. SVM Classification clusters data into the most distant and distinct groups as possible.

SVM has become a more recent default approach to classification problems because it is well suited to very high-dimensional spaces and extremely large datasets. The main idea behind Support Vector Machines is to maximize the distance between the classification datasets and a subset of training data points called support vectors.

- Support vectors are the closest points from every class to the decision boundary.
- The TIBCO Data Science - Team Studio SVM Classification operator handles the classification of both linear and nonlinear data.
- Support Vector Machines, using common Kernel Method transformations, are able to project data that is noisy or not linearly separable, into a higher dimensional linearly separable space.

The TIBCO Data Science - Team Studio SVM Classification operator allows for multi-class classification (as opposed to only binary classification) on both categorical and continuous numerical data. Unlike with the CART, Decision Tree and Random Forest classification operators, there is no tree-like branching structure created with SVMs; rather, the defining details of the Support Vectors that best separates the data are provided.

SVM Classification modeling is often used for complex artificial intelligence applications, such as object detection (computer vision), speech recognition (NLP), handwritten digit recognition, text classification, and bioinformatics (for example, gene classification).

Advantages of Support Vector Machine modeling include:

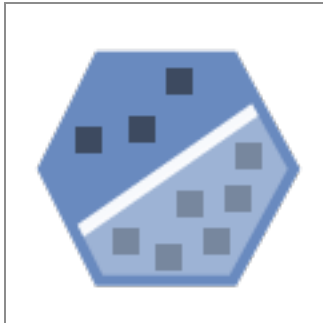
- Ability to handle high dimensionality of large datasets.
- Ability to model complex nonlinear decision patterns.
- Known for being extremely accurate in many cases.

- Less prone to over-fitting than other methods, like a simple Decision Tree or Naive Bayesian classifiers.
- Provides a compact description of learned model by only using the Support Vector data points for each class.

To use SVM in your workflow, see its operator help at [SVM Classification](#) .

## SVM Use Case

The following model demonstrates using an SVM Classification model to assess the varying characteristics of Forest Cover.



## Forest Cover Data Classification

### Datasets

This dataset comes from the UC Irvine site:

<http://archive.ics.uci.edu/ml/datasets/Covertypes>, and the exact version used for this case (with added headers) is [covtype.csv](#) (75.2MB).

The Forest Cover dataset is a standard dataset that data scientists use. The source dataset provides data for determining the type of forest cover to be expected based on soil type, elevation, aspect, slope and other forest variables. It contains 581,012 observations, 54 attribute columns (10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables) and 1 label column (cov\_type). The possible classifications for cov\_type include Spruce/Fir (Type 1), Lodgepole Pine (Type2), Ponderosa Pine (Type 3), Cottonwood/Willow (Type 4), Spruce/Fir and Aspen (Type 5), and Douglas-Fir (Type 6).

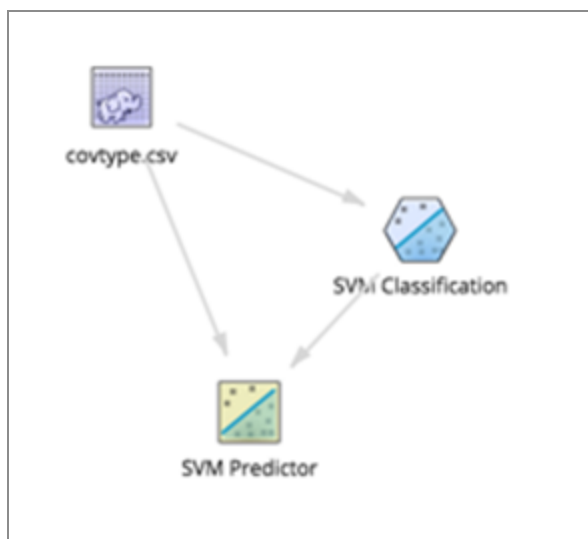
The first few rows and columns are shown below:



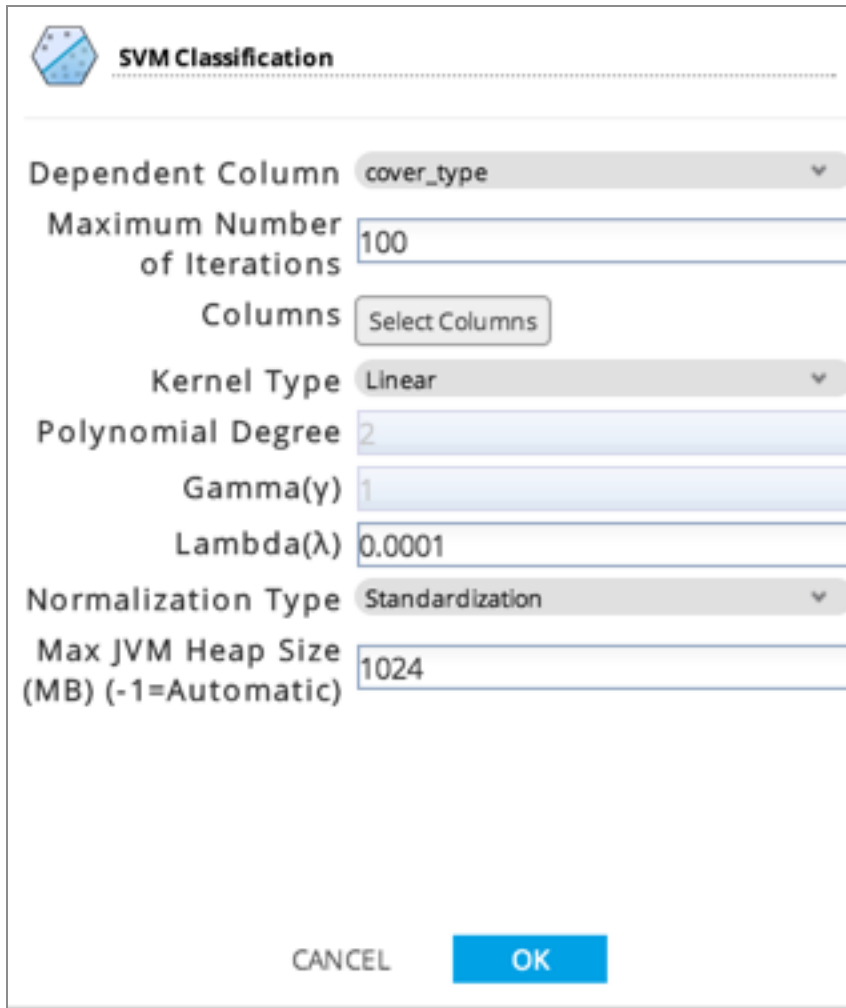
covertype																					
elevation	aspect	slope	hor_dist_to_wet	vert_dist_to_wet	hor_dist_to_rsi	shade_am	shade_noon	shade_pm	hor_dist_to_fire	wilderness_type	soil_type1	soil_type2	soil_type3	soil_type4	soil_type5	soil_type6	soil_type7	soil_type8	soil_type9	soil_type10	soil_type11
2,703	122	30	67	27	3,191	254	201	52	6,123	1	0	0	0	0	0	0	0	0	0	0	0
2,900	45	19	242	20	5,199	221	195	100	4,115	1	0	0	0	0	0	0	0	0	0	0	0
2,703	330	27	30	17	3,141	146	197	184	6,186	1	0	0	0	0	0	0	0	0	0	0	0
2,510	79	14	192	19	891	237	215	106	5,325	1	0	0	0	0	0	0	0	0	0	0	0
2,502	81	7	175	11	912	230	227	129	5,316	1	0	0	0	0	0	0	0	0	0	0	0
2,486	68	5	180	-4	870	225	230	139	5,262	1	0	0	0	0	0	0	0	0	0	0	0
2,860	358	17	175	98	3,705	191	206	151	5,800	1	0	0	0	0	0	0	0	0	0	0	0
2,791	63	10	418	48	2,942	229	221	124	6,606	1	0	0	0	0	0	0	0	0	0	0	0
2,860	31	10	295	98	3,644	218	218	135	5,904	1	0	0	0	0	0	0	0	0	0	0	0
2,567	34	9	190	16	1,136	219	221	138	4,924	1	0	0	0	0	0	0	0	0	0	0	0
2,567	333	1	0	0	1,266	216	237	158	5,079	1	0	0	0	0	0	0	0	0	0	0	0
2,751	88	5	400	30	2,322	228	231	137	6,088	1	0	0	0	0	0	0	0	0	0	0	0
2,767	307	12	30	8	2,796	187	233	186	6,192	1	0	0	0	0	0	0	0	0	0	0	0
2,775	344	13	42	7	2,854	194	219	164	6,177	1	0	0	0	0	0	0	0	0	0	0	0
2,687	20	11	150	39	1,986	212	217	140	4,936	1	0	0	0	0	0	0	0	0	0	0	0
3,109	211	17	424	55	6,089	204	254	181	2,017	1	0	0	0	0	0	0	0	0	0	0	0
2,916	349	10	488	36	4,472	202	223	160	5,502	1	0	0	0	0	0	0	0	0	0	0	0
2,596	51	3	258	0	510	221	232	148	6,279	1	0	0	0	0	0	0	0	0	0	0	0
2,804	139	9	268	65	3,180	234	238	135	6,121	1	0	0	0	0	0	0	0	0	0	0	0
2,612	59	10	247	11	636	228	219	124	6,230	1	0	0	0	0	0	0	0	0	0	0	0
2,612	201	4	180	51	735	218	243	161	6,222	1	0	0	0	0	0	0	0	0	0	0	0
2,489	163	10	30	-4	849	230	243	145	5,486	1	0	0	0	0	0	0	0	0	0	0	0
2,529	68	8	210	-5	666	228	225	130	5,484	1	0	0	0	0	0	0	0	0	0	0	0

## Workflow

An SVM Classification model can be setup quickly to analyze the data, with the following workflow:



The SVM Classification configuration parameters are set as follows.



**SVM Classification**

Dependent Column: **cover\_type**

Maximum Number of Iterations: **100**

Columns: **Select Columns**

Kernel Type: **Linear**

Polynomial Degree: **2**

Gamma( $\gamma$ ): **1**

Lambda( $\lambda$ ): **0.0001**

Normalization Type: **Standardization**

Max JVM Heap Size (MB) (-1=Automatic): **1024**

**CANCEL** **OK**

The column `cover_type` is selected as the dependent column and, for the Select Columns, all 54 other columns are selected as attributes for SVM.

For the first running of the SVM Classification, Linear Kernel Type should be selected.



Kernel Type: **Linear**

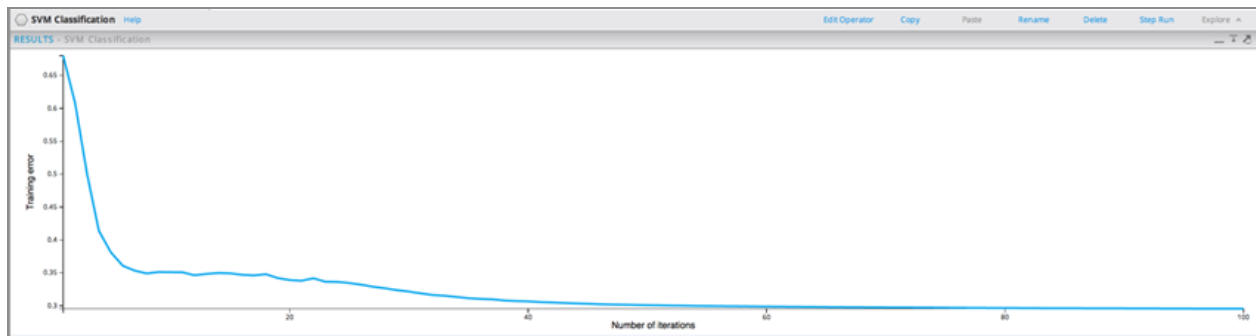
Polynomial Degree: **Gaussian**

Gamma( $\gamma$ ): **Linear**

Polynomial: **Polynomial**

## Results

Running the SVM Classification Model results in the following Training Error curve. After about 40 Iterations the model accuracy stabilizes and it can be considered a fairly good model.

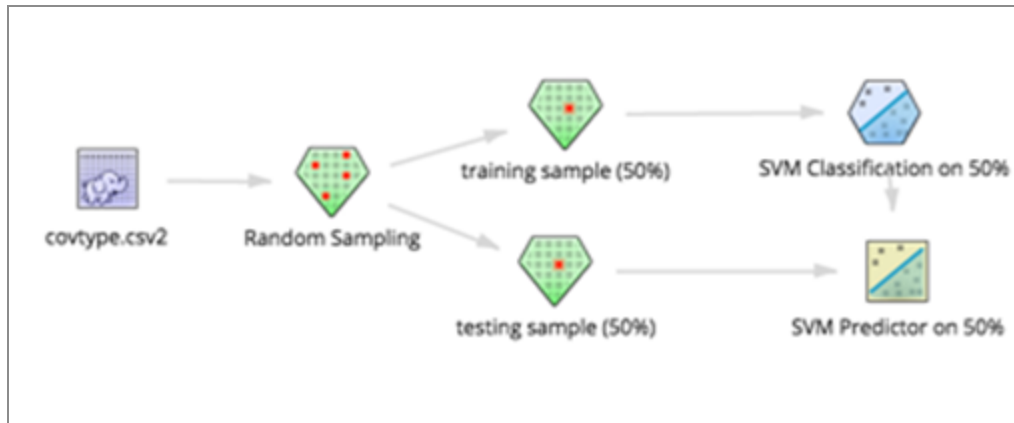


The SVM Predictor output looks as follows (same dataset with 3 more columns at the end):

SVM Predictor														
RESULTS - SVM Predictor														
oil_type32	oil_type33	oil_type34	oil_type35	oil_type36	oil_type37	oil_type38	oil_type39	oil_type40	oil_type41	oil_type42	oil_type43	cover_type	P_cover_type	C_cover_type
0	0	0	0	0	0	0	0	0	0	0	0	6	6	0.9826
0	0	0	0	0	0	0	0	0	0	0	0	6	6	0.9591
0	0	0	0	0	0	0	0	0	0	0	0	6	6	0.9582
0	0	0	0	0	0	0	0	0	0	0	0	6	6	0.9577
0	0	0	0	0	0	0	0	0	0	0	0	6	6	0.957
0	0	0	0	0	0	0	0	0	0	0	0	6	6	0.9584
0	0	0	0	0	0	0	0	0	0	0	0	6	6	0.9989
0	0	0	0	0	0	0	0	0	0	0	0	2	3	0.9989
0	0	0	0	0	0	0	0	0	0	0	0	2	3	0.9988

C\_cover\_type details give the confidence values for each of the 6 cover\_type classes. C\_cover\_type shows the highest confidence value associated with the predicted class P\_cover\_type. The model does not always seem to be predicting the correct type of Forest Cover. Indeed, for some observations the predicted class (P\_cover\_type) is not the same as the real one (cover\_type).

As a next step, the modeler should consider running a Cross-Validation workflow which splits the data into two groups, builds the model based on the first half of the data and then assesses its accuracy on the other half, as follows:



## T-Tests

A t-test is any statistical test of significance that is based on the Student's t- distribution. T-tests are used to determine where two sets of numeric data are drawn from two different samples.

For information about the Student's t-disbribution, see

[https://en.wikipedia.org/wiki/Student%27s\\_t-distribution](https://en.wikipedia.org/wiki/Student%27s_t-distribution)

TIBCO Data Science - Team Studio provides three operators to run three different varieties of t-tests. The following use cases provide sample applications of the three t-tests, which are used to determine where two sets of numeric data are drawn from two different samples.

## Independent Samples T-Test Use Case

The independent samples t-test is used to test whether two groups are significantly different for the same measure.

For more information about the independent samples t-test, see [T-Test - Independent Samples](#).

For an example of this test's usage, see the below sample data from a puppy training program that measures the skills of the puppies before they enter the program and after they leave the program.

Puppy ID	Trainer Name	Score Before Training	Score After Training
123	Rachel	9.5	12.5
124	Jenny	11.6	13.8
125	Jenny	11.1	12.9

Suppose that Rachel argues that Jenny gets assigned more talented puppies. We could test this assumption by running an independent sample t-test on the **Score Before Training** column, and grouping by the **Trainer Name Column**. In this case, the null hypothesis is that the mean **Score Before Training** for Rachel's puppies is not statistically lower than Jenny's puppies. We would reject the null hypothesis if the Lower One Tailed p-value is less than 0.05.

The model assumes that a difference in the mean score of the dependent variable is found because of the influence of the independent variable. Thus, the independent sample t-test is an analysis of dependence. It is one of the most widely used statistical tests. ([source](#))

Using the example of the puppy training program, are Jenny's puppies better at the skills challenge before or after training than Rachel's? In that case, we would select both the **Score Before Training** column and the **Score After Training** column for the **Columns to Test** parameter. We select "Trainer Name" as the **Column to Group By**. Because this test can only be computed on two groups and there may be more than two values in the **Group By** column (in this case, if there were more than two trainers), we have to specify which two groups to run the test on. We fill in the two values in the group by column in the **First Group Value** parameter and the **Second Group Value** parameter. In this case we would enter "Rachel" and "Jenny".



**Caution:** Note that these values must be present in the **Group By** column or the operator fails.

## Paired Samples T-Test Use Case

The paired samples t-test is used to test whether two responses measured on the same statistical unit are significantly different. Mathematically, it is the same as running a single sample t-test on the delta of the two samples for each row against an assumed mean of 0.0.

For more information about the paired samples t-test, see [T-Test - Paired Samples](#).

The difference between the paired sample t-test and the Independent Samples t-test is that in the paired test, the two samples are usually from the same data set, or closely correlated.

Paired sample t-tests can be used for before-and-after studies, such as looking at a student's test scores both before and after a course. For example, see the below sample data from a puppy training program that measures the skills of the puppies before they enter the program and after the program. We might want to ask if training actually improves the puppies' scores. We could do this by running a Paired Samples t-test on the **Score Before Training** and the **Score After Training** columns. Our null hypothesis is that the mean of the difference in the values between the two columns is not statistically different from zero. We would reject the null hypothesis if the Upper One Tailed p-value is less than 0.05.

Puppy ID	Trainer Name	Score Before Training	Score After Training
123	Rachel	9.5	12.5
124	Jenny	11.6	13.8
125	Jenny	11.1	12.9

## Single Sample T-Test Use Case

The single sample t-test is used to test whether a sample population has a significantly different mean from the known population mean.

For more information about the single-sample t-test, see [T-Test - Single Sample](#).

For an example of this test's usage, see the following sample data from a puppy training program that measures the skills of the puppies before they enter the program and after the program.

Puppy ID	Trainer Name	Score Before Training	Score After Training
123	Rachel	9.5	12.5

Puppy ID	Trainer Name	Score Before Training	Score After Training
124	Jenny	11.6	13.8
125	Jenny	11.1	12.9

Suppose that for the above data, we want to ask whether the puppies that enter our training program are scored above average or below average. We know that in general puppies score an average of 10.0 on this test. In this case, we could run a single sample t-test on the **Score Before Training** column against the known mean of 10.0. Formally, we would say that our null hypothesis is that the puppies that enter our training program do not have a statistically higher or lower score on the skills test. We would reject the null hypothesis if the Two Tailed p-value is less than 0.05.

First, we draw a random sample from the population (the connected dataset) and then compare the sample mean with the population mean (the assumed mean here). We can use this to tell whether the sample mean is different from the population (assumed) mean.

For example, we can use this operator if we have election data from a state. We know the population mean (percent of voters who select a certain candidate), but we want to compare that with the sample mean of voters from a certain county. The Single Sample t-test can tell if this difference is statistically significant.

In this example, we can determine whether the puppies score differently than average before and after the training program by selecting the **Score Before Training** and **Score After Training** columns for the **Sample Columns** parameter. We also set the **Assumed Mean** to 10.0, because we know that puppies in general score about 10.0.

## Testing Models for Performance Decay

After models are trained and deployed, differences between the training data and the data to be scored might lead to a degree of deterioration in their predictive performance. This page addresses how to test models to verify that their performance lies within predefined thresholds of acceptability.



## Training a Model

For this example, we use the freely available Bank Marketing data set. The data consists of demographic information, client relationship information, and economic conditions information for 41,188 clients of a Portuguese bank who were targets of a marketing campaign to subscribe to a term deposit (a type of financial product) on offer. The goal is to predict whether the client accepts the offer. This is a classification task.

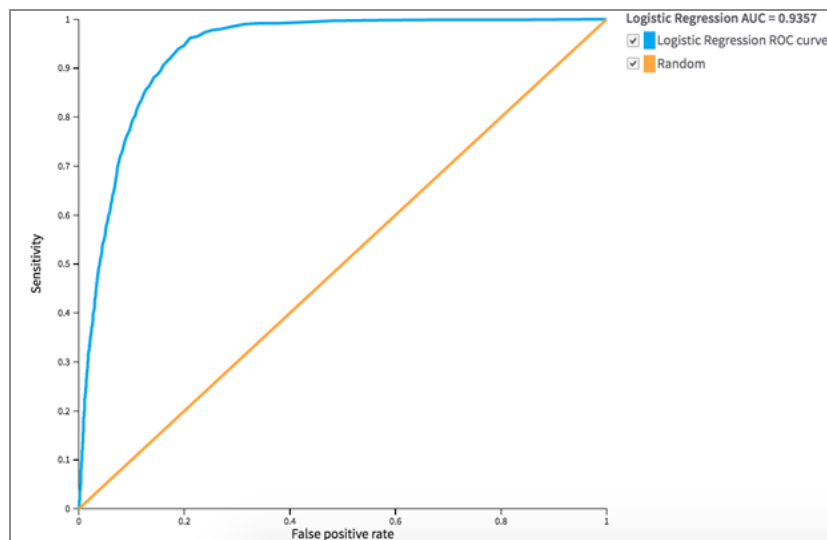
We split the data set into three sets: a training and validation data set that consists of the first 39,000 clients, which represents marketing data from 2007 to 2009 (the dates were verified using the 3M LIBOR column included in the data set). The remaining 2,188 clients were sampled in the year 2010 and are used to test the model for performance decay in operational settings.

Our model training workflow is as follows:



We split the data set that contains marketing data from 2007 to 2009 into a training set and a test set for model validation. We build a logistic regression model on the training data using all the columns in the data set apart from the index and the target variable, and then we validate the performance of the trained logistic regression model on the test data by generating an ROC curve:





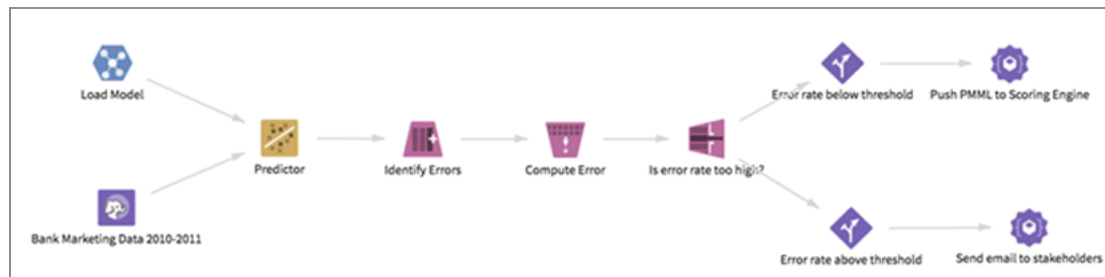
Because the model demonstrates satisfactory performance on the test data, we export the model into the local TIBCO Data Science - Team Studio workspace for use in other operational workflows:

```
[20:11:23] Logistic Regression:persist model saving model to HDFS
[20:11:24] Logistic Regression finished
[20:11:24] Export Model started running .....
[20:11:24] Exporting Logistic Regression
[20:11:24] Export Model finished
[20:11:24] Analytic Flow finished
Click on individual operators to see their output.
```

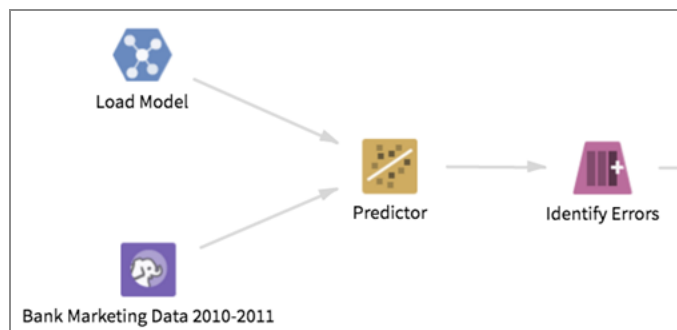
## Testing the Model in an Operational Workflow

Trained models could undergo performance decay in operational settings if the underlying data set that the model was trained on is substantially different than the data set on which it is tested. This is often the case when models are not trained on a sufficiently large and diverse data set that covers a variety of scenarios. Another possibility might be if the newer data set undergoes a non-trivial change in composition as compared to the older data set. For example, the rate of defaults in consumer lending data sets changed dramatically during the financial crisis; predictive models trained on more benign lending environments failed to anticipate this change appropriately.

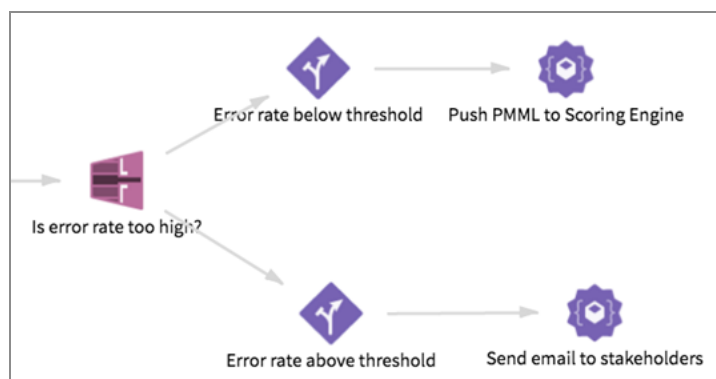
In the following workflow, we create a mechanism to check the performance of a predefined model on recently available bank marketing data. If the model's performance continues to be satisfactory, we push the model to a scoring engine for production usage. If not, we send out an alert email to inform stakeholders of this failure. The overall workflow looks like this:



We begin by loading the saved model from the local TIBCO Data Science - Team Studio workspace. We run the Predictor operator on the loaded model and the latest data set, and check for row-wise prediction errors by introducing a new error variable:



The error variable compares the actual target value (*accepted\_offer*) against the value predicted by the model (*PRED\_LOR*), and is set to 1 if these values are not identical. We compute the average error rate over the entire data set using an Aggregation operator to count the number of errors. We expect the model to exhibit an error rate of less than 5%, and we use a Row Filter operator to check if the error rate is too high (that is, greater than 0.05). Finally, we use Flow Control operators to determine what action to take depending on the performance of the model on this data set:



If the error rate falls above the threshold, the first sub-flow below executes, and an email is sent to stakeholders informing them of a potential drop in model performance. On the other hand, if the error rate falls below the threshold, the second sub-flow below executes,

and the PMML format of the model is pushed to a scoring engine. We run the Predictor operator on the loaded model and the latest data set, and check for row-wise prediction errors by introducing a new error variable.

Note that each of the final actions - Push PMML to Scoring Engine and Send email to stakeholders - are custom operators built using the Custom Operator SDK in TIBCO Data Science - Team Studio, and are not available as part of the core TIBCO Data Science - Team Studio offering.

## Prediction

Prediction operators follow Modeling operators. Each modeling algorithm operator must be assembled with an input data source for analysis and a Prediction operator for running the model against a new source in order to return a predicted dataset.

There are two general types of [Prediction Operators](#).

- Classifier
- Predictor

### Classifier Operators

A Classifier calculates a probability of each possible value for the independent variable as well as the value with the highest probability. In addition, it provides confidence values for the model.

Classifiers are terminal operators within database Workflows (no other operator can follow them).

### Predictor Operators

A Predictor calculates the following:

- For classification algorithms: the value with the highest probability
- For numeric, regression algorithms: the predicted value

Predictors are not terminal operators within a workflow (other operators can follow them for further analysis of the returned results).

## Prediction and Modeling Operator Pairings

In general, the Classification Modeling operators are followed in a workflow by a Classifier operator, and the Regression Modeling operators that predict a value are followed by a Predictor Operator.

However, a Predictor operator can also be applied to Classification models in order to return the value with the highest probability.

The few exceptions are for modeling techniques such as Collaborative Filtering, Time Series, EM Cluster, PCA, and LDA that have their own specialized Predictor operator (shown in the following table). See [Prediction Operators](#) for more information about the classifiers and predictors.

The following table summarizes which Prediction operators can follow Modeling operators (to apply the appropriate modeling algorithms).

Modeling operator	Next operator	Exception
ARIMA Time Series	n/a	Time Series Predictor
Alpine Forest	Classifier or Predictor	n/a
Alpine Forest Regression	Predictor	n/a
Association Rules	n/a	n/a
Collaborative Filter Trainer	n/a	Collaborative Filter Predictor, Collaborative Filter Recommender
Decision Tree	Classifier or Predictor	n/a
Elastic Net Linear	Predictor	
Elastic Net Logistic	Classifier or Predictor	
Generalized Linear	Predictor	

Modeling operator	Next operator	Exception
Regression Models		
Gradient Boosting Classification	Classifier	n/a
Gradient Boosting Regression	Predictor	n/a
K-Means	Classifier*	n/a
Linear Regression	Predictor	n/a
Logistic Regression	Classifier or Predictor	n/a
Naive Bayes	Classifier or Predictor	n/a
Neural Network	Classifier or Predictor	n/a
PCA	n/a	PCA Apply*
SVM Classification	Classifier or Predictor	n/a

\*PCA Apply operator and the Classifier for K-Means are used only within Hadoop workflows.

## Pearson's Chi Square Operations

TIBCO Data Science - Team Studio provides two Chi Square operators for predicting on a Hadoop data set.

## Independence Test

You can use [Chi Square, Independence Test](#) for determining whether categorical columns are statistically independent of a categorical dependent variable column.

For example, if we have a dataset of user churn that includes the factors gender and browser type, we could use the Independence Test operator to determine if there is a significant statistical relationship between gender and churn, or between browser and churn. The Chi Square test of independence can be used as the basis for performing inferential statistics on data with many categorical variables, or as an exploratory data step, such as to determine what factors to include in a logistic regression or decision tree. We leverage the [Mllib implementation](#) of Pearson's Chi Square test of independence.

This operator also includes the option to use a Fisher's exact test. The Fisher's exact test is a similar test for statistical significance to the Chi Square test with slightly different assumptions. The Fisher's exact test is recommended when sample sizes are small, (cell sizes less than five), and can be calculated only on 2 x 2 tables. The operator is based on the formula described in the article [Fisher's exact test](#).

To examine whether the frequencies of events differ from a normal distribution or from a known distribution, consider using the [Chi Square, Goodness of Fit](#) operator.

To examine statistical significance between quantitative variables, consider using one of the T-Test operators. (See [T-Test - Independent Samples](#), [T-Test - Paired Samples](#), or [T-Test - Single Sample](#) for more information.)

Here is a Fun Fact: Fisher developed the test to analyze the result of an experiment to test his friend Dr. [Muriel Bristol](#)'s assertion that she could tell the difference between a cup of tea in which the milk was poured first and a cup of tea with milk added second.

## Goodness of Fit

You can use [Chi Square, Goodness of Fit](#) to test for goodness of fit for a distribution.

In this case, the Chi Square test is performed on two vectors of the frequency of events: the vector of observed frequencies and the vector of expected frequencies. The null hypothesis is that the frequencies in each cell (frequency of each event occurring) are equal in the observed and expected distribution. This test differs from the test of independence, because it assumes that the degrees of freedom are equal to the number of possible events minus one.

This test should be used to test observed outcomes against some known theoretical distribution; therefore, we designed the operator so that it accepts the observed and

expected frequency data in two different datasets, because these datasets might not be the same length. (Most likely, the expected vector is already aggregated with sum of frequencies for each distinct event, while the observed vector comes from some real data with each observation as a row.) Our implementation leverages the [Chi Square Test in Spark's MLlib](#). We use the absolute frequencies of the observed and expected datasets as the vectors input for the test.

## Spark Node Fusion

You can use Spark Node Fusion to have multiple operators in a single Spark job (also called "Spark context"). This allows the job to run faster because it is not recreating a new job and persisting results to HDFS at each analytical step.

**i Note:** Spark Node Fusion is applicable only to workflows that use Hadoop.

Regardless of your workflow's size or the number of operators it contains, performance during runtime is crucial. Being able to specify the use of node fusion on existing workflows, and then revert to the previous setting, must be easy to do. You can accomplish this using the **Use Spark** property. For more information, see [Convert to Spark/Revert to Non-Spark](#).

When a workflow with Spark operators is run through the job scheduler, the results are not made visible to the user, because doing so would make the job run much more slowly. If you want to view results anyway, set **Store Results** to **true** before you run the job.

The following operators have been updated to use Spark Node Fusion. Prior to the release of TIBCO Data Science - Team Studio version 6.4, these operators typically used MapReduce or Pig execution frameworks.

- [Aggregation \(DB\)](#)
- [Collapse](#)
- [Column Filter \(DB\)](#)
- [Correlation \(DB\)](#)
- [Distinct \(HD\)](#)
- [Join \(DB\)](#)

- [Classification Modeling with Naive Bayes](#)
- [Normalization \(DB\)](#)
- [Null Value Replacement \(DB\)](#)
- [Pivot \(DB\)](#)
- [Row Filter \(DB\)](#)
- [Set Operations \(DB\)](#)
- [Variable \(DB\)](#)

## Viewing Results for Individual Operators

When a workflow that contains Spark operators is run, the results are not shown by default. This enables the workflow to run faster. You can see the results for individual operators available for Spark Node Fusion under the following conditions.

- When you run the workflow from the top menu, the workflow preview page, a scheduled job, or an API call, you can see results as follows:
  - If **Store Results** is set to false, only summary information about column names and data types is displayed in the operator preview.
  - If **Store Results** is set to true, preview output is displayed in the **Results** panel.
- When you run the workflow by clicking **Step Run** from the lower menu, or from the right-click option, you can see results as follows:
  - If **Store Results** is set to either true or false, you can see the preview output in the **Results** panel. This can be useful for debugging while you are building workflows.

For more information, see [Convert to Spark/Revert to Non-Spark](#).

## Specialized Tools

Team Studio provides specialized sets of operators for such areas as natural language processing (NLP) and financial fields.

The NLP operators are included with the legacy operators in Team Studio.



# Natural Language Processing Tools

Natural Language Processing tools include a dictionary builder, text featurization, unsupervised text mining, and Latent Dirichlet Allocation (LDA) training and model evaluation tips. Examine the NLP use case to learn to train a model to classify documents into categories, given a training set of documents in both of the categories.

Information about NLP operators is available at [NLP Operators](#).

## Using the Results of Text Featurizer

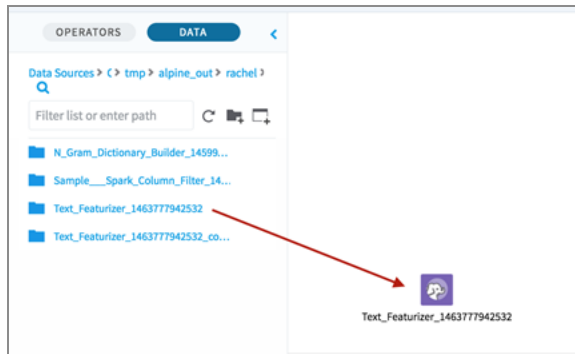
If you click **Yes** for the Text Featurizer's **Use N-gram Values as Column Names** parameter, use this process to access the full result data.

### Procedure

1. After you run the operator, go to the **Summary** tab.
2. On the left panel, click the **Data** tab.
3. Click your data source.
4. Navigate to the folder indicated in the **Summary** tab. (To make this quicker, you can copy the first part of the path, and then paste that into the search bar of the **Data** tab.)
5. Press Enter, and then select the folder whose name corresponds to the results file (in this case, **Text\_Featurizer\_1453777942532**).
6. Drag that folder onto the canvas.

### Result

TIBCO Data Science - Team Studio interprets that folder as a fully configured Hadoop data set that can be connected to other operators.



## Unsupervised Text Mining

You can perform unsupervised text mining to analyze collections of unstructured documents using the LDA (Latent Dirichlet Allocation) operators.

**Important:** Older versions of LDA and LDA Predictor for Database have been removed as of version 6.2. Instead, use the new LDA operators, [LDA Trainer](#) and [LDA Predictor](#).

In LDA, each document can be viewed as a [mixture](#) of various topics. This is similar to [probabilistic latent semantic analysis](#) (pLSA), except that in LDA, the topic distribution is assumed to have a [Dirichlet prior](#). In practice, this results in more reasonable mixtures of topics in a document.

For example, an LDA model might have topics that can be classified as CAT\_related and DOG\_related. A topic has probabilities of generating various words, such as "milk," "meow," and "kitten," which can be classified and interpreted by the viewer as "CAT\_related". Naturally, the word cat itself has high probability given this topic. The DOG\_related topic likewise has probabilities of generating each word: "puppy," "bark," and "bone" might have high probability. Words without special relevance, such as "the," have roughly even probability between classes (or can be placed into a separate category). A topic is not strongly defined, neither semantically nor epistemologically. It is identified on the basis of supervised labeling and (manual) pruning on the basis of their likelihood of co-occurrence. A lexical word might occur in several topics with a different probability, but with a different typical set of neighboring words in each topic.

Each document is assumed to be characterized by a particular set of topics. This is akin to the standard bag-of-words model assumption, and makes the individual words exchangeable.

## Use Cases for LDA

These topic distributions can be used in the following ways.

- **Clustering:** Topics are cluster centers and documents are associated with multiple clusters (topics). This clustering can help organize or summarize document collections.
- **Feature generation:** LDA can generate features for other ML algorithms to use. As mentioned above, LDA infers a distribution over topics for each document; with  $k$  topics, this gives  $k$  numerical features. These features can then be plugged into algorithms such as Logistic Regression or Decision Trees for prediction tasks.
- **Dimensionality reduction:** Each document's distribution over topics gives a concise summary of the document. Comparing documents in this reduced feature space can be more meaningful than comparing them in the original feature space of words.

We leverage the MLLib LDA algorithm (Spark version 1.5.1) with the Online Variational Bayes optimizer (Original Online LDA paper: [Online Learning for Latent Dirichlet Allocation - Hoffman, Blei and Bach, NIPS, 2010](#)). This algorithm uses iterative mini-batch sampling: it processes a subset of the corpus on each iteration, and updates the term-topic distribution adaptively. This makes it memory-friendly, especially on large number of documents or vocabulary. This algorithm is also preferable to the EM algorithm (also available in MLLib LDA), because it can optimize the parameters ( $\alpha$ ) of the Dirichlet prior over the topic mixing weights for the documents, which can create better topics.

LDA Trainer and LDA Predictor work on Hadoop datasets, and you can use them in tandem with our other NLP operators to build complex workflows.

## LDA Training and Model Evaluation Tips

When you are using the LDA Predictor and LDA Trainer, following these guidelines can produce more meaningful results.

- **Select the right n-grams to use:** ensure that the N-Gram Dictionary and the n-gram selection method used are relevant (by specifying/updating a customized [Stop Words](#) in the [N-gram Dictionary Builder](#), and changing the n-gram selection method):
  - Filter common stop words and any other set of words not relevant for your use case.
  - Ensure that you don't allow very high-frequency words to overpower the rest of

the corpus.

- You likely don't want very infrequent words either.
- Run the LDA long enough (it can require many iterations to obtain relevant topics)
- Try different parameters (number of topics, etc.) and evaluate log perplexity on a held-out sample.
- Building a good LDA model often requires many iterations and human feedback. Indeed, log perplexity is good for relative comparisons between models or parameter settings, but its numeric value doesn't really mean much, and it's not correlated to human judgment.
  - Inspect the topics: Look at the highest-likelihood words in each topic. Do they sound like they form a cohesive topic, or just some random group of words?
  - Inspect the topic assignments. Hold out a few random documents from training and see what topics LDA assigns to them. Manually inspect the documents and the top words in the assigned topics. Does it look like the topics really describe what the documents are actually about?
- Look at the density of words of the topics: if you have a topic with weak/low densities for its constituent words, it is most likely a weak topic.

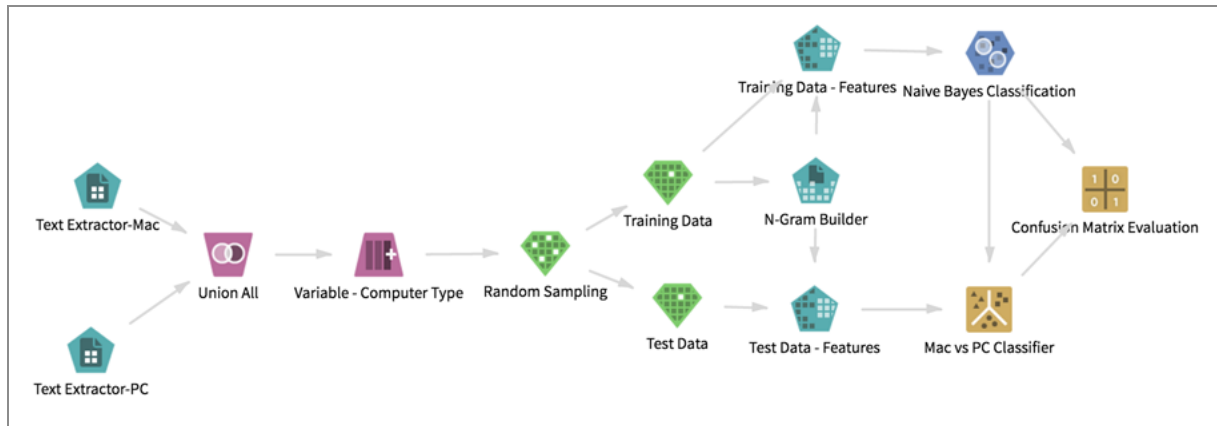
## NLP Use Case

You can use NLP Use Case to train a model to classify documents into categories, given a training set of documents in both of the categories.

### Mac versus PC Article Classification

For our use case, we began with a folder of articles about Macintosh hardware and a folder of articles about IBM PC hardware. We built a classifier that could determine whether a new article is posted by the Mac user group or the PC user group.

Below is the resulting workflow. In this article, we describe how we built the workflow to get meaningful information.



## Datasets

The data came from the 20 Newsgroups Dataset, which contains messages from popular Usenet groups. We used text from two newsgroup folders.

- `comp.sys.mac.hardware`
- `comp.sys.ibm.pc.hardware`

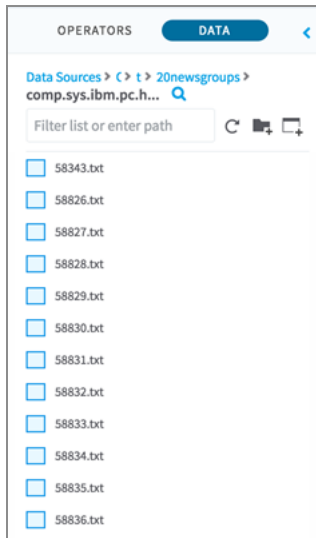
Each of these folders contained 1000 text files, each of which contained one message. The data was in plain text format. We parsed out the relevant information using the available NLP operators.

## Workflow

### Load the Data to HDFS

The data must be in HDFS to be accessible to TIBCO Data Science - Team Studio. To add the data, we connected to the cluster using SSH, and then copied the folder containing the text files into a location accessible by TIBCO Data Science - Team Studio.

In the data source, we named one folder `comp.sys.mac.hardware` and we named a second folder `comp.sys.ibm.pc.hardware`. Each folder contained the 1000 files of the form `<number>.txt`:



## Data Cleaning

We put our imported text data into a tabular format, and then generated features for the classification.

The cleaned data needed to have the following fields.

- The *computer\_type* categorical variable, which is used to indicate whether the message is from the Mac group or the PC group.
- Numeric features (from the NLP operators) such as n-gram counts, which are passed to TIBCO Data Science - Team Studio classification operators.

To train different models, we created two random samples from the dataset. One sample is the training set, and the other sample is the test set.

The data transformation procedure is as follows.

## Procedure

1. We used the [Text Extractor](#) operator on both of the folders. This operation consolidated the messages in each folder into one dataset, with one row per article. The resulting dataset included several columns about the parsing process.

For example, we ran the text extractor on the `comp.sys.mac.hardware` directory to get the following output.



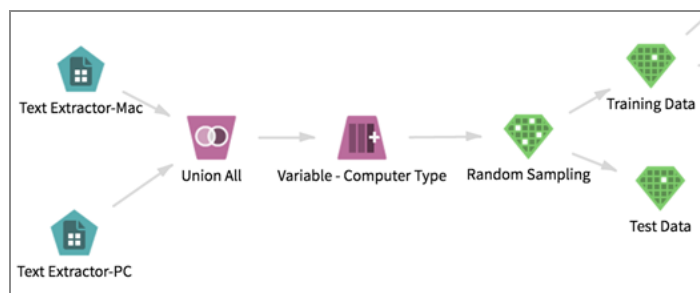
newsgroup path and tagged it with the easier-to-read *computer\_type* variable using a Pig expression in the [Variable \(DB\)](#) operator.

Remember that the two newsgroups are named `comp.sys.mac.hardware` and `comp.sys.ibm.hardware`. We care only about the `mac` and `ibm` part of the file path. To extract that part, we used the Pig expression `SUBSTRING(file_path,28,31)`

✔ **Tip:** You can use TIBCO Data Science - Team Studio to design a similar ETL process. In this example, articles all in the same folder can have names that represent their types (for example, `Mac_1.txt`), or you can join a dataset with the path and some metadata about each article to the output of the Text Featurizer.

4. We used the [Random Sampling \(DB\)](#) operator to break the data into two samples: the training sample and the test sample. We split our samples into 80% for our training sample and 20% for our test sample.
5. We used two [Sample Selector](#) operators to split out random samples into the training set and the test set. We connected the Random Sampling operator to our two Sample Selector operators to achieve this.

At this point, our workflow looked like the following. It was ready to apply the model.



## Modeling

We used the training and test sets to create some features from the data using NLP operators, and then we ran a classification and confusion matrix to see how well the model performed.

1. On the training data, we used the [N-gram Dictionary Builder](#) to create a



dictionary of words used to create the features. We specified tokenization and parsing options. The dictionary builder creates a dictionary of the n-grams (words or phrases) in the training corpus. When you featurize a dataset for either training a model or predicting, you must use the same N-gram Dictionary Builder to ensure that you are creating features from the same n-grams.

The N-gram Dictionary Builder also defines how to tokenize the document. This tokenization must also be consistent across the training and test data. You can learn more about the configuration options for the N-gram Dictionary Builder in its help topic, but in this use case, we used default options.

The output of the N-gram Dictionary Builder displays all the n-grams in all the documents, how frequently those n-grams appear, and in how many documents they appear.

ngram	size_of_ngram	total_count_in_corpus	number_of_documents
wrote to	2	1	1
restrict themselves	2	2	2
pb100 message-id	2	1	1
,fekvh6	1	1	1
is here	2	1	1
lisa it	2	6	6
ronald h	2	1	1
expensive though	2	1	1
system7 date	2	1	1
ncache	1	1	1
unpoke as	2	1	1
conflicts i	2	1	1
and 900	2	1	1
's shi"	2	1	1
dept strathclyde	2	1	1

It also conveys statistics about all of the documents.

Results - N-Gram Builder		
<a href="#">Dictionary (Output)</a> <a href="#">Corpus Statistics</a> <a href="#">Summary</a>	<b>Document Statistics</b>	
	Total Number of Documents	1568
	Total number of N-Grams	731856
	Total number of Unique N-Grams	176027
	<b>Total N-Gram Statistics</b>	
	Total Number of Unigrams	366712
	Total Number of Bigrams	365144
	<b>Total Unique N-Gram Statistics</b>	
	Total Number of Unique Unigrams	32439
	Total Number of Unique Bigrams	143588

The N-gram Dictionary Builder cannot be connected to TIBCO Data Science - Team Studio operators other than the Text Featurizer. However, it does write tabular output to HDFS. To use this output, navigate to the location you specified to store the results, and then drag that result file onto the workflow canvas. (Another example of something you could do with the dictionary results here is run it through a sorting operator to see which n-grams appeared in the most documents.)

2. We used the [Text Featurizer](#) operator to select the most commonly used 500 words from the dictionary. Then, for each message, we scored how many times each word appeared.

To provide features for both the training set and the test set, we did the following.

- a. Connected a Text Featurizer operator to both the training data and the N-Gram Dictionary Builder.
- b. Connected a Text Featurizer operator to the test data and the N-Gram Dictionary Builder.
- c. Configured each operator with the following settings:

Setting	Configuration
Text Column	text_content
N-Gram Selection Method	Appear in the Most Documents
Maximum Number of Unique N-grams to Select (or Feature Hashing Size)	500
For Each N-Gram and Document Calculate	Normalized N-Gram Count
Use N-Gram Values as Column Names	No
Storage Format	CSV
Output Directory	@default_tempdir/tsds_out
Output Name	@operator_name_uuid
Overwrite Output	true
Advanced Spark Settings Automatic Optimization	Yes

For each tokenized article, we counted the number of times each of the 500 selected n-grams appeared, and then normalized the counts against the number of tokens in the document. Thus, we computed a metric of how frequently the word appeared relative to the length of the article. By selecting the *computer\_type* column in the **Columns To Keep** field, we used it as the dependent variable in the classification algorithms.

The result of each Text Featurizer is a dataset where each row represents one article, and each column is a numeric feature.

The first column(s) are the columns from the original dataset we selected to keep. In this case, we selected to keep only one: the *computer\_type* column. The next four columns contain some generic statistics about each article.

- *number\_of\_tokens* - the number of tokens (words) in the article.
- *normalized\_number\_tokens* - the number of tokens in the article, normalized against the average number of tokens in the all of the articles.
- *number\_unique\_tokens* - the number of unique words in the article.
- *normalize\_number\_unique\_tokens* - the above, normalized against the average of the corpus.

computer_type	number_of_tokens	normalized_number_tokens	number_unique_tokens	normalized_number_unique_tokens	ngram1_count_normalized	ngram2_count_normalized	ngram3_count_normalized	ngram4_count_normalized
Item	238	0.00061532	174	0.0013892	0.0040336	0.00420168	0.00420168	0.00420168
Item	163	0.0004221	102	0.0002419	0.0104091	0.00613497	0.00613497	0.00613497
Item	235	0.00060894	170	0.0004032	0.00425532	0.00425532	0.00425532	0.00425532
Item	86	0.00022271	74	0.00219402	0.01162791	0.01162791	0.01162791	0.01162791
Item	166	0.00042987	125	0.00370612	0.0060241	0.0060241	0.0060241	0.01204819
Item	343	0.00088623	216	0.00640417	0.00291345	0.00291345	0.00291345	0.0058309
Item	136	0.00030219	97	0.00287395	0.00735294	0.00735294	0.00735294	0.00735294
Item	204	0.00032628	139	0.00421212	0.00490196	0.00490196	0.00490196	0.00490196
Item	599	0.01351117	334	0.00960275	0.00308835	0.00168945	0.00168945	0.00168945
Item	145	0.00037549	101	0.00299404	0.00689655	0.00689655	0.00689655	0.00689655
Item	91	0.00023965	80	0.00237132	0.01098901	0.01098901	0.01098901	0.01098901
Item	302	0.00076206	197	0.00584084	0.00662352	0.00331126	0.00331126	0.00331126
Item	136	0.00030219	95	0.00281665	0.00735294	0.00735294	0.00735294	0.00735294
Item	508	0.00131352	237	0.0070268	0.00303701	0.0013665	0.0013665	0.0013665

The next 500 columns each represent the normalized count of one of the most commonly used n-grams. *ngram1\_count\_normalized* is the normalized count per document of the most common n-gram in the training corpus.

*ngram1* is still unknown. To find out what it is, we looked at the **N-Grams to Column Names** section of the results.

For this example, note that *ngram1* is the token from.

N-Gram Label	N-Gram Value	N-Gram Size	Total N-Grams in Corpus	Total Number of Documents
ngram1	from	1	2,693	1,612
ngram2	subject	1	1,656	1,606
ngram3	date	1	1,693	1,606
ngram4	cantaloupe	1	1,832	1,606
ngram5	message-id	1	1,611	1,606
ngram6	newsgroups	1	1,665	1,606
ngram7	.srv.cs	1	2,518	1,606
ngram8	apr	1	1,732	1,606
ngram9	path cantaloupe	2	1,606	1,606
ngram10	path	1	1,628	1,606

Now we had a dataset with numeric columns that represent the frequency of the n-grams in each document, and one categorical column with the category of that document. This example uses the simplest classification algorithms, but now we can integrate the text data with nearly all of the analytic power of TIBCO Data Science - Team Studio. We can apply any of the TIBCO Data Science - Team Studio transformation operators to further clean and process the data. We can explore the content of the corpus using the exploration operators, such as Summary Statistics. We can cluster the documents into new, previously unknown categories. We can see if the presence of some n-grams depends on others with any of the regression algorithms. We can even build a Touchpoint on top of the trained classification algorithm to classify new documents.

3. [Classification Modeling with Naive Bayes](#) is one of the most common algorithms used for text classification. We applied Naive Bayes to the featurized dataset, building a classifier that can classify a new document as either from the Mac or IBM user group.

We connected the featurized dataset to the Naive Bayes operator, and then we used the document category as the dependent column and the text features (for example, *Ngram1\_normalize\_count*) or (*Doc\_count*) as independent variables.

Optionally, we could export this model using [Export](#). Instead, we used it in a workflow to predict the subject of new documents.

## Predicting and Classifying

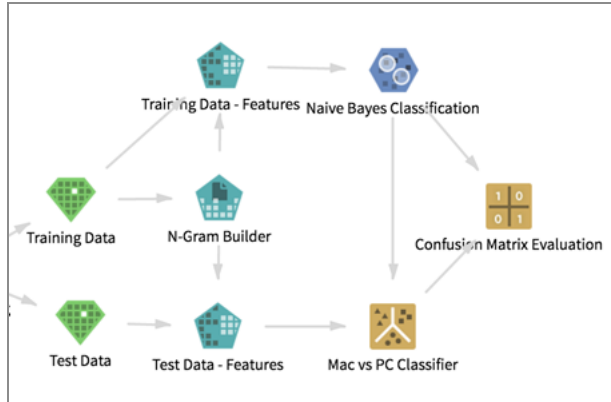
To use a predictor or a classifier with NLP data, you must clean the test data in the same way you cleaned the training data. This means using the same n-gram dictionary that was used on the training data, and using a Text Featurizer that is configured exactly the same way as the one used on the training set. You can copy and paste the operator to get another with the same configuration.

The following three operators can help you with your prediction and classification.

1. **Text Featurizer** (for the test data): Connect the Text Featurizer to the Test Data and the N-Gram Dictionary Builder that has run on the training data.
2. **Classifier (DB)**: Connect a classifier (or a predictor) to the featurized test dataset and to the model. The Classifier operator takes the input from the Naive Bayes operator and the data in the test set to apply the model.
3. **Confusion Matrix**: Use a confusion matrix or other evaluator to evaluate the results of the model. The Confusion Matrix operator feeds the information from the Naive Bayes operator and the Classifier to the Confusion Matrix, where it evaluates the model's performance and reports the precision, recall, and accuracy of the model.

**Important:** The validation is not accurate unless you run the data through the Text Featurizer, and then split it into the test and training samples. The n-gram features are determined based on the entire corpus, rather than just the training set.

After featurizing the test data, we connected it to a classifier and then to a confusion matrix, so that we could evaluate the model.



By looking at the results of the confusion matrix, we could see how well the model performed.

	Predicted (ibm)	Predicted (mac)	Class Recall
Observed (ibm)	179	4	0.9781421
Observed (mac)	1	190	0.9947644
Class Precision	0.9944444	0.9793814	Accuracy: 0.9866310

It predicts the correct class 98% of the time.

## Test Corpus Parsing

The TIBCO Data Science - Team Studio N-gram Dictionary Builder can parse a text corpus, create tokens, and then parse into all possible n-grams (combinations of sequential tokens).

In the following example, the text from Dr. Seuss would be treated as two "documents", with one per line.

```
one fish, two fish,  
red fish, blue fish
```

This would be parsed into the following n-grams:

Length 1 (Unigrams)	one, fish, two, red, blue
Length 2 (Bigrams)	one fish, two fish, red fish, blue fish
Length 3 (Trigrams)	one fish two, red fish blue

The output of the N-gram Dictionary Builder operator would look like the following.

ngram	size_of_ngram	total_count_in_corpus	number_of_documents
one	1	1	1
one fish	2	1	1
fish	1	4	2
...			

Keep in mind that each line in the file refers to one document.

**Important:** The data output of this operator is written to HDFS as a delimited file, but is not recognized by TIBCO Data Science - Team Studio as a tabular dataset. This is because it is a special n-gram dictionary type that is only recognized by the [Text Featurizer](#) operator.

Although you cannot connect a transformation operator such as [Summary Statistics \(DB\)](#) to this operator directly, you can go to the location of the results on HDFS (specified in the **Summary** tab of the results pane), drag the file(s) onto your workflow, and use that as input to other TIBCO Data Science - Team Studio operators. However, keep in mind that, if you use this method, the files might be stored in multiple parts.

## Using Pig User-Defined Functions

The script in the Pig Execute operator can reference a Pig UDF.

The list of acceptable Pig UDFs can be found here: [Pig User Defined Functions](#)

The UDF must be defined in a JAR file located in ALPINE\_DATA\_REPOSITORY/PigUDFJars/. The JAR file is automatically registered when running a workflow.

Within the script, use the UDF as in any Pig script.

- For example, to use the [DataFu](#) include the DataFu JAR in ALPINE\_DATA\_REPOSITORY/PigUDFJars/. In the script include:  

```
define Sessionize datafu.pig.sessions.Sessionize('30m');
```

 Then use Sessionize as desired.

## DateTime Input Values

Pig Execute Operators can process and perform operations on DateTime input values.

DateTime input fields are passed into the Pig Execute Operator as chararray strings, specified in an ISO-acceptable format. The Pig Script template reads the input DateTime format and generates two intermediate columns: one that contains the Date\_as\_string, and one that contains the Date\_as\_datetime object.

The Date\_as\_datetime object is created by using the ToDate Pig Function that references the input format.



Any valid DateTime function can be used within the Pig Script, as listed here: [Apache Pig DateTime Functions](#)

The Result portion of the Pig Script specifies the datetime output values that the Pig Script should generate. In this example, the Day of the Date is returned as well as the date as a string.

Define Pig Script

AND OR NOT
MATCHES IS NULL IS NOT NULL
== != > < >= <=
Pig Syntax

```

1 A = FOREACH alpine_pig_input_1 GENERATE ToDate(Date, 'MM/dd/yy') AS Date_as_datetime, Date AS
Date_as_string, Close, Return;
2 Result = FOREACH A GENERATE GetDay(Date_as_datetime), Date_as_string, Close;
3 STORE Result INTO '/tmp/alpine_out/@user_name/@flow_name/pigexec_0';

```

Alias	Column name
tss.csv.xls (alpine_pig_input_1)	
	Date
	Close
	Return

CANCEL
OK

However, because some Pig Functions have issues handling null values, using the TIBCO Data Science - Team Studio alternatives for certain functions is recommended. Just append 'Alpine' to the method name; for example, DaysBetween becomes DaysBetweenAlpine.

Pig DateTime Function	Alternative Pig DateTime Function
AddDuration	AddDurationAlpine
SubtractDuration	SubtractDurationAlpine
MillisecondsBetween	MillisecondsBetweenAlpine
SecondsBetween	SecondsBetweenAlpine
MinutesBetween	MinutesBetweenAlpine

Pig DateTime Function	Alternative Pig DateTime Function
HoursBetween	HoursBetweenAlpine
DaysBetween	DaysBetweenAlpine
WeeksBetween	WeeksBetweenAlpine
MonthsBetween	MonthsBetweenAlpine
YearsBetween	YearsBetweenAlpine

## Setting Up Notebooks for Python Execute

Python Notebooks are a very flexible tool. To incorporate your work within a Python Execute operator in a visual workflow, here are some best practices for preparing your Notebooks.

### Suggestions for Setting Up Notebooks for Python Execute

The automatically generated tag Ready For Python Execute indicates that you can attach a Notebook to a Python Execute operator. To achieve this attribute, you need the following.

- At least one input or output specified in the notebook with the argument `use_input_substitution = True` or `use_output_substitution = True`.
- The notebook input(s) argument `execution_label` are distinct and use exclusively one of the following strings: "1", "2", or "3".
- All Inputs and output defined with `use_input_substitution = True` must come from the same type of data source (Hadoop or Database).

Run the Notebook in its entirety from the toolbar by clicking **Cell > Run All**. Do NOT run cells out of order - this can cause issues with metadata information that is passed to the Python Execute operator. After running all cells, save the notebook before attempting to run the notebook in a workflow.

You can create an input for substitution using the following steps:

### Procedure

1. Associate a dataset with the workspace.
2. In the notebook toolbar, click **Data**.
3. Select the dataset, and then click **Import**.

This generates a cell with functions to read the data (for example, `cc.read_input_file` or `cc.read_input_table`, depending on whether you selected a database table or a file in HDFS).

4. Change `use_input_substitution=False` to `use_input_substitution=True`.
5. Add a named argument to the function, called `execution_label`. This argument should have the string value "1", "2", or "3", and is used to identify the inputs in the visual workflow Python Execute operator. It should look something like the following.

```
df_account=cc.read_input_table(table_name='account', schema_
name='demo', database_name='miner_demo',use_input_
substitution=True, execution_label="1")
```

6. Run the generated cell. This fetches the data, creates a dataframe in the Notebook, and saves this information as a valid input for the Python Execute operator in the visual workflow editor.

To create an output for substitution.

## Procedure

1. Use the `cc.write_output_file` or `cc.write_output_table` function, depending on whether you want to write a table or a file. You can see the function arguments by executing `help(cc.write_output_table)` in your Notebook.
2. Run the cell. This writes the dataset and saves this information as a valid output for the Python Execute operator in the visual workflow editor.
3. Before using a Python Execute operator, ensure that you have cleaned your Notebook to remove any interactive code (for example, the `help()` function).

A notebook is invalid if any of the following conditions exist.

- It has duplicate execution labels.
- It has execution labels that are NOT "1", "2", or "3".

**i Note:** These values are strings enclosed in double quotations.

- It mixes HDFS and DB inputs with substitution.

**i Note:** Mixing inputs without substitution is allowed

- It has more than one output with substitution.

## Team Studio Commander Functions

TIBCO Data Science - Team Studio Commander is a set of functions available in a Notebook that connects the Python environment to the TIBCO Data Science - Team Studio environment. This includes features such as adding data associations, importing files, and outputting results. To access detailed documentation, run `help(cc)` from a Notebook.

## R Execute

You can use R Execute in a seamless extension of a TIBCO Data Science - Team Studio Workflow to include any existing R-coded model.

### Features

You can write the R script and run it against an input dataset stored in any database supported by TIBCO Data Science - Team Studio, without writing SQL queries, knowing the SQL syntax differences between Oracle, PostgreSQL, and so on.

It allows TIBCO Data Science - Team Studio operators to precede and/or follow any valid R model. These can include things such as column and row filters (pre or post R), histograms (post R), and the input data sources themselves (database table or Hadoop delimited file).

You can write the R script and run it against an input dataset stored in Hadoop data store supported by TIBCO Data Science - Team Studio (for example, HDFS in case of CDH4/5, Pivotal HD 2.x, Hortonworks and Apache, and MapR file system in case of MapR), without having to deal with the complexity and security issues of accessing these data stores directly.

**! Important:** This design also addresses the issue of having to access different clusters at once, with different Hadoop and therefore HDFS/MapR FS versions.

The data is pulled into R only if the input data frame is specified, saving time if you want to pull the data from somewhere else (for example, the web, bypassing the input), or want to ignore the input and simply generate output (for example, Monte Carlo simulation, random number generation, and so on). See below for usage of the input data frame functionality in your R script.

If you specify an output data frame, it can be stored in any database or Hadoop data store supported by TIBCO Data Science - Team Studio without you being concerned about how to do it. See below for usage of the output data frame functionality in your R script.

**i Note:** You do not need direct database permissions. You do not need to know how to write SQL queries to create a new table or to insert data. You do not need to know how to do efficient batch insertions and manage transactions.

## Requirements

You must have the R Connector installed on a server in your TIBCO Data Science - Team Studio deployment. For more information, see your system administrator.

## Important

The R Execute Operator is a batch operator.

- You cannot interactively run an R command or function, see what the interpreter generates, and then have the state of the session stored in memory in order to perform the next experiment. In this sense, the R Execute operator differs from an R session in an R shell/interpreter/REPL (read-evaluate-print loop) or the RStudio IDE.
- Note that if you run an R script that expects interactive user input, the operator hangs, because R waits for user input. Any function that is interactive must not be used in the script to be run by R Execute. This includes the interactive mode of `install.packages`, which requires you to interactively specify the repository from which to get the package. In case of `install.packages`, this can be avoided by specifying the repository explicitly - this prevents R from waiting for interactive user input.

```
install.packages(pkgs = c('dplyr'),  
  repos = c('http://cran.cnr.berkeley.edu/'))
```

- If you inadvertently include an interactive function in the code, the operator hangs as

described above, but the flow can be stopped with the **Stop** link in the TIBCO Data Science - Team Studio user interface, which stops the entire workflow. If you expect the result to return quickly, but the operator is stuck for a very long time, stop the flow and check for interactive user code.

- You must write valid R code and see the results of its execution in the console output and the resulting data frame.
- Try the R code on a very small input dataset first to test the logic.
- Writing code incrementally, not creating an output data frame (see below), and printing to the console means you can debug the code quickly, unless it has already been debugged in RStudio.
- You can visualize the console output in TIBCO Data Science - Team Studio; however, TIBCO Data Science - Team Studio does not allow you to visualize plots using R code. For plot visualization, TIBCO Data Science - Team Studio users should pass the output from the R Execute operator to a subsequent operator, or link with Spotfire or Tableau. R Execute generates helpful messages to identify issues, including R code syntax errors, data type mismatches, and so on. See [R Execute Error Messages](#) for more information.

Clean your data before you run the R Execute operator. By design, the R Execute operator does not do any data cleaning; it uses the specified data as is. If you know that your data is not clean (for example, the header is included in the data, there are missing or incorrect values, and so on), use other TIBCO Data Science - Team Studio operators or R commands to clean your data.

The TIBCO Data Science - Team Studio product can be extended for use with the R language and environment for statistical computing and graphics (see <https://www.r-project.org/>) through use of the R Connector for TIBCO Data Science - Team Studio (the "R Connector"), which is subject to free open-source software license terms and is available on [GitHub](#).

The R Connector is not part of the TIBCO Data Science - Team Studio product and therefore not within the scope of your license for the product. Accordingly, the R Connector is not covered by the terms of your agreement with the TIBCO Data Science - Team Studio product, including any terms concerning support, maintenance, or warranties. Download and use of the R Connector is solely at your own discretion and subject to the free open-source license terms applicable to R Connector.

Similarly, the R language and environment for statistical computing and graphics and related packages (the "R Engine and R Packages") are not within the scope of your license

for the product. Accordingly, the R Engine and R Packages are not covered by the terms of your agreement with the TIBCO Data Science - Team Studio product, including any terms concerning support, maintenance, or warranties. Download and use of any R Engine or R Packages are solely at your own discretion and subject to the free open-source license terms applicable to the same. TIBCO bears no responsibility for the accuracy of R algorithms, bugs in R Packages, the stability of the R Engine, the logic of the user's R code, or the license implications of R itself. Please note that the R Engine is licensed under [GNU General Public License](#) (GPL), versions 2 and 3. R Packages are licensed under various licenses, including, but not limited to, GPL, Affero GPL (AGPL), BSD 2-clause and 3-clause licenses, the Artistic License, and the MIT license ([see here for details](#)). If you have any questions about such open source licenses, consult a software license lawyer for advice.

## R Execute Error Messages

Because R Execute is an operator that executes the user's arbitrary code, many things can go wrong. The R Execute operator includes error messages to help you identify the source of the problem.

### Syntax errors in the R script

A common syntax mistake in an R script can result in an error when you run the R Execute operator.

TIBCO Data Science - Team Studio uses R to parse the R script before executing it. An unclosed parenthesis is reported in an error that looks like the following.

```
Execution of 'R Execute' failed.  
Error details: Error in parse(text=rawScript) ::2:1: unexpected symbol  
1: print(summary(alpine_input) 2: alpine_output^
```

The status results provide an option to download the SQL log for further information.

### Logical errors in the user's R script

A logical mistake in an R script can result in an error when you run the R Execute operator.

For example, dividing the string "1" (which is read as a character vector by R) by the number 0 is a logical error.

```
Execution of 'R Execute' failed.
Error details: Error in "1"/0: non-numeric argument to binary operator
```

The status results provide an option to download the SQL log for further information.

## Input data size limitations

Pulling too much data can result in an error when you run the R Execute operator.

If your script requests data from the database or from Hadoop that is beyond the limit set by the system administrator, R Execute reports an error like the following, suggesting that you sample the data or change the limit.

```
Execution of 'R Execute-1' failed.
Error details: The input data exceeds the size limit of 16 MB.
Please either take a smaller data sample or ask your administrator to
increase the permitted size.
```

The status results provide an option to download the SQL log for further information.

## Output data size limitations

If a returned dataset exceeds the allowed limit, R reports a size limit error.

In this case, R fails to return the dataset to Team Studio and reports a size limit error, similar to the following.

You can increase the size limit, or you can sample the output data before assigning the dataset to the `alpine_output` variable.

```
Execution of 'R Execute-1' failed.
Error details: Error in failOnOversizedDF(alpine_output, 1.6e+07):
alpine_output is 32000776 bytes
but the maximum allowed object size to be returned is 16000000 per
object. Sample the output data or
increase the Akka message sizes on the R server.
```

The status results provide an option to download the SQL log for further information.



## Network issues

A connection failure or a partition between nodes can result in a network error.

The R server should run on a different computer than the computer where Team Studio is installed. In the case of a network disconnection or partition between nodes, R Execute jobs are ended on the Team Studio computer, and the jobs are also ended on the R server side (which directly handles the R workers).

The following message can be displayed if the R server Java Virtual Machine (JVM) terminates. It can also be displayed if there is a firewall between the Team Studio node and the R server node that blocks access on the ports chosen before Team Studio and the R server were started.

After the administrator ensures that both Team Studio and the R server are working correctly, and that the network is up and the firewall is properly configured, you should be able to retry the flow.

```
Execution of 'R Execute-4' failed.
Error details: Connection to R server lost while executing operator.
Please have your administrator check your network connection and confirm
that your R server is running.
```

The status results provide an option to download the SQL log for further information.

## Missing output reference in the R script

If the output properties in the R Execute Operator are set differently than that the script expects, an output error can occur.

If the option **Pass Output Table/Pass Output File** is set to **Yes** and **Results Table Structure/Results FileStructure** is provided, and if there is no reference to the 'alpine\_output' in the R script, then R does not generate the output data frame to be returned to Team Studio), and the following message is displayed.

You can set **Pass Output Table/Pass Output File** to **No** (in which case the R Execute operator becomes terminal), or you can edit the R script to generate the 'alpine\_output' data frame object in R.

```
Execution of 'R Execute' failed.
Error details: 'Pass Output Table' set to 'Yes' but there is no 'alpine_
output' data frame in the script.
```

The 'alpine\_output' data frame determines what is stored in the database for use in subsequent operators.

The status results provide an option to download the SQL log for further information.

## Column name or type mismatches

Column names, column types, and column count specified in the R Execute operator must match those returned by the R script.

If **Pass Output Table/Pass Output File** is set to **Yes** and **Results Table Structure/Results File Structure** is provided, but the output data frame returned by R has different column names, column types or column count than that specified in **Results Table Structure/Results File Structure**, then the R Execute operator displays an error reporting that part of the flow is ended. This error report informs you that what the specified inputs for the subsequent operator is at odds with what R actually returned.

You can either change the names or types in **Results Table Structure/Results File Structure** to match those that R returns (and adjust the subsequent operators), or you can change the R script so that the R output matches **Results Table Structure/Results File Structure**.

For example, if you expect an integer column (for example, for database storage), but R returns a floating-point number, then the column type can be changed to a FLOAT or DOUBLE, or you can cast the floating-point values for that data frame column to an integer using the R function call `as.integer()`.

```
Execution of 'R Execute' failed.
Error details: There is a mismatch between the column names in the
'Results Table Structure' and those returned by R.
Expected column names: [humidity, outlook, temperature, wind]
Returned column names: [humidity, outlook, play, temperature, wind]
```

or

```
Execution of 'R Execute' failed.
Error details: There is a mismatch between the column types in the
'Results Table Structure' and those returned by R.
Column 'play': Expected INTEGER, returned VARCHAR.
Please correct types returned by R to match design time types or change
design time types to match.
```

The status results provide an option to download the SQL log for further information.

## Type coercion error

R sometimes performs type coercion, which you might not expect. Type coercion can cause problems because the R Execute operator expects R to return a data frame if you assign the *alpine\_output* variable in the R script.

One common case of type coercion is when you select a single column from the data frame. The column is coerced from a data frame to a "numeric" type, which is returned to Team Studio as an array of doubles. This causes a `java.lang.ClassCastException` because data frames are returned as `java.util.Map`, but the R coercion results in a numeric vector being returned, rather than a data frame. You must do type casting in R to undo the type coercion. However, calling `as.data.frame` on a numeric vector from a projected column associates it with a wrong name, so you must rename the column after you create the data frame, as in the following example.

```
> x <- rnorm(1:10)
> y <- rnorm(1:10)
> xy <- data.frame(x=x, y=y)
> head(xy)
      x      y
1 0.43765921 -0.5160450
2 0.69197730  0.1938183
3 0.08869384 -1.2843015
4 0.99896046 -1.2151482
5 -1.08242907  0.1109868
6 -0.55645055 -1.3973622
> class(xy)
[1] "data.frame"
> xOnly <- xy$x
> head(xOnly)
[1] 0.43765921 0.69197730 0.08869384 0.99896046 -1.08242907 -
0.55645055
> class(xOnly)
[1] "numeric"
> xOnlyAsDataFrame <- as.data.frame(xy$x)
> head(xOnlyAsDataFrame)
      xy$x
1 0.43765921
2 0.69197730
3 0.08869384
4 0.99896046
5 -1.08242907
```

```

6 -0.55645055
> names(xOnlyAsDataFrame) <- "x"
> head(xOnlyAsDataFrame)
      x
1  0.43765921
2  0.69197730
3  0.08869384
4  0.99896046
5 -1.08242907
6 -0.55645055
> class(xOnlyAsDataFrame)
[1] "data.frame"
>

```

## Workflow Operator Reference

The TIBCO Data Science - Team Studio workflow operators are categorized by specific function or type.

Each workflow operator represents a unit of computation or a data source. Operators can perform tasks such as data loading and transformation or execute more advanced tasks such as training and evaluating machine learning models.

In addition to working with existing operators, you can create custom operators that allow you to incorporate new or existing custom analytics code.

## Legacy Operators

The legacy operators provide the tools to load data, transform the data, explore and sample the data, and perform modeling, prediction, and validation operations on the data.

## Data Operators

Data Extraction (Load Data) operators connect to a data source.

Load operators, also known as Data Extraction operators, create connections to data sources to bring data into the workflow. Load operators also provide options to convert the source data formats between database and Hadoop.

## Copy Between Databases

Provides a mechanism for copying data from a table on one database server to another database server.



### Information at a Glance

Parameter	Description
Category	Load Data
Data source type	DB
Send output to other operators	Yes
Data processing tool	No

### Input

A database table or a database operator that emits tabular output.

### Bad or Missing Values

Missing values in the original database table are transferred as null values to the table in the target database.

### Restrictions

If you are using Google BigQuery as a data source, and your data set is very large, the copy operation can time out unexpectedly. See your system administrator to increase the value of the `timeout` property if you encounter this problem.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Copy To</b>	The database to copy the data set to.
<b>Destination</b>	The schema to copy the data set to.
<b>Table Name</b>	The name for the new table.  Default value: <b>alp@user_id_@flow_id_dbtodb_0</b> .
<b>If Table Exists</b>	The desired behavior if the table exists in the destination location. <ul style="list-style-type: none"> <li>• <b>Append</b> - Append the data to the end of the table.</li> <li>• <b>Drop</b> (the default) - Drop the table and replace it with the new one.</li> <li>• <b>Error</b> - Produce an error.</li> <li>• <b>Skip</b> - Skip this table.</li> </ul>
<b>Result Table Structure</b>	The table structure for the resulting table to which to copy. When the dialog is opened, if the resulting table exists, the schema is read from the resulting table. If the resulting table does not exist, the schema is determined from the source table. Data types for the resulting table are inferred for the target database, but the data types can be edited.
<b>Fetch Size</b>	The number of rows at a time to fetch from the source table. Depending on the features of the source database JDBC driver, changes to the fetch size can improve performance.  Default value: <b>20000</b> .
<b>Batch Size</b>	The number of rows to insert into the resulting table at one time.  Default value: <b>5000</b> .

Parameter	Description
<b>Note:</b> Larger batches can improve speed, but they use more memory.	

## Output

### Visual Output

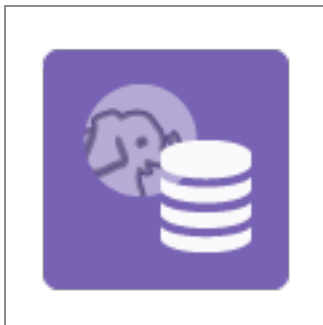
A preview of the rows of the resulting copied data on the target database server.

### Data Output

A data set that corresponds to the destination table.

## Copy To Database

Provides a mechanism for copying data from Hadoop into a relational database.



## Information at a Glance

Parameter	Description
Category	Load Data
Data source type	HD
Send output to other operators	Yes
Data processing tool	Sqoop

Copy to Database usually creates a new table in the database in which to store the copied data. It uses the column and data type information associated with the Hadoop file to determine the table's structure. If the destination table named by the user already exists, the operator can be configured to respond in one of the following ways.

- Drop the table first.
- Append the new data.
- Skip the operation.
- Produce an error.

The copy process can be run in parallel mode or simple mode.

## Input

A Hadoop file or a Hadoop operator that produces a stored file (that is, any Hadoop operator whose **Store Results** option is set to **true**).

## Restrictions

The operator works only in Simple mode with TIBCO Data Virtualization. For Parallel mode, see the supported databases in [Sqoop documentation](#).

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Copy to</b>	The data source connection through which the data is copied.
<b>Destination</b>	The schema into which the data is copied.
<b>Table Name</b>	The name of the table in which the data is stored. For more information about the variables used in the default table name, see <a href="#">Workflow Variables</a> .  Default name: <b>alp@user_id_@flow_id_todb_0</b> .



Parameter	Description
<b>If Table Exists</b>	<p>The option to use if the destination table specified by <b>Table Name</b> already exists.</p> <ul style="list-style-type: none"> <li>• <b>Drop</b> (the default) - Drop the table first.</li> <li>• <b>Extend</b> - Append the new data.</li> <li>• <b>Error</b> - Report an error and stop execution of the workflow.</li> <li>• <b>Skip</b> - Skip the operation.</li> </ul>
<b>Copy Mode</b>	<p>The copy method.</p> <ul style="list-style-type: none"> <li>• <b>Parallel</b>(the default) - Copy in parallel using the underlying Sqoop technology.</li> <li>• <b>Simple</b> - Copy using the batch processing copy process.</li> </ul>
<b>Number of Copy Tasks</b>	<p>The number of parallel processes to use for the Sqoop parallel processing copy mode. For <b>Parallel</b> copy mode only.</p> <p>Default value: <b>4</b>.</p>
<b>Advanced Parameters</b>	<p>Click <b>Configure</b> to display the <a href="#">Advanced Parameter Configuration dialog</a> and set the advanced configuration parameters for parallel copy with Sqoop.</p>

## Outputs

### Visual Output

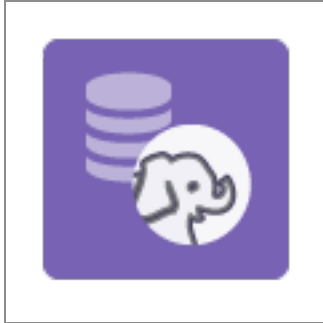
A preview of the rows of the resulting copied data.

### Data Output

A data set that corresponds to the destination table. You can use the output of the Copy to Database operator as the input to any operator that accepts database tables or views.

## Copy To Hadoop

Provides a mechanism for copying relational data into a Hadoop cluster.



## Information at a Glance

Parameter	Description
Category	Load Data
Data source type	DB
Send output to other operators	Yes
Data processing tool	Sqoop

The Copy to Hadoop operator usually creates a new file on the Hadoop file system for storing the copied data. The column and data type information associated with the database table is used to associate a structure with the Hadoop file.

If the destination file named by the user already exists, the operator can drop the file first, skip the operation, or produce an error. The operator might also be able to append the new data to the existing file, but only if the Hadoop cluster supports this operation.

The copy process can be run in parallel mode or simple mode.

## Input

A data set or an operator that produces results in a database.

## Restrictions

Pig and Sqoop are used to copy your data to Hadoop. Pig does not accept some characters in column names. If a column name contains a character that is not [ A-Z a-z 0-9 \_ ] +, the non-conforming character is replaced with an underscore character ( \_ ) to create a valid

column name. If the data contains columns that could cause a name collision, an underscore and an integer (\_1, \_2, and so on) are appended to these column names. For example, consider a table with columns named column@a and column#a. In this case, the columns are renamed column\_a and column\_a, and then differentiated as column\_a\_1 and column\_a\_2.

Additionally, Pig requires the first character of a column name to be a letter. If it is not, the column name is prepended with "a," for TIBCO Data Science - Team Studio. Therefore, a column named /column is renamed a\_column.

**i Note:** Because Pig and Sqoop both contain a bug that can cause errors when column names contain backslashes, you should not use backslashes in your column names.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Copy to</b>	The data source connection through which the data is copied. Default value: <b>CDH5</b> .
<b>Destination</b>	The folder into which you want the data copied.  Click <b>Choose File</b> to browse the existing Hadoop file structure and specify the destination location within it.
<b>File Name</b>	The name of the file in which the data is stored. Default value: <b>tohd_0</b> .
<b>If File Exists</b>	If the destination table specified by <b>File Name</b> already exists, select one of the following options. <ul style="list-style-type: none"> <li>• <b>Drop</b> (the default) - Drop the table first.</li> <li>• <b>Extend</b> - Append the new data.</li> <li>• <b>Error</b> - Report an error and stop execution of the workflow.</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Skip</b> - Skip the operation.</li> </ul> <p>The operator also can append the new data to the existing file, but only if the Hadoop version supports this operation.</p>
<b>Copy Mode</b>	<p>The copy method.</p> <ul style="list-style-type: none"> <li>• <b>Parallel</b> (the default) - Copy in parallel using the underlying Sqoop technology.</li> <li>• <b>Simple</b> - Copy using the batch-processing copy process.</li> </ul>
<b>Number of Copy Tasks</b>	<p>For <b>Parallel</b> copy mode only. The number of parallel processes to use for the Sqoop parallel-processing copy mode.</p> <p>Default value: <b>4</b>.</p>
<b>Divide Up Work By</b>	<p>The database column to use for saving the data in the Hadoop file system structure. You must specify one column.</p>
<b>Advanced Parameters</b>	<p>Click <b>Configure</b> to display the <a href="#">Advanced Parameter Configuration dialog</a> and set the advanced configuration parameters for parallel copy with Sqoop.</p>
<b>Fetch Size</b>	<p>The number of entries to read from the database at once. This is equivalent to the <code>--fetch-size</code> Sqoop parameter.</p> <p>Default value: <b>20000</b>.</p>

## Output

The output of the Copy to Hadoop operator can be used as the input to any operator that accepts Hadoop files.

### Visual Output

A preview of the rows of the resulting copied data.

### Data Output

A Hadoop file that corresponds to the destination file.

## Dataset

Connects a database table or view, allowing the data to be incorporated into the workflow. Dataset is a source operator.



### Information at a Glance

Parameter	Description
Category	Load Data
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

The database table or view can subsequently be used in the following projects.

- A data-mining algorithm
- A prediction algorithm
- A statistical analysis

### Input

None. Dataset is a source operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Data source</b>	The database connection for access to the database where the dataset (Table/View) resides. The data source select list is populated with the database data sources associated with the current workflow.  Default value: <b>miner_demo</b> .
<b>Schema Name</b>	The schema name of the dataset (Table/View). The list of schemas is populated with the schemes found in the selected <b>Data Source</b> .  Default value: <b>public</b> .
<b>Table Name</b>	The name of the data set (Table/View). The list of tables is populated with the tables found in the selected <b>Schema Name</b> .  Default value: <b>test</b> .

## Output

### Visual Output

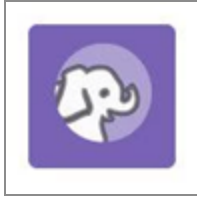
None.

### Data Output

A data set that corresponds to the specified database table/view.

## Hadoop File

Specifies a file or files stored on your Hadoop data source, allowing the data to be incorporated into the workflow.



## Information at a Glance

Parameter	Description
Category	Load Data
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce

The Hadoop data can subsequently be used in data mining algorithms, prediction algorithms, and statistical analyses.

TIBCO Data Science - Team Studio automatically handles files that are stored in compressed format with the gzip or deflate codec.

## Input

None. Hadoop File is a source operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Data Source</b>	The Hadoop connection for access to the Hadoop file system where the file

Parameter	Description
<b>Name</b>	resides.
<b>Hadoop File Name</b>	<p>The path and name of the file. Click <b>Choose File</b> to display the Hadoop File Explorer dialog, and to browse the Hadoop file structure and select the file location.</p> <p><b>Note:</b> To process multiple files using wildcard characters, see <a href="#">Selecting Groups of HDFS files</a>.</p>
<b>Hadoop File Format</b>	<p>The source file format. If the file name extension is available, the file format automatically defaults to a setting based on that extension. You can override it manually.</p> <p>The following file formats are available.</p> <ul style="list-style-type: none"> <li>• Avro</li> <li>• Parquet</li> <li>• Text file</li> </ul>
<b>Hadoop File Structure</b>	<p>Click <b>Hadoop File Structure</b> to display the Configure Columns dialog. The file type determines the dialog display and configuration options. The following file types are supported.</p> <ul style="list-style-type: none"> <li>• <a href="#">Configure Columns: Text Files</a></li> <li>• <a href="#">Configure Columns: XML and JSON Files</a></li> <li>• <a href="#">Configure Columns dialog</a></li> </ul> <p>The Hadoop file structure settings specify the delimiters, columns, and data types in the Hadoop file.</p> <p>For more information, see <a href="#">Configure Columns dialog</a>.</p>

## Output

### Visual Output

A preview of the data output.



## Data Output

A Hadoop file.

## Hive Table

Connects a Hive Hadoop table, allowing the data to be incorporated into the workflow. Hive Table is a source operator.



## Information at a Glance

Parameter	Description
Category	Load Data
Data source type	HD
Send output to other operators	Yes
Data processing tool	n/a

The Hive Hadoop table can subsequently be used in data mining algorithms, prediction algorithms, and statistical analyses.

## Input

None. Hive Table is a source operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Data Source Name</b>	The database connection for access to the Hive database where the table resides. The data source select list is populated with the Hive Hadoop data sources associated with the current workflow.
<b>Hive Database Name</b>	The name of the database for the Hive table. The list of databases is populated with the databases in <b>Data Source Name</b> .
<b>Hive Table Name</b>	The name of the Hive table. The list of tables is populated with the tables in <b>Hive Database Name</b> .

## Output

### Visual Output

A preview of the data output.

### Data Output

A data set that corresponds to the specified Hive table.

## Import Excel (DB)

Imports an Excel workbook sheet (or a portion of the sheet) as a database table.



## Information at a Glance

Parameter	Description
Category	Load Data
Data source type	DB
Send output to other operators	Yes <sup>1</sup>
Data processing tool	n/a



**Note:** The Import Excel (DB) operator is for database data only. For Hadoop data, use the [Import Excel \(HD\)](#) operator.

The Excel workbook can be stored in the current workspace.

Formula cells, styles, dates, currencies, percentages, and so on are supported, and are parsed as numeric values. Non-tabular data such as images and pivot tables are skipped. Hidden columns and protected sheets are parsed as normal.

## Input

Import Excel (DB) is a source operator. No inputs are required.

### Bad or Missing Values

Blank cells or empty cells are converted to null values.

## Restrictions

Excel files are read on the TIBCO Data Science - Team Studio server. Depending on the memory available on your instance, loading very large Excel files on this server might require a large amount of memory and cause out-of-memory issues. For more information, see Apache POI limitations at <https://poi.apache.org/spreadsheet/limitations.html>.

---

<sup>1</sup>The full output schema is not available until you step-run the operator. After the operator is run, the output schema automatically updates, and subsequent operators are either validated or turn red depending on the structure of the output data.

TIBCO Data Science - Team Studio uses the configuration parameter `custom_operators`, set in the `alpine.conf` file, to avoid loading files that are too large. If the Excel file is bigger than this limit, it does not load and an error message is displayed. The default value is 30.0 (MB). The administrator of your TIBCO Data Science - Team Studio instance can modify the default value.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Data Source (DB)</b>	The data source where the output should be stored (and where the <b>Column Metadata Table</b> is stored).
<b>Schema</b>	The schema where the output should be stored (and where the <b>Column Metadata Table</b> is stored).
<b>Chorus Workfile</b>	The Excel workbook stored in the current workspace. Only workbooks with the .xls, .xlsx, and .xlsm extensions are displayed.
<b>Sheet Number</b>	The number of the sheet to extract (the first sheet is 1).
<b>Top Left Corner Cell</b>	<p>The cell address that defines the top-left cell of the data portion to extract in the selected sheet (for example, B10).</p> <p><b>Note:</b> If the header is included in the sheet, skip it in the selection. For example, if the header starts in A1, the top-left corner cell entered should be B1. This is because the header is read separately in the <b>Column Metadata File (CSV)</b> parameter, from a CSV file that includes both names and column types.</p>
<b>Right Cut-Off Column Letter</b>	The optional column letter that defines where the portion of data to extract should be cut at the right. If not specified, the data extraction is cut at the last defined cell of the first row selected (the row number in <b>Top Left Corner Cell</b> ).

Parameter	Description
	<p><b>Note:</b> If this parameter is not specified, and if the first row selected is empty or not defined, an error is displayed.</p>
<b>Bottom Cut-Off Row Number</b>	The optional row number that defines where the portion of data to extract should be cut at the bottom. If this is not specified, the data extraction is cut at the last defined row in the sheet.
<b>Column Metadata Table</b>	<p>The database table that defines the header for the output. It should be an empty table with only the column names (the types are defined at table creation, either when uploading via the user interface, or manually).</p> <p>This table is read at runtime, and the output schema is available to the subsequent operator only after the operator is run.</p>
<b>Drop If Exists</b>	<p>Specifies what to do if a table of the same name already exists.</p> <ul style="list-style-type: none"> <li>• If <b>Yes</b> (the default), drop the existing table of the same name and create a new one.</li> <li>• If <b>No</b>, stop the flow and alert the user that an error has occurred.</li> </ul>

## Output

### Summary tab

Summary of the selected parameters.

Results - Import Excel	
Output	Parameters selected
Summary	Excel Input Name: BB_golfnew.xlsx Header File Path: /user/brittney/BB_golfnew_header.csv Sheet selected: 1 Top Left Corner Cell: A1 Right Cut-Off Column: Bottom Cut-Off Row:
	<b>Output</b> The output of the operator is stored at /user/brittney/import_Excel__1499375909914

## Output tab

Preview of the data extracted from the Excel workbook sheet.

Results - Import Excel					
Output	outlook	temperature	humidity	wind	play
Summary	outlook	N/A	N/A	wind	play
	sunny	85	85	FALSE	no
	rain	70	96	FALSE	yes
	sunny	80	90	TRUE	no
	rain	68	80	FALSE	yes
	overcast	83	78	FALSE	yes
	rain	65	70	TRUE	no
	overcast	64	65	TRUE	yes
	rain	75	80	FALSE	yes

## Data Output

A single tabular data set that is extracted from the sheet, and can be transmitted to subsequent operators only after the operator is run.

## Import Excel (HD)

Imports an Excel workbook sheet (or a portion of the sheet) as an HDFS input.



## Information at a Glance

Parameter	Description
Category	Load Data
Data source type	HD
Send output to other operators	Yes <sup>1</sup>

<sup>1</sup>The full output schema is not available until you step-run the operator. After the operator is run, the output schema automatically updates, and subsequent operators are either validated or turn red depending on the structure of the output data.

Parameter	Description
Data processing tool	n/a



**Note:** The Import Excel (HD) operator is for Hadoop data only. For database data, use the [Import Excel \(DB\)](#) operator.

The Excel workbook can be stored in HDFS or in the current workspace.

Formula cells, styles, dates, currencies, percentages, and so on are supported, and are parsed as numeric values. Non-tabular data such as images and pivot tables are skipped. Hidden columns and protected sheets are parsed as normal.

## Input

Import Excel (HD) is a source operator. No inputs are required.

## Bad or Missing Values

Blank cells or empty cells are converted to null values.

If a datetime type is specified in the **Column Metadata File** parameter but the Excel sheet contains values that cannot be parsed with this format, null values are used.

## Restrictions

Excel files are read on the TIBCO Data Science - Team Studio server. Depending on the memory available on your instance, loading very large Excel files on this server might require a large amount of memory and cause out-of-memory issues. For more information, see Apache POI limitations at <https://poi.apache.org/spreadsheet/limitations.html>.

TIBCO Data Science - Team Studio uses the configuration parameter `custom_operators`, set in the `alpine.conf` file, to avoid loading files that are too large. If the Excel file is bigger than this limit, it does not load and an error message is displayed. The default value is 30.0 (MB). The administrator of your TIBCO Data Science - Team Studio instance can modify the default value.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Data Source (Hadoop)</b>	The data source where the output should be stored (and where the <b>Column Metadata Table</b> is stored).
<b>Hadoop File</b>	<p>The Excel workbook if stored in HDFS.</p> <p><b>Note:</b> Supported formats are .xls, .xlsx and .xlsm.</p> <p>If this field is left blank, a work file must be identified (in the <b>Work File</b> parameter below).</p>
<b>Chorus Workfile</b>	<p>The Excel workbook if stored in the current workspace. Only workbooks with the following extensions are displayed: .xls, .xlsx, and .xlsm.</p> <p>If this field is left blank, a Hadoop file must be identified (in the <b>Data Source (Hadoop)</b> parameter above).</p>
<b>Sheet Number</b>	The number of the sheet to extract (the first sheet is 1).
<b>Top Left Corner Cell</b>	<p>The cell address that defines the top-left cell of the data portion to extract in the selected sheet (for example, B10).</p> <p><b>Note:</b> If your workbook sheet contains a header, skip it and select a cell on the following row. This is necessary because the header is read separately in the <b>Column Metadata File (CSV)</b> parameter, from a CSV file that includes both names and column types.</p>
<b>Right Cut-Off Column Letter</b>	An optional column letter that defines where the portion of data to extract should be cut at the right. If not specified, the data extraction is cut at the last defined cell of the first row selected (row number in <b>Top Left Corner Cell</b> ).



Parameter	Description
	<p><b>Note:</b> If this parameter is not specified and the first row selected is empty or not defined, an error is displayed.</p>
<b>Bottom Cut-Off Row Number</b>	An optional row number that defines where the portion of data to extract should be cut at the bottom. If not specified, the data extraction is cut at the last defined row in the sheet.
<b>Column Metadata File (CSV)</b>	<p>The HDFS file (the CSV file) that defines the header and column types for the output. The file should contain the following two rows of the same length.</p> <ul style="list-style-type: none"> <li>• The first row for column names.</li> <li>• The second row for column types (supported types are int, long, double, float, chararray, and datetime, with format specified (for example, datetime yyyy-MM-dd)).</li> </ul> <p>This file is read at runtime and the output schema is available to the subsequent operator only after the operator is run.</p>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>

## Output

### Summary tab

Summary of the parameters selected, as shown in the following image.

Results - Import Excel (HD)	
Output Summary	<b>Parameters selected</b>
	Excel Input Name: BB_golfnew.xlsx Header File Path /user/brittney/BB_golfnew_header.csv Sheet selected: 1 Top Left Corner Cell: A1 Right Cut-Off Column: Bottom Cut-Off Row:
	<b>Output</b> The output of the operator is stored at /user/brittney/Import_Excel__HD__1499375909914

### Output tab

Data preview of the data extracted from the Excel workbook sheet, as shown in the following image.

Results - Import Excel (HD)					
Output Summary	outlook	temperature	humidity	wind	play
	outlook	N/A	N/A	wind	play
	sunny	85	85	FALSE	no
	rain	70	96	FALSE	yes
	sunny	80	90	TRUE	no
	rain	68	80	FALSE	yes
	overcast	83	78	FALSE	yes
	rain	65	70	TRUE	no
	overcast	64	65	TRUE	yes
	rain	75	80	FALSE	yes

### Data Output

A single tabular data set that is extracted from the sheet, and can be transmitted to subsequent operators only after the operator is run.

## Load To Hive

Provides a mechanism for saving a table directly to a Hive database. You can use any tabular Hadoop file as the input, as long as the data source is configured for the chosen Hive Hadoop source.



## Information at a Glance

Parameter	Description
Category	Load Data
Data source type	HD
Send output to other operators	Yes
Data processing tool	n/a

## Input

A Hadoop operator with tabular output that has been connected to a Hive table upstream in the workflow, or a Hadoop data set. The Hadoop data source must be connected to the chosen Hive database.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Hive Result Database Name</b>	The database into which the data is copied.
<b>Hive Result Table Name</b>	The name of the table in which the data is stored. Default value: <b>alp@user_id_@flow_id_load_0</b> .

Parameter	Description
	For more information about the variables used in the default table name, see <a href="#">Workflow Variables</a> .
<b>Drop If Exists</b>	Specify what to do if a table of the same name already exists. <ul style="list-style-type: none"> <li>• If <b>Yes</b> (the default), drop the existing table of the same name and create a new one.</li> <li>• If <b>No</b>, stop the flow and alert the user that an error has occurred.</li> </ul>

## Outputs

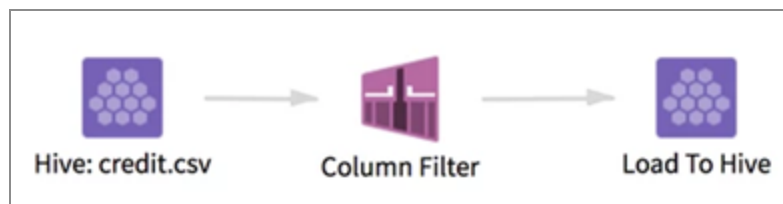
### Visual Output

A preview of the rows of the resulting data in the destination Hive table.

### Data Output

A data set that corresponds to the destination table.

## Example



## Exploration Operators

Exploration operators provide different ways to explore and visualize your data.

Knowing which type of visualization that is appropriate for your data is important for successful modeling. For more information about selecting the best visualization type, see [Visualizing data with charts and graphs](#).

## Bar Chart

Visualizes the attributes of a data set in bar chart format.



### Information at a Glance

Parameter	Description
Category	Explore
Data source type	DB, HD
Send output to other operators	No
Data processing tool	Pig

Use this operator to choose the following three attributes (columns) in the data table as the dimensions in the bar chart:

- The X dimension (**Category**) takes a category-type attribute to construct the X-axis.
- The Y dimension (**Value**) takes a numerical attribute to construct the Y-axis.
- The grouped dimension (**Series**) accepts a category-type attribute to group the bars in the chart.

### Input

A data set from the preceding operator.

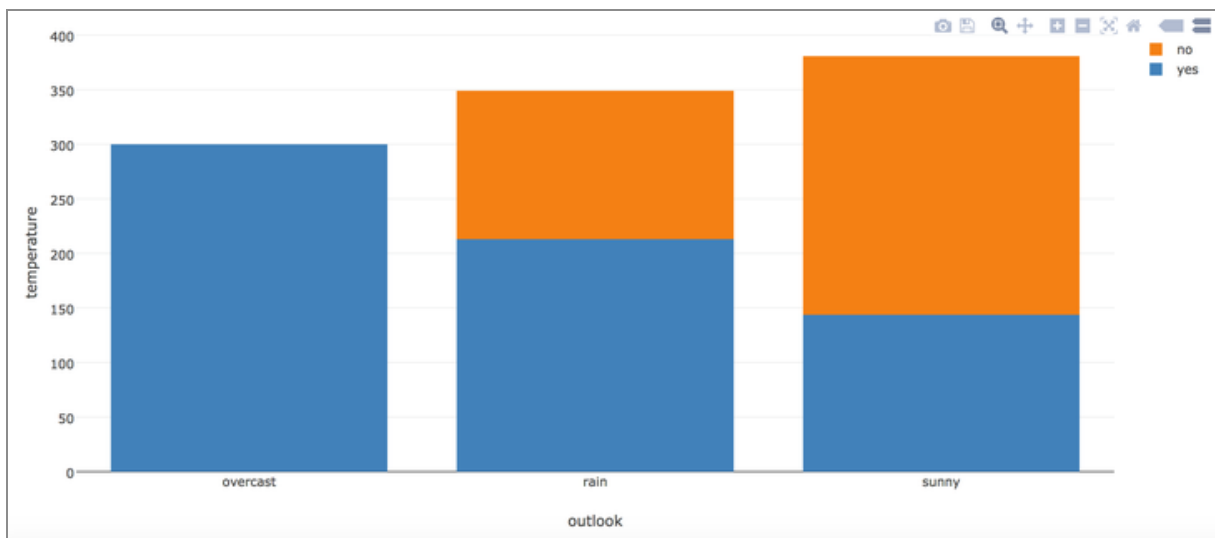
## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Category</b>	The categorical column for the X-axis.
<b>Series</b>	The categorical column for the groups.
<b>Value</b>	The numerical column for the Y-axis.

## Output

### Visual Output

A bar chart such as the following example.

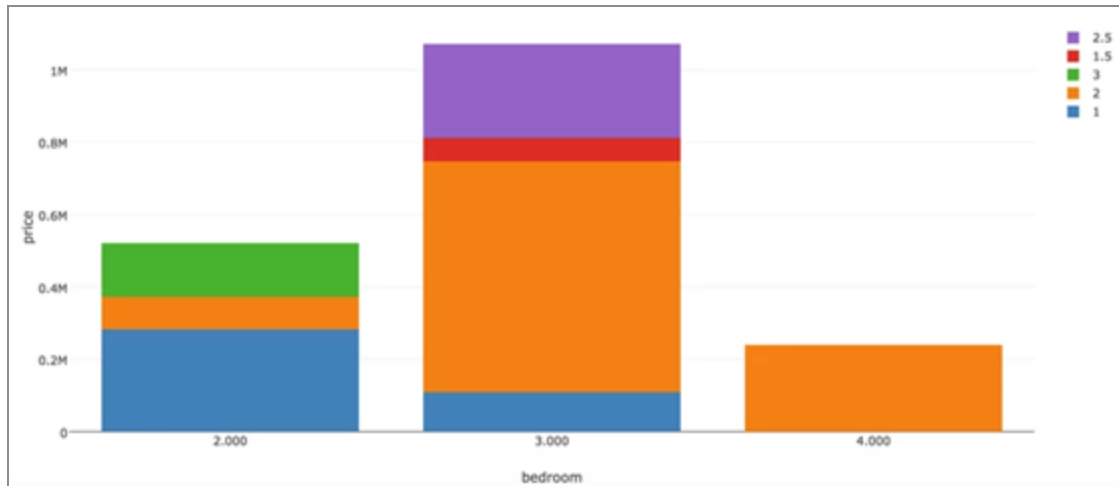


### Data Output

None. This is a terminal operator.

## Example

The following example shows housing sale prices (**Value**) by number of bedrooms (**Category**) for each number of baths (**Series**).



## Box Plot

Visualizes the attributes of a data set in box plot format.



## Information at a Glance

Parameter	Description
Category	Explore
Data source type	DB, HD
Send output to other operators	No
Data processing tool	Pig

Use this operator to create a graphical representation of a data set's attributes using the box-and-whisker diagrams.

For each box-and-whisker, the following applies.

- The bottom and top of the box indicate the 25th and 75th percentile of the data.
- The band in the middle of the box indicates the 50th percentile (median).
- The bottom and top of the whisker represents the minimum and maximum within the data set.
- The mean is denoted by a small circle.

The Box Plot operator allows you to choose the following three attributes (columns) in the data table:

- X dimension (**Analysis Type**) - Accepts a category-type attribute to construct the box plot's X-axis.
- Y dimension (**Analysis Value**) - Accepts a numerical attribute to construct the box plot's Y-axis.
- Grouped dimension (**Analysis Series**) - Accepts a category-type attribute, which is shown in diagrams of different colors.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Analysis Series</b>	The categorical column for the groups.
<b>Analysis Type (X-axis)</b>	The categorical column for the X-axis.



Parameter	Description
<b>Analysis Value (Y-axis)</b>	The numerical column for the Y-axis.
<b>Use approximation (faster)</b>	Specify whether to use approximation - <b>Yes</b> (the default) or <b>No</b> .

## Output

### Visual Output

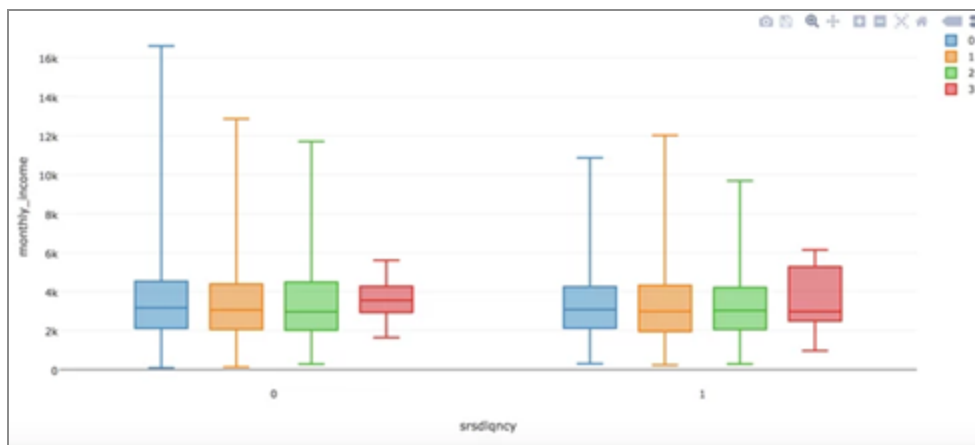
A box plot diagram.

### Data Output

None. This is a terminal operator.

## Example

The following example shows monthly income (**Analysis Value**) for delinquent/non-delinquent credit customers (**AnalysisType**) grouped by the Number of Times 90 Days Late (**Analysis Series**).



## Correlation (DB)

Use to specify two or more numeric type attributes (columns) in a data set for relative analysis against each other by calculating the correlation between each pair of selected columns.



### Information at a Glance

Parameter	Description
Category	Explore
Data source type	DB
Send output to other operators	No
Data processing tool	n/a

**i Note:** The Correlation (DB) operator is for database data only. For Hadoop data, use the [Correlation \(HD\)](#) operator.

### Algorithm

The covariance between two variables ( $X$  and  $Y$ ) is calculated as shown in the following formula:

$$\text{Cov}(X, Y) = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X}) (Y_i - \bar{Y}),$$

where  $\bar{X}$  and  $\bar{Y}$  are the mean values for  $X$  and  $Y$ , respectively.

The correlation is calculated by normalizing the covariance, as shown in the following formula:

$$\text{Corr}(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Cov}(X, X)\text{Cov}(Y, Y)}} = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}}$$

**i Note:** The [Principal Component Analysis](#) operator is a multivariate modeling operator that also determines the covariance and correlation between variables. However, it goes a step further by applying a mapping of the variables into a reduced Principal Component space.

For information about correlation and covariance, see [Correlation and Covariance](#).

## Input

A data set from the preceding operator.

### Bad or Missing Values

In TIBCO Data Science - Team Studio, all null values are filtered for Correlation Analysis.

## Restrictions

The algorithm is relevant only when run on numeric data.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	The numeric columns for which the correlation should be calculated.

## Output

### Visual Output

The correlation coefficient table. Each coefficient value provides a measure of how related the two variables are to each other. The value is 1 when the column is being

compared against itself. A negative value means an opposite, negative relationship (that is, as one value goes up, the other goes down).

Results - Correlation - DB				
Column Name	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1	-0.11	0.87	0.82
sepal_width	-0.11	1	-0.42	-0.36
petal_length	0.87	-0.42	1	0.96
petal_width	0.82	-0.36	0.96	1

**i Note:** These values are equivalent to the correlation coefficients in a linear regression (with the column name as the dependent variable). This output could be useful, for example, in deciding which variables to include in a linear regression model.

## Data Output

None. This is a terminal operator.

## Correlation (HD)

Use to specify two or more numeric type attributes (columns) in a data set for relative analysis against each other by calculating the correlation between each pair of selected columns.



## Information at a Glance

Parameter	Description
Category	Explore
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce

**i Note:** The Correlation (HD) operator is for Hadoop data only. For database data, use the [Correlation \(DB\)](#) operator.

## Algorithm

The covariance between two variables ( $X$  and  $Y$ ) is calculated as shown in the following formula:

$$\text{Cov}(X, Y) = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y}),$$

where  $\bar{X}$  and  $\bar{Y}$  are the mean values for  $X$  and  $Y$ , respectively.

The correlation is calculated by normalizing the covariance, as shown in the following formula:

$$\text{Corr}(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Cov}(X, X)\text{Cov}(Y, Y)}} = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}}.$$

**i Note:** The [Principal Component Analysis](#) operator is a multivariate modeling operator that also determines the covariance and correlation between variables. However, it goes a step further by applying a mapping of the variables into a reduced Principal Component space.

For information about correlation and covariance, and the algorithms that describe them, see [Correlation and Covariance](#).

## Input

A data set from the preceding operator.

## Bad or Missing Values

In TIBCO Data Science - Team Studio, all null values are filtered for Correlation Analysis.

## Restrictions

The algorithm is relevant only when run on numeric data.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	The numeric columns for which the correlation or covariance should be calculated.
<b>Group by</b>	When you select one or more group-by columns, the operator calculates a separate correlation (or covariance) matrix for every combination of values in the group-by columns. You can select one or more group-by columns.  <b>Note:</b> The <b>Group by</b> column selection cannot overlap with the main <b>Columns</b> selection.
<b>Calculate</b>	Specify whether to calculate the <b>Correlation</b> (the default) or the <b>Covariance</b> .  Correlation is normalized covariance, scaled so that the correlation between any variable and a positive multiple of itself is always 1.

<b>Store Results?</b>	<p>Specifies whether to store the results.</p> <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	<p>The HDFS directory where the results of the operator are stored. This is the main directory, the sub-directory of which is specified in <b>Results Name</b>. Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.</p>
<b>Results Name</b>	<p>The name of the file in which to store the results.</p>
<b>Overwrite</b>	<p>Specifies whether to delete existing data at that path and file name.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Advanced Spark Settings</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit</b></li> </ul>

**Automatic Optimization**

**Settings** to customize Spark optimization. See [Advanced Settings dialog](#) for more information.

## Output

### Visual Output

One correlation (or covariance) matrix for each combination of specified group-by values.

**i Note:** If a group-by requirement is not specified, only one matrix is output.

### Data Output

For Hadoop data set analysis, the visual output also is passed as its output to any following operator.

## Example

The following example shows both the Hadoop correlation matrix and the corresponding covariance matrix output for different group by classes of Iris flowers. Note that when the correlation attribute is compared to itself, the resulting correlation coefficient value is 1 (this is not the case for the covariance data).

Results - Correlation - Hadoop					
"class"	Attribute	sepal_length	sepal_width	petal_length	petal_width
Iris-setosa	sepal_length	1	0.75	0.26	0.28
Iris-setosa	sepal_width	0.75	1	0.18	0.28
Iris-setosa	petal_length	0.26	0.18	1	0.31
Iris-setosa	petal_width	0.28	0.28	0.31	1
Iris-versicolor	petal_length	0.75	0.56	1	0.79
Iris-versicolor	petal_width	0.55	0.66	0.79	1
Iris-versicolor	sepal_length	1	0.53	0.75	0.55
Iris-versicolor	sepal_width	0.53	1	0.56	0.66
Iris-virginica	petal_width	0.28	0.54	0.32	1
Iris-virginica	sepal_width	0.46	1	0.4	0.54
Iris-virginica	petal_length	0.86	0.4	1	0.32
Iris-virginica	sepal_length	1	0.46	0.86	0.28



Results - Covariance - Hadoop					
"class"	Attribute	sepal_length	sepal_width	petal_length	petal_width
Iris-setosa	sepal_length	0.12	0.1	0.02	0.01
Iris-virginica	petal_width	0.05	0.05	0.05	0.08
Iris-setosa	sepal_width	0.1	0.15	0.01	0.01
Iris-versicolor	petal_length	0.18	0.08	0.22	0.07
Iris-versicolor	petal_width	0.06	0.04	0.07	0.04
Iris-versicolor	sepal_length	0.27	0.09	0.18	0.06
Iris-virginica	sepal_length	0.4	0.09	0.3	0.05
Iris-setosa	petal_length	0.02	0.01	0.03	0.01
Iris-versicolor	sepal_width	0.09	0.1	0.08	0.04
Iris-virginica	sepal_width	0.09	0.1	0.07	0.05
Iris-setosa	petal_width	0.01	0.01	0.01	0.01
Iris-virginica	petal_length	0.3	0.07	0.3	0.05

## Frequency

Analyzes the values of selected fields in a table, helping to interpret the shape of the data column by column.



## Information at a Glance

Parameter	Description
Category	Explore
Data source type	DB, HD
Send output to other operators	No

Parameter	Description
Data processing tool	Pig

## Algorithm

Frequency Analysis provides statistics on the distinct values of the data, including the count and percentage of each value of a column.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	Click <b>Select Columns</b> to select the available columns from the input data set for analysis. For more information, see <a href="#">Select Columns dialog</a> .  <b>Note:</b> Typically, it does not make sense to run a Frequency Analysis on unique text (name) or ID columns, because the frequency of each ID value is, by definition, 1.

## Output

### Visual Output

#### Counts

Displays the shape analysis graph for the count of each distinct value of the selected data column (name) for analysis. View the Frequency results of any analyzed column by selecting the column from the name dropdown list.

**Important:** To learn more about the visualization available in this operator, see [Explore Visual Results](#).

## Data

Displays the column name, value, count, and frequency percentage for each selected column (name), as shown in the following example:

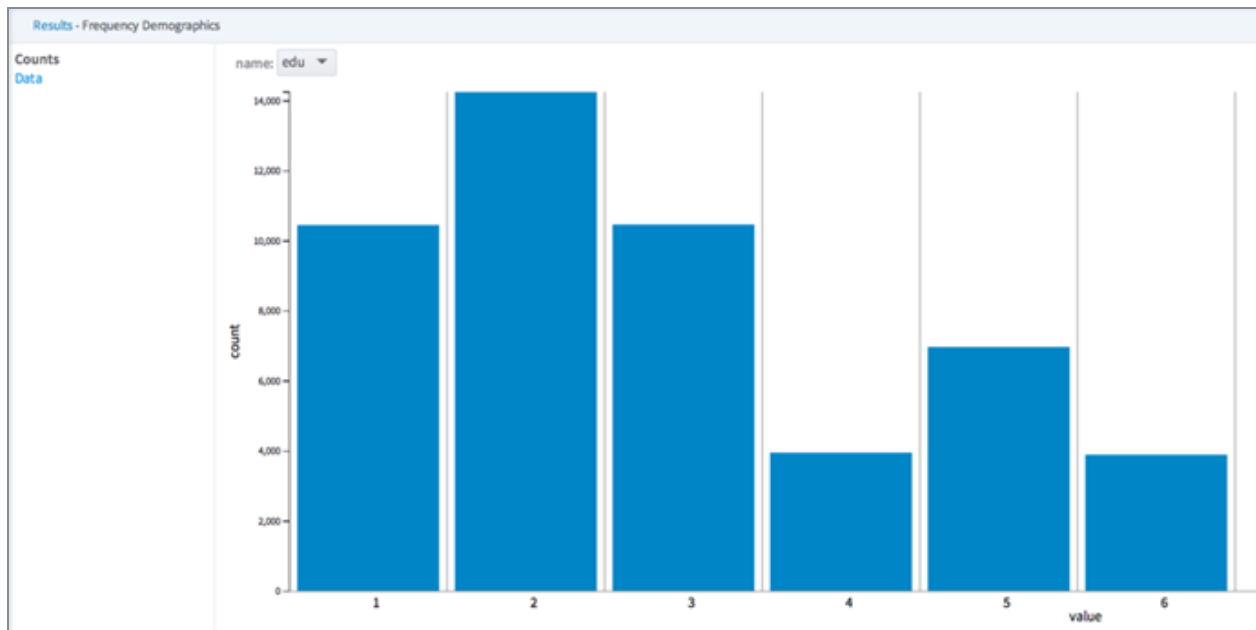
Results - Frequency Demographics			
Counts Data	name: edu ▼		
	name	value	count
	edu	1	10,453
	edu	2	14,249
	edu	3	10,468
	edu	4	3,958
	edu	5	6,973
	edu	6	3,899
Percentage			
20.906%			
28.498%			
20.936%			
7.916%			
13.946%			
7.798%			

## Data Output

None. This is a terminal operator.

## Example

The following example shows a shape analysis graph that displays the frequency count of various educational levels.



## Histogram

Analyzes the values of the selected fields of a data set, and generates a graphical representation of the frequency distribution of the numeric data.



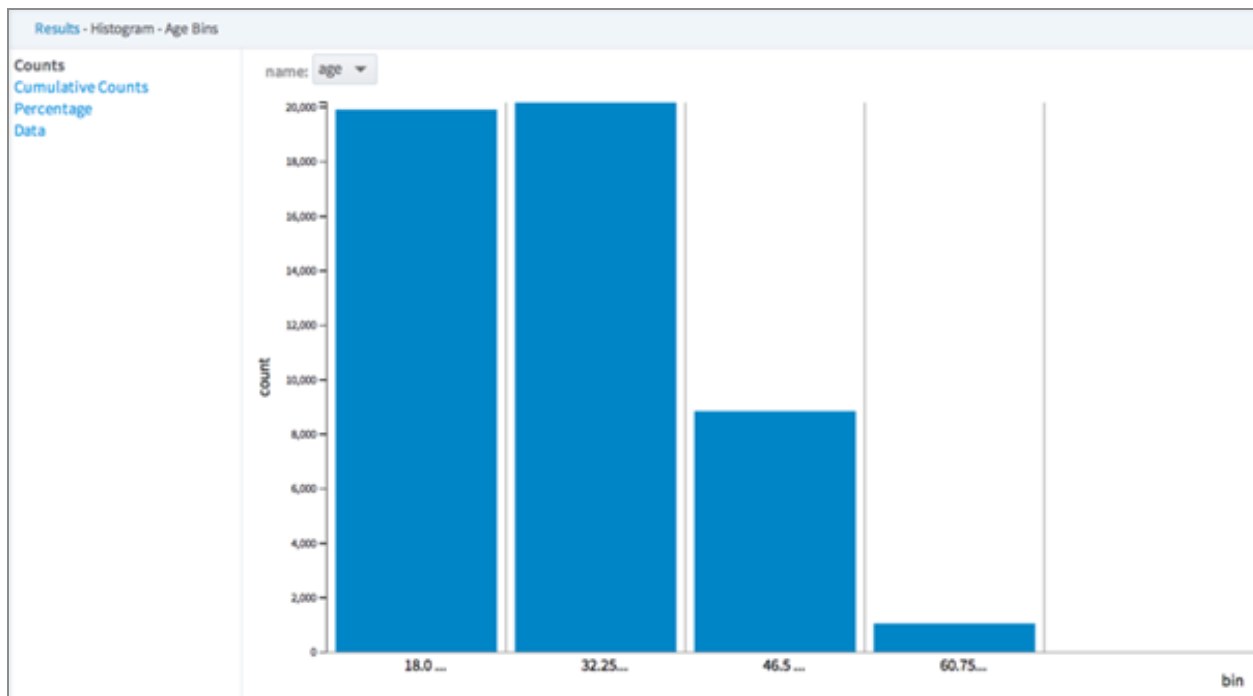
## Information at a Glance

Parameter	Description
Category	Explore
Data source type	DB, HD

Parameter	Description
Send output to other operators	No
Data processing tool	Pig

## Algorithm

Histogram analysis calculates data frequency for a specific column.



For each column specified, users input either the number of bins to generate or the width of the bins. A bin is an interval that is divided equally between minimum and maximum value or by the width.

For example, a specific column's minimum value is 0 and maximum value is 100. If the user specifies five bins, each bin covers 20 units. If 10 bins are specified, each bin covers 10 units.

Bounds of each bin are defined as (**Minimum**, **Maximum**].

**i Note:** When a user defines a minimum value, this value is included or displayed in the bin. The first bin begins at the lowest value above the defined minimum, and the last bin includes the defined maximum value.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Bins</b>	Select <b>Bin Configuration</b> to choose the available columns from the input data set for analysis.  See <a href="#">Bin Configuration dialog</a> .

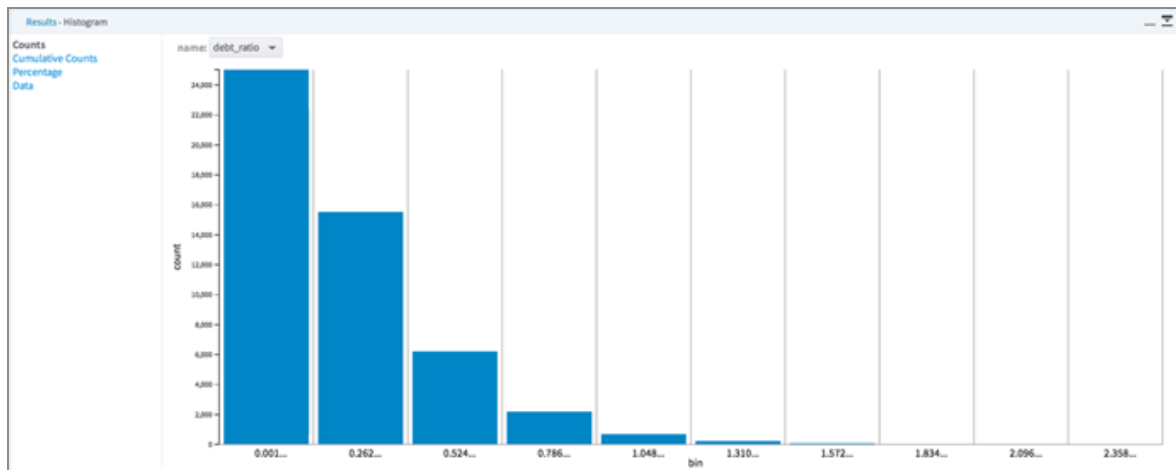
## Output

### Visual Output

Four sections are displayed: **Counts**, **Cumulative Counts**, **Percentage**, and **Data**.

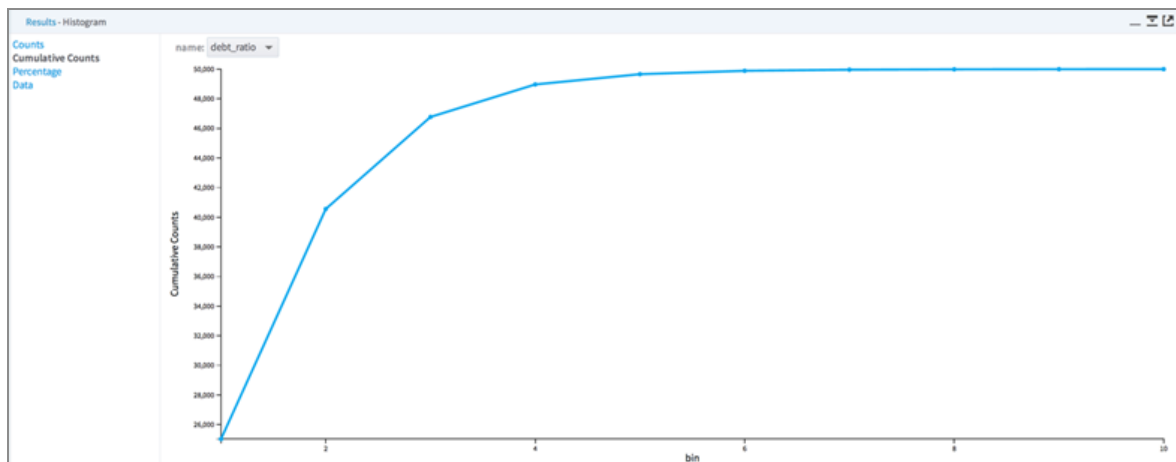
#### Counts

Displays the histogram for one column at a time according to the defined groups (bins). Users can select a column from the **name** dropdown list.



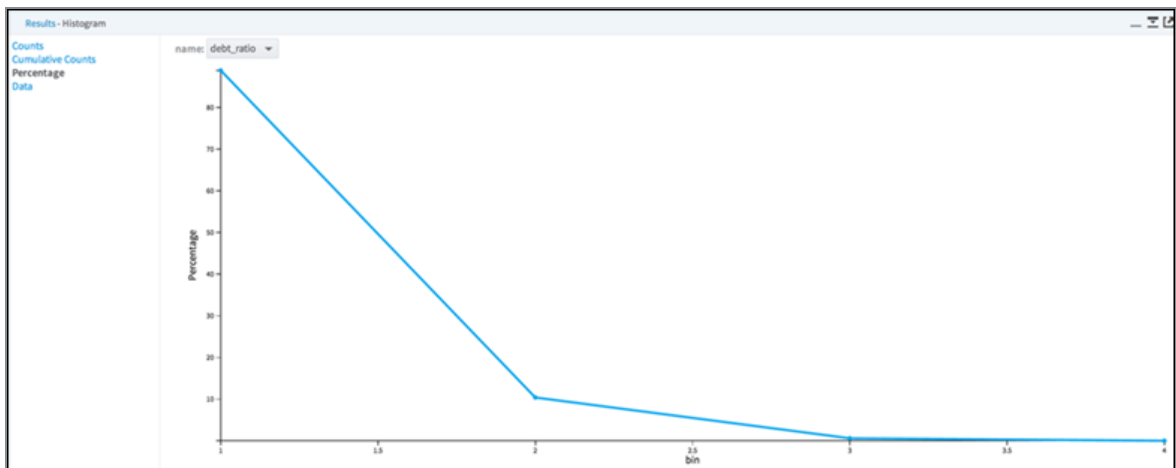
## Cumulative Counts

Displays a graph of the number of rows included with each additional bin.



## Percentage

Displays a graph showing what percentage of the input column each bin represents.



## Data

Summarizes information about each histogram, with numerical measures for:

- Bin name
- Bin number
- Bin start point
- Bin end point
- Count
- Percentage
- Cumulative counts
- Cumulative %

Results - Histogram							
Counts	name: debt_ratio						
Cumulative Counts							
Percentage							
Data							
	name	bin	begin	end	count	Percentage	Cumulative Counts
	debt_ratio	1	0	0.66	44,449	88.898%	44,449
	debt_ratio	2	0.66	1.31	5,211	10.422%	49,660
	debt_ratio	3	1.31	1.97	318	0.636%	49,978
	debt_ratio	4	1.97	2.62	22	0.044%	50,000



**i Note:** To learn more about the visualization available in this operator, see [Explore Visual Results](#).

## Data Output

None. This is a terminal operator.

## Information Value

Calculates both the information value (IV) and weight of evidence (WOE) of attributes. These are measures of the overall "relevance" of a data variable in predicting the dependent column's desired value or outcome.



## Information at a Glance

Parameter	Description
Category	Explore
Data source type	DB
Send output to other operators	No
Data processing tool	n/a

For more information about IV and WOE, see [Information Value and Weight of Evidence Analysis](#).

## Algorithm

The Information Value operator uses the following formulas for calculating IV and WOE:

$$\text{Weight of Evidence} = \ln(\text{Distribution Good} / \text{Distribution Bad}) * 100$$

$$\text{Information Value} = \text{sum}((\text{Distribution Good} - \text{Distribution Bad}) * \ln(\text{Distribution Good} / \text{Distribution Bad}))$$

where `Distribute Good` refers to percentage of values, for each given independent variable grouping, that results in the desired "Value to Predict" for the dependent variable and `Distribution Bad` is the percentage of values within each grouping that is not the "Value to Predict."

The following table provides an example.

Attribute	Count Goods	Distribution Good	Count Bads	Distribution Bad	WOE
Missing	1	10%	3	30%	-109.9
true	3	30%	2	20%	40.55
false	6	60%	5	50%	18.23

$$\text{Information Value} = (10\% - 30\%) * \ln(10\% / 30\%) + (30\% - 20\%) * \ln(30\% / 20\%) + (60\% - 50\%) * \ln(60\% / 50\%) = 0.2785$$

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the

Parameter	Description
	operator.
<b>Dependent Column</b>	<p>The column to use as the class variable.</p> <p>Note: The <b>Dependent Column</b> must be a categorical (not continuous) variable.</p>
<b>Value to Predict</b>	<p>The value stored in the <b>Dependent Column</b> that represents the event to analyze (for example, Active vs. Inactive).</p> <p>The <b>Value to Predict</b> must be a value that exists for the <b>Dependent Column</b>. It is considered the "good" event.</p>
<b>Columns</b>	<p>Columns to use to analyze the relevance of or effect on the <b>Dependent Column</b> value equaling the <b>Value to Predict</b>.</p> <p>Click <b>Select Columns</b> to open the dialog to select the available columns from the input data set for analysis. See <a href="#">Select Columns dialog</a> for more information.</p> <p>Column names selected must be categorical values.</p>

## Output

### Visual Output

The results display provides the IV and WOE for each selected independent variable, providing insight into how effective each variable is in predicting the desired dependent variable value.

Results - Information Value		
<b>times90dayslate</b> <b>times30dayslate_2years</b> <b>age</b> <b>num_dep</b> <b>edu</b>	Information Value / Weight of Evidence	Value
	IV(times30dayslate_2years)	0.13231212
	WOE(5)	NaN
	WOE(4)	212.53440704
	WOE(3)	91.77414981
	WOE(2)	130.09877793
	WOE(1)	72.53508650
	WOE(0)	-15.24967334
	WOE(alpine_miner_null)	NaN

Results - Information Value		
<b>times90dayslate</b> <b>times30dayslate_2years</b> <b>age</b> <b>num_dep</b> <b>edu</b>	Information Value / Weight of Evidence	Value
	IV(edu)	0.00186406
	WOE(6)	2.49693191
	WOE(5)	-1.93869123
	WOE(4)	-11.24720222
	WOE(3)	4.27886696
	WOE(2)	-2.52436483
	WOE(1)	3.36807717
	WOE(alpine_miner_null)	NaN

## Data Output

None. This is a terminal operator.

## Line Chart

Produces a line chart for a user-specified X-axis and Y-axis, and aggregates the values in the Y-axis.



### Information at a Glance

Parameter	Description
Category	Explore
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark SQL

Line Chart takes the parameters you specify and charts the aggregated X-axis column(s) and Y-axis columns, using Plotly to display an interactive chart. You can choose multiple columns to graph as long as they all use the same aggregation method.

### Input

A single tabular data set.

### Bad or Missing Values

Line Chart filters data on the Y-axis selector to select numeric values only, because this column has numeric operations performed on it. The operator cannot automatically convert dates that are put in as character arrays, so conversion is the user's responsibility. The operator handles null values by skipping fully null data points, but it also can manage them in aggregations.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>X-Axis Values</b>	The column by which to group or aggregate and display on the x-axis.
<b>Y-Axis Values</b>	The column used on the y-axis of the chart. These are the numeric values that are aggregated based on the x-axis value or category.
<b>Aggregation Method</b>	<p>The type of aggregation to perform on the numeric y-axis values. The options are the following.</p> <ul style="list-style-type: none"> <li>• Average</li> <li>• Count</li> <li>• Maximum</li> <li>• Minimum</li> <li>• Sum</li> </ul>
<b>Maximum Number of Plot Points</b>	<p>The maximum number aggregations that are plotted on the line.</p> <p>For example, if you have 100 values, and you aggregate the sum according to each value and set the parameter to 90, then Line Chart plots only the first 90 aggregated values and ignores the last 10 aggregated values.</p>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>

Parameter	Description
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

A custom visualization for the X/Y line chart, and a table of resulting aggregations.

### Data Output

An aggregated data set as shown in the table component.

## Scatter Plot Matrix

Creates pairwise scatter plots of the selected columns. This gives a visual sense of the relationship between each of the paired attributes, as well as the calculated correlation.



## Information at a Glance

Parameter	Description
Category	Explore

Parameter	Description
Data source type	DB, HD
Send output to other operators	No
Data processing tool	Pig

## Input

A data set from the preceding operator.

## Configuration

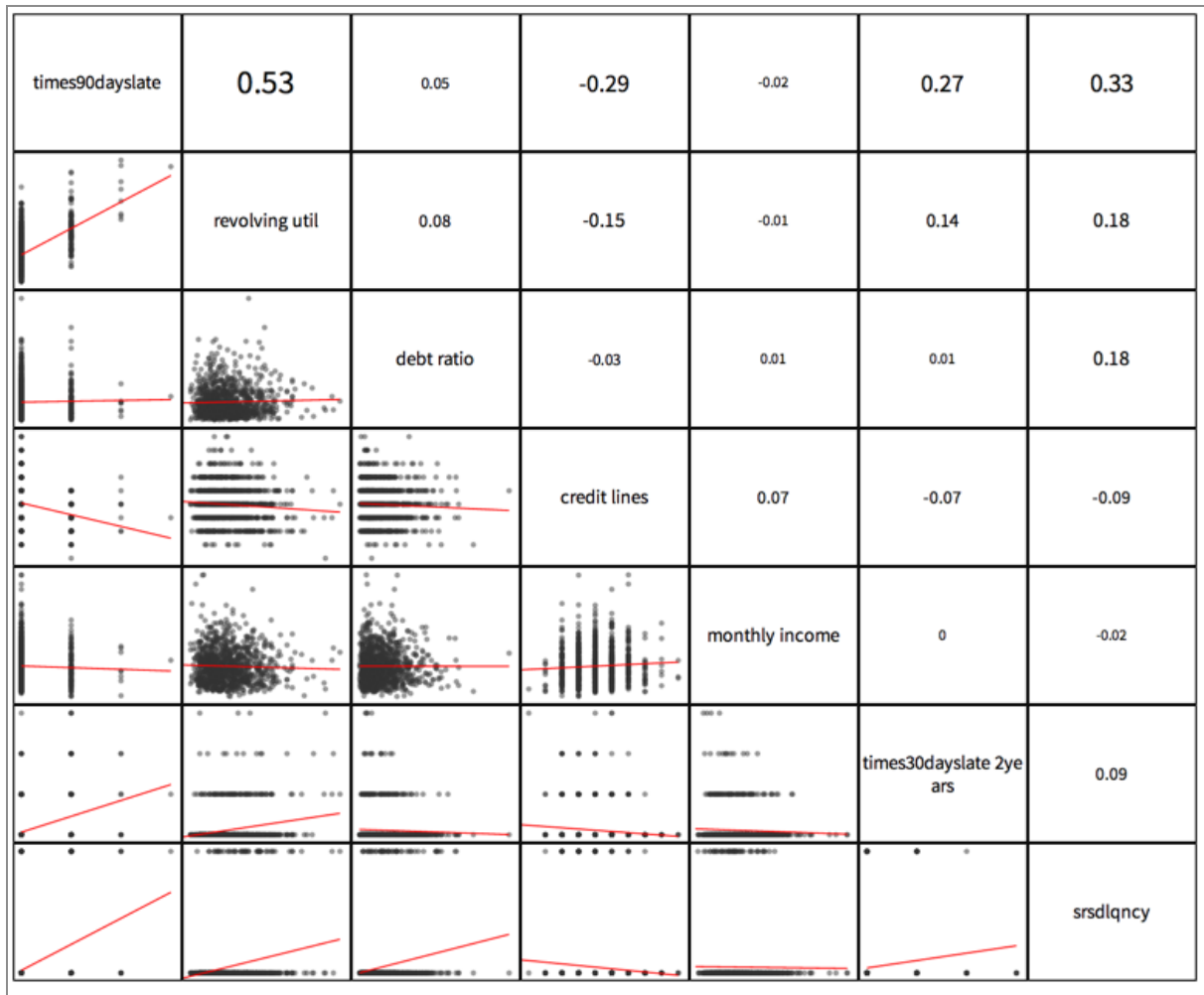
Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Select Columns</b>	Click to select the columns from the input data set. Choose a set of numeric columns to view a scatter plot matrix. The matrix uses 200 data points as the default sample set to create the scatter plot.

## Output

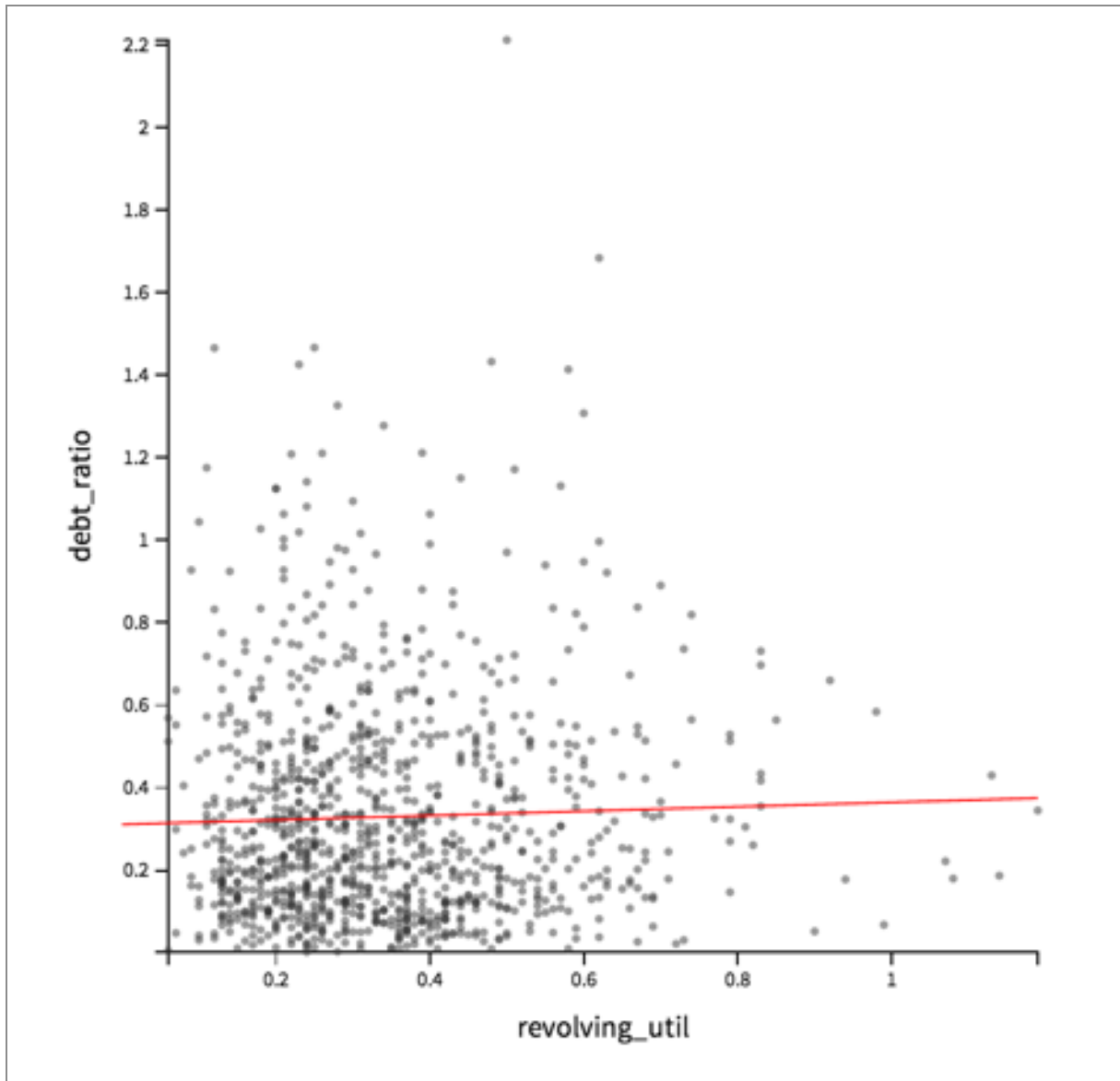
### Visual Output

A matrix of the generated scatter plots based on the various pairs of selected data columns (variables). The plotted points are selected randomly from the input data set. The correlations are computed based on the entire data set, not on the sample points.





Double-click a small scatter plot to display a full size scatter plot.



**i Note:** For more information about the full-size scatter plot visualization, see [Explore Visual Results](#).

### Data Output

None. This is a terminal operator.

## Summary Statistics (DB)

Provides useful summary information for the selected columns of the data set passed by the preceding operator.



### Information at a Glance

Parameter	Description
Category	Explore
Data source type	DB
Send output to other operators	No
Data processing tool	n/a



**Note:** The Summary Statistics (DB) operator is for database data only. For Hadoop data, use the [Summary Statistics \(HD\)](#) operator.

### Input

A data set from the preceding operator.

### Restrictions

The Summary Statistics operator does not work with generic JDBC data sets. See the Summary Statistics (DB).

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	<p>Select the numeric columns for which the summary statistics should be displayed.</p> <ul style="list-style-type: none"> <li>Click <b>Select Columns</b> to open the dialog for selecting the available columns from the input dataset for analysis.</li> <li>Select or clear the box in front of the column names to select or deselect the column.</li> <li>Click <b>All</b> to select all the columns.</li> <li>Click <b>OK</b> to commit the selection changes.</li> <li>Click <b>Cancel</b> to cancel all the selection changes.</li> </ul>
<b>Group By</b>	Click <b>Select Columns</b> to open the dialog for selecting the available columns from the input dataset to group results by.
<b>Calculate the Number of Distinct Values (slower)</b>	<p>Determines whether to calculate the number of distinct values for selected columns.</p> <p>Calculating distinct values can add significant processing time.</p> <p>Default value: Yes</p>
<b>Number of most Common Values to Display</b>	<p>Determines the maximum number of the most common values to output for each column.</p> <p>Only enabled if <b>Calculate the Number of Distinct Values</b> is enabled.</p>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user

Parameter	Description
	ID, workflow ID, and operator.
<b>Storage Parameters</b>	Advanced database settings for the operator output. Available only for <b>TABLE</b> output.  See <a href="#">Storage Parameters dialog</a> for more information.
<b>Drop If Exists</b>	Specifies whether to overwrite an existing table. <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

A table that displays the analysis results of the selected fields. The following list shows the default table contents.

- Name
- Data type
- Count
- Unique value count
- Null value count
- Empty value count
- Zero value count
- Min value
- 25% (approx.) - Approximate 25% value for numerical columns.
- Median (approx.) - Approximate median value for numerical columns.
- 75% (approx.) - Approximate 75% value for numerical columns.

- Maximum value
- Standard deviation
- Average
- Positive value count
- Negative value count
- Most Common (Value) - The most common value for the column.
- Most Common (Percentage) - The percentage of the total which are the most common value.
- 2nd Most Common (Value) - The second most common value.
- 2nd Most Common (Value) - The percentage of the total which are the second most common value.

### Data Output

A data set of the analysis results (that is, the same data shown in the visual output).

## Summary Statistics (HD)

Provides useful summary information for the selected columns of the data set passed by the preceding operator.



## Information at a Glance

Parameter	Description
Category	Explore
Data source type	HD
Send output to other operators	No
Data processing tool	Pig



**Note:** The Summary Statistics (HD) operator is for Hadoop data only. For database data, use the [Summary Statistics \(DB\)](#) operator.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	Select the numeric columns for which the summary statistics should be displayed. Click <b>Select Columns</b> to open the dialog to select the available columns from the input data set for analysis.
<b>Group By</b>	Columns from the input data set by which to group results.
<b>Calculate the Number of Distinct Values</b>	Determines whether to calculate the number of distinct values for selected columns - <b>Yes</b> (the default) or <b>No</b> .

Parameter	Description
(slower)	<b>Note:</b> Calculating distinct values can add significant processing time.
<b>Number of Most Common Values to Display</b>	Determines the maximum number of the most common values to output for each column.  Only enabled if <b>Calculate the Number of Distinct Values</b> is enabled.
<b>Store Results?</b>	Specifies whether to store the results. <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	The HDFS directory where the results of the operator are stored. This is the main directory, the subdirectory of which is specified in <b>Results Name</b> . Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.
<b>Results Name</b>	The name of the file in which to store the results.
<b>Overwrite</b>	Specifies whether to delete existing data at that path and file name. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

A table that displays the analysis results of the selected fields. The following list shows the default table contents.

- Name



- Data type
- Count
- Unique value count
- Null value count
- Empty value count
- Zero value count
- Min value
- 25% (approx.) - Approximate 25% value for numerical columns.
- Median (approx.) - Approximate median value for numerical columns.
- 75% (approx.) - Approximate 75% value for numerical columns.
- Maximum value
- Standard deviation
- Average
- Positive value count
- Negative value count
- Most Common (Value) - The most common value for the column.
- Most Common (Percentage) - The percentage of the total which are the most common value.
- 2nd Most Common (Value) - The second most common value.
- 2nd Most Common (Value) - The percentage of the total which are the second most common value.

### **Data Output**

A data set of the analysis results (that is, the same data shown in the visual output).

## **Variable Selection (DB)**

Identifies and prioritizes the variables of interest to a prediction task or model. This is especially helpful when there are a large number of potential variables for a model, enabling the modeler to focus on only a subset of those that show the strongest relevance.



## Information at a Glance

**Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Explore
Data source type	DB
Send output to other operators	No
Data processing tool	n/a

**Note:** The Variable Selection (DB) operator is for database data only. For Hadoop data, use the [Variable Selection \(HD\)](#) operator.

## Algorithm

For database, there are three information gain-based scoring metrics for variable selection: Information Gain, Information Gain Ratio, and Transformed Information Gain. For numerical columns, we discretize them. We recommend a preliminary threshold of the variables by comparing their scores to a random benchmark.

## Information Gain

Information Gain is a measure of the change in the entropy (or uncertainty) of a random variable  $Y$  when it is conditioned on another (categorical) variable  $X$ . In our case,  $Y$  is the class to be predicted (the dependent variable), and  $X$  is a candidate driver.

The entropy of a categorical random variable  $Y$  with  $n$  possible values (classes) is given by

$$H(Y) = - \sum_{i=1}^n p(y_i) \log_2(p(y_i))$$

The conditional entropy of  $Y$  given the values of a discrete variable  $X$  that takes on the  $m$  values is given by

$$H(Y|X) = \sum_{j=1}^m p(x_j) H(Y|X = x_j)$$

The information gain about  $Y$ , given that we know  $X$  measures how much more we know about  $Y$  because we know  $X$ :

$$IG(Y, X) = H(Y) - H(Y|X)$$

## Information Gain Ratio

The standard way to adjust for the biases of information gain is to normalize by the entropy of  $X$ . This is called the Information Gain Ratio.

$$\text{Gain Ratio}(Y, X) = IG(Y, X)/H(X)$$

**i Note:** The higher the information gain ratio, the better  $X$  is at predicting  $Y$ .

## Transformed Information Gain

Another way to adjust for bias is to map all candidate features into the same number of classes.

For binary output variables, we can create a simple predictor from each candidate feature, and then measure the information gain in  $Y$ , given the simple predictions from  $X$ .

One way to build a simple predictor is as follows:

- Calculate the prior probabilities of the true output class,  $P$
- Calculate the probability of the true output class for each of the input classes:  $p_i = p(Y = \text{TRUE} | X = x_i)$
- If  $p_i > P$ , predict TRUE for all the members of the class  $x_i$ , otherwise predict FALSE.

This transforms the variable  $X$  to the simple predictor, which takes on the same number of classes as  $Y$ . The score for  $X$  is now given by  $IG(Y, \text{"simple predictor"})$ .



**Note:** The higher the Transformed Information Gain, the better  $X$  is at predicting  $Y$ .

## Score Threshold by Chance

Score threshold by chance is a score we can get just by chance, even if  $X$  is not truly predictive of  $Y$ . We generate  $X$  that are designed to be independent of  $Y$  according to the distribution of  $Y$  and then calculate the score. We can generate a lower-bound threshold  $T$ . Any candidate feature that scores lower than  $T$  is almost certainly not predictive of the output variable, and can be eliminated. In practice,  $T$  is quite small, and probably does not eliminate too many variables. However, it still gives a useful sense of scores that correspond to meaningful and less-meaningful variables.

## Handling numerical columns with Information Gain

In a database, we approximate the probability density of continuous/numerical  $X$  by histograms. To do this, we bin  $X$  into a fairly large number of discrete classes, and then use the equation up, or the transform technique to calculate the scores.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	Define the column to use as the class variable. For R2 scoring, the dependent column must be numeric.
<b>Score Type</b>	Database options: <ul style="list-style-type: none"> <li>• <b>Info gain</b></li> <li>• <b>Info gain ratio</b></li> <li>• <b>Transformed info gain</b></li> </ul>
<b>Columns</b>	Use <b>Select Columns</b> to open the dialog to select the available columns from the input data set for analysis.
<b>Score Threshold</b>	Indicates what a good value is for your score type. Variables above the score threshold are stored and passed to subsequent operators.  Default value: <b>0.1</b> .
<b>Always included columns</b>	Columns in this list are automatically passed to subsequent operators, whether or not their score is above the threshold.
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Drop If Exists</b>	Specifies whether to overwrite an existing table.

Parameter	Description
	<ul style="list-style-type: none"><li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li><li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li></ul>

## Output

### Visual Output

The category and number column score by chance and variable selection score of each column.

### Data Output

None. This is a terminal operator.

## Variable Selection (HD)

Identifies and prioritizes the variables of interest to a prediction task or model. This is especially helpful when there are a large number of potential variables for a model, enabling the modeler to focus on only a subset of those that show the strongest relevance.



## Information at a Glance

Parameter	Description
Category	Explore
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce



**Note:** The Variable Selection (HD) operator is for Hadoop data only. For database data, use the [Variable Selection \(DB\)](#) operator.

## Algorithm

For Hadoop, there are two choices: Information Gain and R2. The Hadoop-based information gain option calculates both information gain and information gain ratio.

### Information Gain

Information Gain is a measure of the change in the entropy (or uncertainty) of a random variable  $Y$  when it is conditioned on another (categorical) variable  $X$ . In our case,  $Y$  is the class to be predicted (the dependent variable), and  $X$  is a candidate driver.

The entropy of a categorical random variable  $Y$  with  $n$  possible values (classes) is given by

$$H(Y) = - \sum_{i=1}^n p(y_i) \log_2(p(y_i))$$

The conditional entropy of  $Y$  given the values of a discrete variable  $X$  that takes on the  $m$  values is given by

$$H(Y|X) = \sum_{j=1}^m p(x_j) H(Y|X = x_j)$$

The information gain about  $Y$ , given that we know  $X$  measures how much more we know about  $Y$  because we know  $X$ :

$$IG(Y, X) = H(Y) - H(Y|X)$$

### Score Threshold by Chance


Score threshold by chance is a score we can get just by chance, even if  $X$  is not truly predictive of  $Y$ . We generate  $X$  that are designed to be independent of  $Y$  according to the distribution of  $Y$  and then calculate the score. We can generate a lower-bound threshold  $T$ . Any candidate feature that scores lower than  $T$  is almost certainly not predictive of the output variable, and can be eliminated. In practice,  $T$  is quite small, and probably does not eliminate too many variables. However, it still gives a useful sense of scores that correspond to meaningful and less-meaningful variables.

### Handling numerical columns with Information Gain

For numerical columns in Hadoop, we compute mutual information between dependent and independent variables without discretization. The Hadoop Variable Operator does not perform Minimum Description Length (MDL) discretization like the database version, because it is extremely expensive when dealing with big data.

### R<sup>2</sup>

For each independent variable  $X$ ,  $R^2$  is the coefficient of determination for a simple linear regression between  $X$  and the dependent variable  $Y$ .

 **Note:**  $R^2$  indicates how well a linear model  $f(X) = \alpha + \beta X$  fits the given data.

### Input

A data set from the preceding operator.



## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	Define the column to be used as the class Variable.  For R2 scoring, the dependent column must be numeric.
<b>Score Type</b>	Hadoop options: <ul style="list-style-type: none"> <li>• R2</li> <li>• Info Gain</li> </ul>
<b>Columns</b>	Use <b>Select Columns</b> to open the dialog for selecting the available columns from the input dataset for analysis.
<b>Score Threshold</b>	Indicates what a good value is for your score type. Variables above the score threshold are stored and passed to subsequent operators.  Default value: <b>0.5</b> .
<b>Always included columns</b>	Columns in this list are automatically passed to subsequent operators, whether or not their score is above the threshold.
<b>Do power transformations?</b>	If <b>Yes</b> , and if an independent variable is numerical, the Variable Selection operator determines the score for the original variable plus a series of power transforms.  The power transforms currently supported are: <ul style="list-style-type: none"> <li>• exponent</li> <li>• natural log</li> <li>• square</li> <li>• square root</li> <li>• inverse</li> </ul>

Parameter	Description
	<p>If data in a column does not make sense for a particular transform, that transform's score is not determined. For example, if a column includes negative numbers, we do not determine the score for natural log for that variable.</p> <p>Power transforms are not performed on <b>Always Included Columns</b>, unless that column is also selected in <b>Columns</b>.</p> <p>Default value: <b>No</b>.</p>
<b>Variable Mask Directory</b>	Directory where the variable mask is stored. This variable mask is a file that stores which columns (with what transforms) should be passed to subsequent operators.
<b>Variable Mask Name</b>	File name where the variable mask is stored.

## Output

### Visual Output

Column name, R2 or info gain, and status values, as shown in the following table.

Status	Meaning
Approved	R2 or info gain value is above the set threshold for this variable.
Mandatory	R2 or info gain value is not above the set threshold, but this column is in the "always included" list.
Rejected	R2 or info gain value is not above the set threshold.
Selected	If checked, this variable is added to the variable mask.

For R2

Results - Variable Selection - Hadoop			
Column name	R2	Status	Selected
debt_ratio	0.0006	Rejected	
monthly_income	0.0046	Rejected	
revolving_util	0.0226	Rejected	
srsdlqncy	0.0085	Rejected	
times30dayslate_2years	0.0056	Rejected	
times90dayslate	0.0834	Approved	✓

For info gain

VARIABLE SELECTION (?)				
Results - Variable Selection - Hadoop				
Column name	Info gain	Info gain ratio	Status	Selected
times90dayslate	0.0766	0.13	Rejected	
revolving_util	0.0396	0.0067	Rejected	
debt_ratio	0.1439	0.0148	Approved	✓
monthly_income	0.3885	0.0335	Approved	✓
times30dayslate_2years	0.0054	0.0097	Rejected	
srsdlqncy	0.0069	0.0259	Rejected	

## Data Output

The subset of suggested variables from the Variable Selection operator can be passed to the Naive Bayes, Linear Regression, and Logistic Regression operators. These operators use the results of the variable selection to automatically choose the appropriate independent values.

## Transformation Operators

Transformation operators provide different ways to prepare data for modeling.

## Aggregation (DB)

Performs aggregate calculations on a data set by specifying a group-by configuration and an aggregate expression.



### Information at a Glance

**i Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

**i Note:** The Aggregation (DB) operator is for database data only. For Hadoop data, use the [Aggregation \(HD\)](#) operator.

The Aggregation operator also supports the ability to define window fields.

### Input

A data set from the preceding operator.

## Bad or Missing Values

Null values are not allowed and result in an error.

**i Note:** Before using this operator, you can filter out null values or use the [Null Value Replacement \(DB\)](#) operator to replace them with a compatible value.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Aggregates</b>	Specify the <b>Group By</b> and <b>Aggregators</b> fields.  To display the Define Column Aggregations dialog, click <b>Define Aggregations</b> .
<b>Window Field List</b>	Specifies window columns, associated window function expressions, and the window OVER specifications for the aggregate data set.  A window column is calculated from aggregate data by assessing the values for all other data source columns first, then walking through the results and, based on the groups of records, calculating a new value for each group. This is helpful, for example, when sorting data by percent of the total or ranking data groups.  To display the Window Column Configuration dialog, click <b>Window Field List</b> . For more information, see <a href="#">Window Column Configuration dialog</a> .
<b>Output Type</b>	<ul style="list-style-type: none"> <li>• <b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li>• <b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.

Parameter	Description
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	Advanced database settings for the operator output. Available only for <b>TABLE</b> output.  See <a href="#">Storage Parameters dialog</a> for more information.
<b>Drop If Exists</b>	Specifies whether to overwrite an existing table. <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

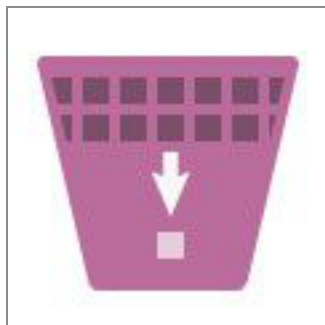
The data rows of the output table/view displayed (up to 2000 rows of the data).

### Data Output

A new data set.

## Aggregation (HD)

Performs aggregate calculations on a data set by specifying a group-by configuration and an aggregate expression.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Pig



**Note:** the Aggregation (HD) operator is for Hadoop data only. For database data, use the [Aggregation \(DB\)](#) operator.

## Input

A data set from the preceding operator.

### Bad or Missing Values

Null values are not allowed and result in an error.



**Note:** Before using this operator, you can filter out null values or use the [Null Value Replacement \(HD\)](#) operator to replace them with a compatible value.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Aggregates</b>	Specify the <b>Group By</b> and <b>Aggregators</b> fields.  To display the <b>Define Aggregations</b> dialog, click Define Aggregates.

<b>Store Results?</b>	<p>Specifies whether to store the results.</p> <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	<p>The HDFS directory where the results of the operator are stored. This is the main directory, the subdirectory of which is specified in <b>Results Name</b>. Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.</p>
<b>Results Name</b>	<p>The name of the file in which to store the results.</p>
<b>Overwrite</b>	<p>Specifies whether to delete existing data at that path and file name.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Use Spark</b>	<p>If <b>Yes</b> (the default), uses Spark to optimize calculation time.</p>



**Advanced  
Spark Settings  
Automatic  
Optimization**

- **Yes** specifies using the default Spark optimization settings.
- **No** enables providing customized Spark optimization. Click **Edit Settings** to customize Spark optimization. See [Advanced Settings dialog](#) for more information.

## Output

### Visual Output

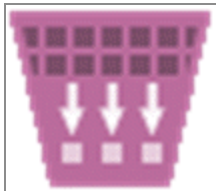
The data rows of the output table/view displayed (up to 2000 rows of the data).

### Data Output

A new data set.

## Batch Aggregation

Performs aggregations on multiple columns.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

## Input

One data set from HDFS. The operator requires at least some numeric columns on which to compute the aggregations. It can include a group by column of any type. For example, you can use an input data set of counties and demographic information to get some aggregations about counties by state.

### Bad or Missing Values

**Dirty data:** When parsing delimited data, the Batch Aggregation operator removes dirty data (such as strings in numeric columns, doubles in integer columns, or rows with the incorrect number of values) as it parses. These "dirty" rows are silently removed because Spark is incapable of handling them.

**Null values:** Before calculating any of the aggregations, the operator filters any rows that contain null values in the group by column, or in any of the columns selected for aggregations. The operator then processes these rows with null values according to the value of the **Write Rows Removed Due to Null Data To File** parameter. The number of rows removed due to null data are reported in the **Summary** tab of the visual output.

## Restrictions

**Median with many distinct values in group by:** This operator might not be able to calculate medians if the group by column has many distinct values. Specifically, if there are more distinct values in the group by column than fit in driver memory, the operator might fail with an out-of-memory exception. The default driver memory in Spark is set to 1024 MB, so if your input data has more than 1 million distinct values in the group-by column(s), you might need to increase the driver memory using the Advanced Spark Settings dialog on the operator configuration screen. Because the group-by columns are stored as strings, reducing the size of each value in the group by column might increase this limit.

**Wide data:** The operator is very performant on long data, but performance might slow down dramatically if aggregations are calculated on thousands of columns. Increasing Spark's executor memory might improve performance.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Group By</b>	Columns by which to group the data.
<b>Find Maximum</b>	Maximum value for each of these columns for each group.
<b>Find Minimum</b>	Minimum value for each of these columns for each group.
<b>Calculate Sum</b>	Sum for each of these columns for each group.
<b>Calculate Mean</b>	Mean value for each of these columns for each group.
<b>Calculate Variance</b>	Variance for each of these columns for each group. See <a href="#">Aggregation Methods for Batch Aggregation</a> for implementation and performance details.
<b>Calculate Standard Deviation</b>	Standard deviation for each of these columns for each group. See <a href="#">Aggregation Methods for Batch Aggregation</a> for implementation and performance details.
<b>Calculate Number of Distinct (slower)</b>	Number of distinct values (excluding null values) for each of these columns for each group. See <a href="#">Aggregation Methods for Batch Aggregation</a> for implementation and performance details.
<b>Calculate Median (slower)</b>	Median for each of these columns for each group. See <a href="#">Aggregation Methods for Batch Aggregation</a> for implementation and performance details.
<b>Use Approximate Median</b>	Specify whether to use approximate median - <b>Yes</b> or <b>No</b> (the default).
<b>Column Name</b>	Indicates whether the aggregation type is added to the beginning or the

Parameter	Description
<b>Format</b>	<p>end of the column name in the output.</p> <p>Default value: <b>suffix</b>. For example, a column that calculates the mean for a set of ages would be titled 'age_mean'.</p>
<b>Write Rows Removed Due to Null Data To File</b>	<p>Rows with null values are removed from the analysis. This parameter allows you to specify that the data with null values is written to a file. The file is written to the same directory as the rest of the output. The file name is suffixed with <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI.</li> <li>• <b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Write Up to 1000 Null Rows to File</b> - remove null value data and write the first 1000 rows of that data to the external file.</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

The operator returns visual output with three tabs: **Output**, **Parameters**, and **Summary**.

**Output** - A preview of the data output (see the Data Output section).

**Parameters** - A list of the parameters that were selected, as shown in the following example.

<b>Aggregations selected</b>	
Group by:	county,state
Min cols:	
Max cols:	
Sum cols:	
Mean cols:	pop, democrat, republican
Median cols:	
Variance cols:	
Standard div cols:	
Distinct:	
<b>Spark Parameters</b>	
Storage Level:	MEMORY_AND_DISK
Number of Partitions:	Default (number of input partitions)
Driver MB	Default
Executor MB	Default
Number of Executor Cores	Default
Number of Executors	Default

**Summary** - A summary of the number of rows removed due to null data and a message about where the results were stored.

Input data size:	249 rows
Input size after removing rows due to null values:	247 rows
Rows removed due to null values:	2 rows (1%)
The output of the operator is stored at /user/alpine_out/allison/DocTests/Batch_Aggregation_1464032809388	

## Data Output

A wide data set that can be accepted by any TIBCO Data Science - Team Studio operator that accepts HDFS data as input. Each of the group by columns is in the output. The group\_size column shows the number of non-null values in that group. For each aggregation type, there is a column for each input column selected.

Suppose that we had used a data set about election data across counties and states, selecting the "suffix" option and the aggregations shown above. Our new data set would have the following structure:

county	state	group_size	pop_mean	democrat_mean	republican_mean
Hancock	GA	1	8,908	78	16
Cleveland	AR	1	7,781	56	33.40000153
Webster	KY	1	13,955	59.70000076	24.89999962
Macon	IL	1	117,206	49.40000153	33.59999847
Sumter	GA	1	30,228	49	39.40000153
Houghton	MI	1	35,446	43.20000076	36.70000076

As you can see, each of the aggregation columns is named with the original column name + underscore + a suffix that describes the type of aggregation performed. If we had used the "prefix" option, the aggregation would come first; for example, "max\_pop". See the Available Aggregation Methods table above for the exact abbreviations used for each aggregation. The columns are ordered by aggregation and then by input column, thus all the mean columns are grouped together, and so on. The aggregations are listed in the same order as the parameters and are computed after the rows with null values are removed.

As this example demonstrates, the groups are not in alphabetically sorted order. To order the aggregations, connect this operator to a sorting operator.

## Data Output

A data set that contains the aggregated values as specified.

## Collapse

Transforms the data contained in a column of a table by means of subtotals (or other calculations) that are defined by another column in the same list. The other calculations might be averages and counts. The result is a collapsed or condensed data set.



### Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce

### Algorithm

For typical data entry and storage, data usually appears in flat tables, meaning that it consists of only columns and rows. While such data can contain a lot of information, it can be difficult to get summarized information. A collapsed table can help quickly summarize the flat data, giving it depth and highlighting the desired information by combining data from multiple rows into one row. The output format for a collapsed column is "sparse."

For example, see the following data set.

id	ice_cream_flavor	number_of_scoops
1	chocolate	2
2	vanilla	1
1	vanilla	3
3	strawberry	1
3	vanilla	1
1	chocolate	3

To determine the average number of scoops per flavor a customer orders, group by **id**, and aggregate "average" with the aggregation column **number\_of\_scoops**. The resulting data set is shown below.

id	ice_cream_flavor_collapsed
1	{{"vanilla":"3.0","chocolate":"2.5"}}
2	{{"vanilla":"1.0"}}
3	{{"vanilla":"1.0","strawberry":"1.0"}}

To just determine how many times a customer ordered each flavor, group by **id** and aggregate "count" (no aggregation column is necessary). The resulting data set is shown below.

id	ice_cream_flavor_collapsed
1	{{"vanilla":"1","chocolate":"2"}}
2	{{"vanilla":"1"}}
3	{{"vanilla":"1","strawberry":"1"}}

This is similar to the [Pivot \(DB\)](#) operator, except that the end results are stored in one column instead of pivoted out to  $(n-1)$  columns, where  $n$  is the number of categorical values in the pivot/collapse column.

## Input

A data set from the preceding operator.



## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns to Collapse</b>	Define the rules for collapsing columns. See <a href="#">Choose Collapse Columns dialog</a> .
<b>Group By</b>	Select the columns to group. See <a href="#">Select Columns dialog</a> .
<b>Store Results?</b>	Specifies whether to store the results. <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	The HDFS directory where the results of the operator are stored. This is the main directory, the subdirectory of which is specified in <b>Results Name</b> . Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.
<b>Results Name</b>	The name of the file in which to store the results.
<b>Overwrite</b>	Specifies whether to delete existing data at that path and file name. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Compression</b>	Select the type of compression for the output. Available Parquet compression options are the following. <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options are the following.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

The data rows of the output table/view displayed (small sample).

Results - Collapse		
times90dayslate	srsdlqncy	MaxMthlyIncome
1	0	{{3690.096503;0,2919.939911;0,6076.92255;1,1768.788355;1,3308.104026;0,7111.309848;0,6430.881968;0,6420.30...
0	1	{{1366.638079;0,729.0173193;0,1846.799649;0,4144.195635;0,6950.651136;0,3004.383423;0,3963.299074;0,4254.4...
0	0	{{4660.21546;0,3834.775637;0,2787.202613;0,4813.59089;0,4160.09424;0,3289.705291;1,1076.871747;0,899.42907...
3	1	{{1930.496431;1,3200.641145;2,2680.486394;4,5382.982071;2,3077.846982;2,2488.158903;1,4071.060391;1,955.35...
3	0	{{4072.148825;2,4258.695987;1,2928.255419;1,5599.498814;1,1636.37955;2,3013.459309;1}}
2	1	{{3439.44672;1,4793.206374;1,3764.43215;1,3915.844424;1,1658.205236;1,2948.018776;2,3290.05177;0,5212.8036...
2	0	{{3309.801556;1,2334.329207;0,2946.386632;1,2180.05515;0,6756.290624;0,5398.061092;0,5366.002587;0,1855.46...
1	1	{{339.5362409;0,6470.293143;1,2784.51591;0,1859.442683;0,5146.992753;0,2392.416109;2,2888.1029;0,3215.7286...

### Data Output

A data set of the newly created file. The collapsed column is of type "sparse."

## Additional Notes

Sparse columns are typically used with Naive Bayes, SVM, and Association Rules operators. Other operators accept sparse data types, but treat the values as strings.

## Column Filter (DB)

Selects a subset of the columns from data source. Only the columns selected remain in the output dataset.



## Information at a Glance

**Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

This operator is particularly helpful when there are many irrelevant columns in a data source that are not needed for the data analysis workflow.

**Note:** The Column Filter (DB) operator is for database data only. For Hadoop, use the [Column Filter \(HD\)](#) operator.

## Input

A dataset from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	<p>The columns to be made available for analysis.</p> <p>Click <b>Select Columns</b> to open the dialog to specify columns. See <a href="#">Select Columns dialog</a> for more information.</p>
<b>Output Type</b>	<ul style="list-style-type: none"> <li>• <b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li>• <b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	<p>Advanced database settings for the operator output. Available only for <b>TABLE</b> output.</p> <p>See <a href="#">Storage Parameters dialog</a> for more information.</p>
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

The data rows of the output table/view displayed (up to 2000 rows of the data).

### Data Output

A dataset of the newly created table or view.

## Column Filter (HD)

Selects a subset of the columns from data source. Only the columns selected remain in the output dataset.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Pig

This operator is particularly helpful when there are many irrelevant columns in a data source that are not needed for the data analysis workflow.

**Note:** The Column Filter (HD) operator is for Hadoop data only. For database data, use the [Column Filter \(DB\)](#) operator.

## Input

A dataset from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	The columns to be made available for analysis.  Click <b>Select Columns</b> to open the dialog to specify columns. See <a href="#">Select Columns dialog</a> for more information.
<b>Store Results?</b>	Specifies whether to store the results. <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	The HDFS directory where the results of the operator are stored. This is the main directory, the subdirectory of which is specified in <b>Results Name</b> . Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.
<b>Results Name</b>	The name of the file in which to store the results.
<b>Overwrite</b>	Specifies whether to delete existing data at that path and file name. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Storage Format</b>	Select the format in which to store the results. The storage format is determined by your type of operator.  Typical formats are <b>Avro</b> , <b>CSV</b> , <b>TSV</b> , or <b>Parquet</b> .
<b>Compression</b>	Select the type of compression for the output.

Parameter	Description
	<p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

The data rows of the output table/view displayed (up to 2000 rows of the data).

### Data Output

A dataset of the newly-created Hadoop file.

## Correlation Filter (DB)

Filters numeric columns so the remaining columns are not correlated strongly with each other.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes <sup>1</sup>
Data processing tool	SQL



**Note:** The Correlation Filter (DB) operator is for database data only. For Hadoop data, use the [Correlation Filter \(HD\)](#) operator.

## Input

The input data is a database source. You can choose the columns from which you want distinct combinations, and the operator performs the calculation.

## Restrictions

- If some of the **Columns to Filter** or **Dependent Column** contain null values, these values are skipped to compute the correlations.
- If a column contains only null values, the correlation with any other column is 0.

---

<sup>1</sup>The full output schema is not available until you step run the operator. After you run this operator, the output schema automatically updates, and subsequent operators either validate or turn red, depending on the structure of the output data.



## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns to Filter</b> required	Select two or more numeric columns. Columns selected in this parameter are compared with each other, and columns are removed from this set until all of the remaining columns have correlations under the threshold defined below.
<b>Dependent Column</b> required	Select a numeric column. When determining which columns to remove due to high correlation with another column, the one with higher correlation with the dependent variable is selected.
<b>Correlation Threshold</b> required	Enter a number greater than 0 and less than or equal to 1. This threshold is used to determine whether each pair of columns are considered collinear.
<b>Maximum Number of Filtered Columns</b> required	Enter an integer greater than 0 or -1. If -1, the operator returns all columns whose correlations are under the threshold. If $n > 0$ , the operator returns the top $n$ columns, ranked by their correlation with the dependent variable.
<b>Pass Through Other Columns?</b>	Choose yes to include columns not selected in <b>Columns to Filter</b> in the final results. The <b>Dependent Column</b> is always included.
<b>Output Type</b>	<ul style="list-style-type: none"> <li><b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li><b>VIEW</b> outputs a database view.</li> </ul>
<b>Output</b>	The schema for the output table or view.

Parameter	Description
<b>Schema</b>	
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

The final output schema of the Correlation Filter operator is cleared if one of the following occurs:

- You change the configuration properties of the Correlation Filter
- You change the input connected to the Correlation Filter
- You clear the step run results of the Correlation Filter

In this case, the output schema transmitted to subsequent operators again becomes the partial schema defined at design time (hence, subsequent operators can turn invalid). You must run the Correlation Filter operator again to transmit the new output schema.

## Outputs

### Visual Output

The **Output** tab displays a preview of the output data set.

The **Summary** tab displays information about the parameters selected and the output.

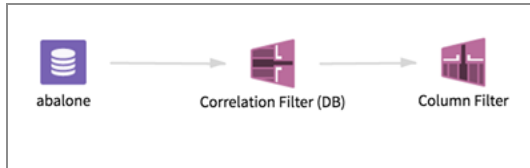
The **Correlation Results** tab displays which columns have been selected with additional details (correlation with dependent variable, reason why columns were not selected).

### Data Output

Output is a data set of the newly created table/view.

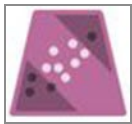
A partial schema can be transmitted to subsequent operators at design time, but you must run the operator for subsequent operators to see the final output schema.

## Example



## Correlation Filter (HD)

Filters numeric columns so the remaining columns are not correlated strongly with each other.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes <sup>1</sup>
Data processing tool	Spark

**Note:** The Correlation Filter (HD) operator is for Hadoop data only. For database data, use the [Correlation Filter \(DB\)](#) operator.

<sup>1</sup>The full output schema is not available until you step run the operator. After you run this operator, the output schema automatically updates, and subsequent operators either validate or turn red, depending on the structure of the output data.

## Input

A file on HDFS. You can choose the columns you want distinct combinations from, and the operator performs the calculation.

### Bad or Missing Values

If a row of the input contains null values in at least one of the selected **Columns to Filter**, the entire row is skipped before computing the correlation matrix.

After the columns to keep are determined based on the correlations, the input null values from the input are conserved in the output data set for the columns concerned.

This is a different behavior from the [Correlation Filter \(DB\)](#) of this operator. Replace null values before running this operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns to Filter</b> *required	Select two or more numeric columns. Columns selected in this parameter are compared with each other, and columns are removed from this set until all of the remaining columns have correlations under the threshold defined below.
<b>Dependent Column</b> *required	Select a numeric column. When determining which columns to remove due to high correlation with another column, the one with higher correlation with the dependent variable is selected.
<b>Correlation Threshold</b> *required	Enter a number greater than 0 and less than or equal to 1. This threshold is used to determine whether each pair of columns are considered collinear.
<b>Maximum Number of</b>	Enter an integer greater than 0 or -1. If -1, the operator returns all columns whose correlations are under the threshold. If $n > 0$ , the operator returns

Parameter	Description
<b>Filtered Columns</b>  *required	the top $n$ columns, ranked by their correlation with the dependent variable.
<b>Pass Through Other Columns?</b>	Choose yes to include columns not selected in <b>Columns to Filter</b> in the final results. The <b>Dependent Column</b> is always included.
<b>Correlation Method</b>	Choose the correlation method to compute. Supported methods are Pearson or Spearman correlations. <div> <b>Note:</b> The Pearson correlation coefficient is the most widely used. It measures the strength of the linear relationship between normally distributed variables. When the variables are not normally distributed or the relationship between the variables is not linear, it might be more appropriate to use the Spearman rank correlation method.           </div>
<b>Write Rows Removed Due to Null Data To File</b>	<p>Rows with at least one null value in the <b>Columns to Filter</b> are skipped during the correlation analysis (but kept in the output). This parameter allows you to specify if rows with null values are written to a file.</p> <p>The file is written to the same directory as the rest of the output. The filename is suffixed with <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI.</li> <li>• <b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Write Up to 1000 Null Rows to File</b> - remove null value data and write the first 1000 rows of that data to the external file.</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	Select the format in which to store the results. The storage format is determined by your type of operator.

Parameter	Description
	Typical formats are <b>Avro</b> , <b>CSV</b> , <b>TSV</b> , or <b>Parquet</b> .
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

- The **Output** tab displays a preview of the output data set.

- The **Summary** tab displays information about the selected parameters and the output.
- The **Correlation Results** tab displays which columns have been selected with additional details (correlation with dependent variable, reason why columns were not selected).

## Data Output

The data set created with filtered columns.

**i Note:** A partial schema can be transmitted to subsequent operators at design time, but the user must run the operator for subsequent operators to see the final output schema.

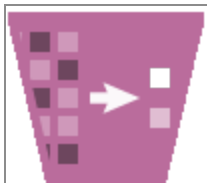
The final output schema of the Correlation Filter operator is cleared if one of the following occurs:

- The user changes the configuration properties of the Correlation Filter.
- The user changes the input connected to the Correlation Filter.
- The user clears the step run results of the Correlation Filter.

In this case, the output schema transmitted to subsequent operators again becomes the partial schema defined at design time (hence, subsequent operators can turn invalid), and the user must run the Correlation Filter operator again to transmit the new output schema.

## Distinct (DB)

Returns only distinct combinations of values from specified columns of a database source. Rows are not returned in any particular order, but each combination of values within a row is distinct from other rows.



## Information at a Glance

**i Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	SQL

**i Note:** The Distinct (DB) operator is for database data only. For Hadoop data, use the [Distinct \(HD\)](#) operator.

## Input

A database source. Users choose the columns from which they want distinct combinations, and the operator performs the calculation.

### Bad or Missing Values

Missing values are considered as part of determination of distinct values. If a column has a missing value, a missing value is considered distinct from a value.

This operator handles null values by eliminating them from the input calculation. To prevent this behavior, use the [Null Value Replacement \(DB\)](#) operator on the initial training data to replace bad or missing values.



## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Distinct Columns</b>	Select one or more columns from the data source by which to generate rows of data, where each row has a distinct combination of column values.  *required
<b>Output Type</b>	<ul style="list-style-type: none"> <li>• <b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li>• <b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

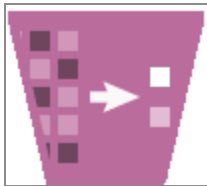
### Data Output

A subset of data with only selected columns, and each row only distinct combinations of values in those columns.

CompanyLvlDesc	LocationLvlDesc	DepartmentDesc
Discontinued Operations	Joseph Webb (CA)	N/A
Windy City Distributing, LLC	Aurora - WCD	Transportation
Discontinued Operations	Pedi Bros (IL)	N/A
Windy City Distributing, LLC	Aurora - WCD	Warehouse
Discontinued Operations	Ultra Product (AZ)	N/A
Windy City Distributing, LLC	Aurora - WCD	Transportation (Drivers)
Premium Distributors of Washington DC, LLC	Washington, DC	Transportation
Premium Distributors of Washington DC, LLC	Washington, DC	N/A

## Distinct (HD)

Returns the unique value combinations across selected columns.



### Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce

**i Note:** The Distinct (HD) operator is for Hadoop data only. For database data, use the [Distinct \(DB\)](#) operator.

Removing duplicate records is often a required step before data analysis and modeling can begin. Any record with the same values for the field (or selected fields) is considered a duplicate, and the Distinct operator can be used to remove these entries.

An example use case might be applied to a marketing database in which an individual's data appears several times with different addresses or company information.

## Input

A Hadoop data set from the preceding operator.

### Bad or Missing Values

This operator handles null values by eliminating them from the input calculation. To prevent this behavior, use the [Null Value Replacement \(DB\)](#) operator on the initial training data to replace bad or missing values.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Column Names in Order</b>	Specify the column or columns to use as the unique value combination criteria. At least one column must be selected.
<b>Store Results?</b>	Specifies whether to store the results. <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	The HDFS directory where the results of the operator are stored. This is the main directory, the subdirectory of which is specified in <b>Results Name</b> . Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.
<b>Results Name</b>	The name of the file in which to store the results.

Parameter	Description
<b>Overwrite</b>	<p>Specifies whether to delete existing data at that path and file name.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

The data rows of the output table or view displayed (up to 100 rows of the data).

In the following example, three columns (iyear, imonth, and iday) were selected as the distinct criteria. Each results row has a unique combination of values for the three data columns.

RESULTS - Distinct		
First 100 rows are being displayed.		
iyear	imonth	iday
1,970	1	1
1,970	1	25
1,970	3	27
1,970	4	2
1,970	4	28

### Data Output

The data set of the distinct data table.

## Fuzzy Join

Performs a fuzzy matching join to connect two data sets based on nearly matching string values.



### Information at a Glance

Parameter	Description
Category	Transform

Parameter	Description
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

## Input

Exactly two HDFS tabular inputs.

### Bad or Missing Values

If the output contains empty strings in the first column, these empty strings are converted to a double quote instead (and are not detected as empty strings by subsequent operators).

This issue is due to a bug in spark-csv for Spark version 1.6.1, where the CSV writer always quotes empty strings if they are first in a row (see <https://issues.apache.org/jira/browse/CSV-63> for details). To work around it, replace all empty strings/null values in the first column in the Fuzzy Join output before you run it.

## Restrictions

This operator accepts two inputs only.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Left data set</b>	The left data set input for the join.
*required	

Parameter	Description
<b>Join On Column</b> *required	The joining column from the <b>Left data set</b> . This column must be categorical (chararray).
<b>Columns To Keep</b>	The columns to keep from the <b>Left data set</b> .
<b>Right data set</b>	The right data set input for the join.
<b>Join On Column</b> *required	The joining column from the <b>Right data set</b> . This column must be categorical (chararray).
<b>Columns to Keep</b>	The columns to keep from the <b>Right data set</b> .
<b>Join Type</b> *required	<p>The join type.</p> <ul style="list-style-type: none"> <li>• <b>Inner</b></li> <li>• <b>Left Outer</b></li> <li>• <b>Right Outer</b></li> <li>• <b>Full Outer</b></li> </ul>
<b>Pre-Processing on Columns to Match</b>	<p>Select zero or more text pre-processing options to apply to the two <b>Join On</b> columns selected before executing the fuzzy join.</p> <ul style="list-style-type: none"> <li>• <b>Normalize text</b> (trim, strip diacritics and accents, transform to lower case)</li> <li>• <b>Remove Punctuation</b></li> <li>• <b>Stem Words</b> (using Snowball Stemmer from open Apache open-nlp tools: <a href="#">Open NLP - Snowball Stemmer documentation v 1.6.0</a>)</li> <li>• <b>Clear Stop Words</b> (from default list or custom list of stop words; see <b>Custom Stop Words File</b> parameter below)</li> <li>• <b>Sort Words Alphabetically</b> (words are considered distinct when separated by a space)</li> </ul>
<b>Custom Stop Words File</b>	If the <b>Clear Stop Words</b> option is selected in <b>Pre-Processing on Columns to Match</b> and this box is left blank, a standard set of stop

Parameter	Description
	<p>words is used. You can find this list <a href="#">Stop Words</a>.</p> <p>Otherwise, choose a file that contains a list of the stop words. This list must be one word per line and should be small enough to fit in memory.</p> <p><b>Note:</b> If additional pre-processing transformations are selected, these transformations are applied to the list of stop words as well.</p>
<b>Stemmer Algorithm</b>	<p>Define the stemming algorithm to use if the <b>Stem Words</b> option is selected in <b>Pre-Processing on Columns to Match</b>.</p> <ul style="list-style-type: none"> <li>• English (the default)</li> <li>• Porter</li> <li>• French</li> <li>• German</li> <li>• Italian</li> <li>• Russian</li> <li>• Dutch</li> <li>• Finnish</li> <li>• Swedish</li> <li>• Spanish</li> </ul> <p>If you use a language other than English, you must add a custom stop word file.</p>
<b>Match Threshold (%)</b> *required	<p>Select the minimum match threshold (inclusive) between the (pre-processed or not) two strings so that they are considered a match. It must be a Double in [0,100].</p> <p>The match threshold between strings A and B is computed using the formula: <math>100.0 - (100 * \text{Damerau-Levenshtein distance}(A, B) / \text{max\_length}(A, B))</math>.</p> <p>If both values A and B are null or empty (that is, if <math>\text{max\_length}(A, B) =</math></p>



Parameter	Description
	<p>0), the match score is 100. Thus, they are considered a match.</p> <p>For details, see <a href="#">Wikipedia: Damerau-Levenshtein distance</a>.</p>
<b>Suffix for Duplicate Columns in Right data set</b>  *required	<p>If some columns to keep from left and right data sets have the same name, this suffix is appended (preceded by an underscore) to the output columns of the right data set.</p>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>

Parameter	Description
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

**Output** - A preview of the data output.

Results - Fuzzy Join-1					
<b>Output</b> <a href="#">Summary</a>	title	rating	press	title_bis	duration
	inglorious nasterds	4.5	4.2	inglorious basterds	120
	500 days summer	2	N/A	500 days summer	110
	maids room	3	3	maids room	94
	wolf wal stret	2	2	wolf wall street	101
	django unchain	4.3	4.5	django unchained	98
	wolfrine	1.2	1	wolverine	95
	lord strings	3	2	lord rings	102

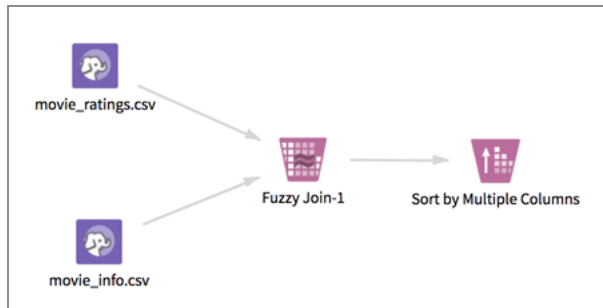
**Summary** - A summary of the selected parameters and a message that provides the results location.

### Data Output

The joined Hadoop data set with columns to keep selected from each input, plus the match score column (match\_score).

**i Note:** If some pre-processing operations are selected on the join columns, the transformed columns are present in the output (and not the initial join input columns).

## Example



## Join (DB)

Performs a table join on the input data sets by allowing users to define the input data set alias, the output columns, and the join condition.



## Information at a Glance

**i Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

**i Note:** The Join (DB) operator is for database data only. For Hadoop data, use the [Join \(HD\)](#) operator.

## Input

Accepts two or more data sets.

**i Note:** Both tables must be located in the same database. Join does not work on tables located in different databases. See the Join (DB) for any data source exceptions for the Join operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Create Sequence ID</b>	Click <b>Yes</b> to create an ID column on the output data set of the Join operator.
<b>Join Conditions</b>	Click <b>Set Table Join Parameters</b> to display the Join Properties dialog. For information about the options, see <a href="#">Join Properties - Database dialog</a> . For information about creating the Join condition, see <a href="#">Creating a Join condition for a database join</a> .
<b>Output Type</b>	<ul style="list-style-type: none"> <li><b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li><b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is

Parameter	Description
	generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	Advanced database settings for the operator output. Available only for <b>TABLE</b> output.  See <a href="#">Storage Parameters dialog</a> for more information.
<b>Drop If Exists</b>	Specifies whether to overwrite an existing table. <ul style="list-style-type: none"> <li><b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li><b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

The resulting joined data rows of the output table or view are displayed in the Results window (up to 200 rows of the data):

JOIN (?) <span>Edit Operator</span> <span>Copy</span>												
Results - Join-Hadoop												
id_creditcov	times90dayslate	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslate	snsolvency	id_demographic	name	age	num_dep	edu
99,947	0	0.28	0.21	7	3,729.875112	0	0	99,947	Ana Vaughn	33	3	5
99,695	0	0.27	0.01	5	3,697.607768	0	0	99,695	Lawrence Curry	36	2	1
99,596	0	0.31	0.324	5	5,187.319526	0	0	99,596	Christy Payne	30	0	1
99,587	0	0.32	0.259	4	5,349.321331	0	0	99,587	Christopher Richardson	42	1	4
99,589	0	0.29	0.214	6	4,800.799297	0	0	99,589	Bethany Fowler	24	0	2
99,550	0	0.49	0.108	4	2,698.400004	0	0	99,550	Caitlin Gibson	34	0	5
99,362	1	0.49	0.139	3	5,357.889993	0	0	99,362	Wayne Reese	26	2	1
99,316	1	0.57	0.673	5	2,950.600061	1	0	99,316	Corey Watson	22	1	4
99,226	0	0.13	0.429	8	3,960.838628	0	0	99,226	Allison Hamilton	35	1	4
99,217	0	0.53	0.779	3	2,528.229184	0	0	99,217	Tiffany Alexander	32	3	3
99,073	0	0.3	0.203	3	4,399.007364	1	0	99,073	Robert Stewart	36	2	3
99,894	0	0.1	0.196	4	7,699.02001	0	0	99,894	Austin Cohen	26	0	6
99,876	0	0.67	0.57	4	1,632.331944	0	0	99,876	Andrea Rivera	25	2	1
99,696	0	0.26	0.185	6	2,594.574568	0	0	99,696	Veronica Yates	33	3	5
99,678	0	0.06	0.825	7	3,365.434273	0	0	99,678	Dylan Curry	33	1	1
99,524	0	0.89	0.537	4	4,262.990298	0	0	99,524	Molly Frazier	31	3	4
99,317	0	0.12	0.202	4	2,052.369028	0	0	99,317	Jonathan Douglas	39	2	3

### Data Output

A data set of the output table or view of the joined data set.

## Join (HD)

Performs a table join on the input data sets by allowing users to define the input data set alias, the output columns, and the join condition.



### Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Pig, MapReduce



**Note:** The Join (HD) operator is for Hadoop data only. For database data, use the [Join \(DB\)](#) operator.

### Input

Accepts two data sets.

### Configuration

Parameter	Description
Notes	Notes or helpful information about this operator's parameter settings. When

Parameter	Description
	you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Join Conditions</b>	<p>Click <b>Define Join Conditions</b> to select the appropriate columns for joining the data and to specify the join type and output fields.</p> <p>For more information, see <a href="#">Define Join Conditions dialog (Hadoop)</a>.</p>
<b>Store Results?</b>	<p>Specifies whether to store the results.</p> <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	<p>The HDFS directory where the results of the operator are stored. This is the main directory, the subdirectory of which is specified in <b>Results Name</b>. Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.</p>
<b>Results Name</b>	The name of the file in which to store the results.
<b>Overwrite</b>	<p>Specifies whether to delete existing data at that path and file name.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> </ul>

- no compression

Available Avro compression options.

- **Deflate**
- **Snappy**
- no compression

### Use Spark

If **Yes** (the default), uses Spark to optimize calculation time.

### Advanced Spark Settings Automatic Optimization

- **Yes** specifies using the default Spark optimization settings.
- **No** enables providing customized Spark optimization. Click **Edit Settings** to customize Spark optimization. See [Advanced Settings dialog](#) for more information.

## Output

### Visual Output

The resulting joined data rows of the output table or view are displayed in the Results window (up to 200 rows of the data):

JOIN (?)													Edit Operator	Copy
Results - Join-Hadoop														
id_creditcv	times90dayslate	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslate	unsolvency	id_demographic	name	age	num_dep	edu		
99,947	0	0.28	0.21	7	3,729,875,112	0	0	99,947	Ana Vaughn	33	3	5		
99,895	0	0.27	0.01	5	3,897,607,798	0	0	99,895	Lawrence Curry	36	2	1		
99,596	0	0.31	0.324	5	5,187,919,926	0	0	99,596	Christy Payne	30	0	1		
99,587	0	0.32	0.259	4	5,349,321,131	0	0	99,587	Christopher Richardson	42	1	4		
99,569	0	0.29	0.214	6	4,800,799,297	0	0	99,569	Bethany Fowler	24	0	2		
99,550	0	0.49	0.138	4	2,898,460,004	0	0	99,550	Calvin Gibson	34	0	5		
99,361	1	0.49	0.139	3	5,357,889,993	0	0	99,361	Wayne Reese	26	2	1		
99,316	1	0.57	0.673	5	2,950,600,901	1	0	99,316	Corey Watson	22	1	4		
99,226	0	0.13	0.429	8	3,660,638,828	0	0	99,226	Alison Hamilton	35	1	4		
99,217	0	0.53	0.779	3	2,328,229,084	0	0	99,217	Tiffany Alexander	32	3	3		
99,073	0	0.3	0.203	3	4,399,007,364	1	0	99,073	Robert Stewart	36	2	3		
99,894	0	0.1	0.196	4	7,889,000,001	0	0	99,894	Austin Cohen	26	0	6		
99,876	0	0.67	0.57	4	1,632,331,944	0	0	99,876	Andrea Rivera	25	2	1		
99,696	0	0.26	0.185	6	2,594,574,958	0	0	99,696	Veronica Yates	33	3	5		
99,678	0	0.06	0.815	7	3,365,834,173	0	0	99,678	Dylan Curry	33	1	1		
99,524	0	0.89	0.537	4	4,282,990,298	0	0	99,524	Molly Frazier	31	3	4		
99,317	0	0.12	0.202	4	2,952,369,028	0	0	99,317	Jonathan Douglas	39	2	3		

### Data Output

A data set of the output table or view of the joined data set.



## Normalization (DB)

Performs normalization on the selected columns of the input data set. Normalization means adjusting values measured on different scales to a notionally common scale.



### Information at a Glance

**i Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

**i Note:** The Normalization (DB) operator is for database data only. For Hadoop data, use the [Normalization \(HD\)](#) operator.

### Algorithm

You can accomplish normalization in various ways.

- By specifying a user-defined minimum and maximum value.
- By a z-transformation (for example, on mean 0 and variance 1).

- By a transformation as proportion of the average or sum of the respective attribute.

Your selection translates into four possible types of normalization to select.

- Z-Transformation.
- Proportion Transformation.
- Range Transformation.
- Divide-By-Average Transformation.

See Method under Configuration for a definition of each type.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Method</b>	<p>Normalization method to use.</p> <p>Options:</p> <ul style="list-style-type: none"> <li>• <b>Divide-By-Average Transformation:</b> calculate by sample's average.</li> <li>• <b>Proportional Transformation:</b> calculate by sample's sum.</li> <li>• <b>Z-Transformation:</b> calculate by sample's mean and variance.</li> <li>• <b>Range Transformation:</b> calculate by sample's Min and Max value.</li> </ul>
<b>Range Minimum</b>	Specify the minimum value in Range transformation.
<b>Range Maximum</b>	Specify the maximum value in Range transformation.

Parameter	Description
<b>Columns</b>	Click <b>Column Names</b> to open the dialog for selecting the available numerical columns for the columns to normalize.
<b>Output Type</b>	<ul style="list-style-type: none"> <li>• <b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li>• <b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	<p>Advanced database settings for the operator output. Available only for <b>TABLE</b> output.</p> <p>See <a href="#">Storage Parameters dialog</a> for more information.</p>
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

The data rows of the output table or view displayed (up to 200 rows of the data).

Results - Normalization - DB										
times90dayslat	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslat	srslqncy	id	age	num_dep	edu
1	0.28	0.056	0.0376	3,493.1239	0	0	2,172	38	3	1
1	0.66	0.184	-1.4846	3,323.402	0	1	3,764	35	0	4
1	0.31	1.24	-1.4846	6,029.8735	0	1	6,523	22	2	2
1	0.51	0.041	-1.4846	3,423.218	0	0	9,930	33	2	3
1	0.43	0.186	0.0376	4,206.4527	0	0	11,932	35	2	6
1	0.64	0.055	0.0376	3,633.0517	0	0	15,757	44	1	5
1	0.84	0.064	0.7986	4,517.9835	0	0	17,970	24	2	1
1	0.5	0.356	-2.2456	2,490.3712	0	1	19,566	19	1	2
1	0.49	0.046	0.0376	2,859.6002	0	0	21,518	20	1	4
1	0.81	0.193	-0.7235	3,042.5369	0	0	24,563	36	2	2
1	0.51	0.116	0.0376	3,220.8254	0	0	26,782	36	1	4
1	0.65	0.074	0.0376	4,291.8507	1	0	28,830	25	0	3
1	0.35	0.553	-0.7235	3,331.2712	1	0	30,669	22	0	3
1	0.55	0.122	-0.7235	3,997.277	0	0	31,630	35	3	5
1	0.51	0.337	-0.7235	4,293.0874	0	0	34,437	23	1	5
1	0.52	0.494	-1.4846	2,613.0579	1	0	40,076	30	1	2
1	0.41	0.066	0.0376	3,504.0965	0	1	41,822	18	0	3

## Data Output

The data set of the normalized data.

## Normalization (HD)

Performs normalization on the selected columns of the input data set. Normalization means adjusting values measured on different scales to a notionally common scale.



## Information at a Glance

Parameter	Description
Category	Transform

Parameter	Description
Data source type	HD
Send output to other operators	Yes
Data processing tool	Pig



**Note:** The Normalization (HD) operator is for Hadoop data only. For database data, use the [Normalization \(DB\)](#) operator.

## Algorithm

You can accomplish normalization in various ways.

- By specifying a user-defined minimum and maximum value.
- By a z-transformation (for example, on mean 0 and variance 1).
- By a transformation as proportion of the average or sum of the respective attribute.

Your selection translates into four possible types of normalization to select.

- Z-Transformation.
- Proportion Transformation.
- Range Transformation.
- Divide-By-Average Transformation.

See Method under Configuration for a definition of each type.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Method</b>	<p>Normalization method to use.</p> <ul style="list-style-type: none"> <li>• <b>Divide-By-Average Transformation:</b> calculate by sample's average.</li> <li>• <b>Proportion Transformation:</b> calculate by sample's sum.</li> <li>• <b>Z-Transformation:</b> calculate by sample's mean and variance.</li> <li>• <b>Range Transformation:</b> calculate by sample's Min and Max value.</li> </ul>
<b>Range Minimum</b>	Specify the minimum value in Range transformation.
<b>Range Maximum</b>	Specify the maximum value in Range transformation.
<b>Columns</b>	Click <b>Select Columns</b> to open the dialog for selecting the available numerical columns for the columns to normalize.
<b>Store Results?</b>	<p>Specifies whether to store the results.</p> <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	<p>The HDFS directory where the results of the operator are stored. This is the main directory, the subdirectory of which is specified in <b>Results Name</b>. Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.</p>
<b>Results Name</b>	The name of the file in which to store the results.

<b>Overwrite</b>	<p>Specifies whether to delete existing data at that path and file name.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Use Spark</b>	<p>If <b>Yes</b> (the default), uses Spark to optimize calculation time.</p>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

The data rows of the output table or view displayed (up to 200 rows of the data).

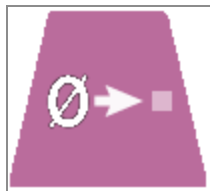
times90dayslat	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslat	srslqncy	id	age	num_dep	edu
1	0.28	0.056	0.0376	3,493.1239	0	0	2,172	38	3	1
1	0.66	0.184	-1.4846	3,323.402	0	1	3,764	35	0	4
1	0.31	1.24	-1.4846	6,029.8735	0	1	6,523	22	2	2
1	0.51	0.041	-1.4846	3,423.218	0	0	9,930	33	2	3
1	0.43	0.186	0.0376	4,206.4527	0	0	11,932	35	2	6
1	0.64	0.055	0.0376	3,633.0517	0	0	15,757	44	1	5
1	0.84	0.064	0.7986	4,517.9835	0	0	17,970	24	2	1
1	0.5	0.356	-2.2456	2,490.3712	0	1	19,566	19	1	2
1	0.49	0.046	0.0376	2,859.6002	0	0	21,518	20	1	4
1	0.81	0.193	-0.7235	3,042.5369	0	0	24,563	36	2	2
1	0.51	0.116	0.0376	3,220.8254	0	0	26,782	36	1	4
1	0.65	0.074	0.0376	4,291.8507	1	0	28,830	25	0	3
1	0.35	0.553	-0.7235	3,331.2712	1	0	30,669	22	0	3
1	0.55	0.122	-0.7235	3,997.277	0	0	31,630	35	3	5
1	0.51	0.337	-0.7235	4,293.0874	0	0	34,437	23	1	5
1	0.52	0.494	-1.4846	2,613.0579	1	0	40,076	30	1	2
1	0.41	0.066	0.0376	3,504.0965	0	1	41,822	18	0	3

## Data Output

The data set of the normalized data.

## Null Value Replacement (DB)

Replaces null values of the selected fields of the data set with designated values. This is helpful as a pre-cleansing data step.



## Information at a Glance



**Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.



Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a



**Note:** The Null Value Replacement (DB) operator is for database data only. For Hadoop data, use the [Null Value Replacement \(HD\)](#) operator.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Replace Null Columns</b>	The columns in which to replace null values. See <a href="#">Null Value Replacement Configuration dialog (DB)</a> for more information.
<b>Output Type</b>	<ul style="list-style-type: none"> <li>• <b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li>• <b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is

Parameter	Description
	generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	Advanced database settings for the operator output. Available only for <b>TABLE</b> output.  See <a href="#">Storage Parameters dialog</a> for more information.
<b>Drop If Exists</b>	Specifies whether to overwrite an existing table. <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

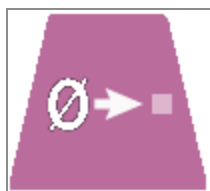
The data rows of the output table or view displayed (up to 2000 rows of the data).

### Data Output

A data set of the newly created table or view.

## Null Value Replacement (HD)

Replaces null values of the selected fields of the data set with designated values. This is helpful as a pre-cleansing data step.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Pig



**Note:** The Null Value Replacement (HD) operator is for Hadoop data only. For database data, use the [Null Value Replacement \(DB\)](#) operator.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Replace Null Columns</b>	The columns in which to replace null values. See <a href="#">Null Value Replacement Configuration dialog (HD)</a> for more information.
<b>Store Results?</b>	Specifies whether to store the results. <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>

<b>Results Location</b>	The HDFS directory where the results of the operator are stored. This is the main directory, the subdirectory of which is specified in <b>Results Name</b> . Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.
<b>Results Name</b>	The name of the file in which to store the results.
<b>Overwrite</b>	Specifies whether to delete existing data at that path and file name. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Storage Format</b>	Select the format in which to store the results. The storage format is determined by your type of operator.  Typical formats are <b>Avro</b> , <b>CSV</b> , <b>TSV</b> , or <b>Parquet</b> .
<b>Compression</b>	Select the type of compression for the output.  Available Parquet compression options. <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> Available Avro compression options. <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

The data rows of the output table or view displayed (up to 2000 rows of the data).

### Data Output

A data set of the newly created table or view.

## Numeric to Text (DB)

Converts a numeric type column to a text type column.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a



**Note:** The Numeric to Text (DB) operator is for database data only. For Hadoop data, use the [Numeric to Text \(HD\)](#) operator.

## Input

A data set from the preceding operator.

### Bad or Missing Values

All null values from the input data set are kept as is.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	Click <b>Select Columns</b> to select the columns to convert to text. Only numeric type columns are shown.
<b>Output Type</b>	<ul style="list-style-type: none"> <li>• <b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li>• <b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	<p>Advanced database settings for the operator output. Available only for <b>TABLE</b> output.</p> <p>See <a href="#">Storage Parameters dialog</a> for more information.</p>
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

A preview of the output table.

(Example: selected "credit\_lines" to convert to text type)

Operation created output with 8 columns. Visualization of rows limited to 20. Please refer to output file for full results.

id	times90dayslate	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslate_2years	srsdlqncy
2	0	0.26	0.324	4	1,956.5922	0	0
6	0	0.25	0.927	6	3,983.4564	0	0
52	0	0.46	0.372	7	3,595.3707	0	0
76	2	0.88	0.154	3	1,729.9645	2	0
84	0	0.14	0.69	7	2,507.1652	0	0
86	0	0.11	0.131	5	2,126.5996	0	0

### Data Output

A data set of the result table or view.

## Numeric to Text (HD)

Converts a numeric type column to a text type column.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	n/a

**Note:** The Numeric to Text (DB) operator is for database data only. For Hadoop data, use the [Null Value Replacement \(DB\)](#) operator.

## Input

A data set from the preceding operator.

## Bad or Missing Values

All null values from the input data set are kept as is.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	Click <b>Select Columns</b> to convert to text. Only numeric type columns are shown.
<b>Storage Format</b>	Select the format in which to store the results. The storage format is determined by your type of operator.  Typical formats are <b>Avro</b> , <b>CSV</b> , <b>TSV</b> , or <b>Parquet</b> .
<b>Compression</b>	Select the type of compression for the output.  Available Parquet compression options. <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> Available Avro compression options. <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> </ul>



Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	Specifies whether to delete existing data at that path. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

A preview of the output table.

(Example: selected "credit\_lines" to convert to text type)

Operation created output with 8 columns. Visualization of rows limited to 20. Please refer to output file for full results.

id	times90dayslate	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslate_2years	srsdlqncy
2	0	0.26	0.324	4	1,956.5922	0	0
6	0	0.25	0.927	6	3,983.4564	0	0
52	0	0.46	0.372	7	3,595.3707	0	0
76	2	0.88	0.154	3	1,729.9645	2	0
84	0	0.14	0.69	7	2,507.1652	0	0
86	0	0.11	0.131	5	2,126.5996	0	0

### Data Output

A data set with the same columns as in the input, but with the selected numeric columns converted to text columns.

## One-Hot Encoding

Performs one-hot encoding on a set of categorical columns selected: it encodes categorical features using a one-hot scheme (also known as "one-of-K" scheme), and outputs a binary column for each distinct category in the input column.



### Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes <sup>1</sup>
Data processing tool	Spark

The One-Hot Encoding operator is useful for turning categorical predictors into numeric (binary) predictors for algorithms that do not support categorical variables natively.

**i Note:** Most of the ML algorithms available in TIBCO Data Science - Team Studio already include one-hot encoding as a preprocessing step directly.

### Input

A single HDFS tabular data set.

---

<sup>1</sup>The full output schema is not available until you step run the operator. After you run this operator, the output schema automatically updates, and subsequent operators either validate or turn red, depending on the structure of the output data.

## Bad or Missing Values

**Dirty data:** When parsing delimited data, the One-Hot Encoding operator removes dirty data (such as strings in numeric columns, doubles in integer columns, or rows with the incorrect number of values) as it parses. These rows are silently removed because Spark is incapable of handling them.

**Null values:** Before performing one-hot encoding, the operator filters any rows that contain null values in the specified **Columns to Encode**. The operator then processes these rows with null values according to the value of the **Write Rows Removed Due to Null Data To File** parameter. The number of rows removed due to null data is reported in the **Summary** tab of the visual output.

## Restrictions

For the maximum number of categories for each column selected on **Columns to Encode**, the default value is 30; this value can be modified in the **Advanced Spark Settings** menu (in the parameter **Max Column Distinct Categories**).

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns to Encode</b>  *required	Select the categorical column(s) on which to perform one-hot encoding.
<b>Keep Encoded Columns</b>	Define whether the input columns to encode should be kept in the output - <b>yes</b> or <b>no</b> (the default).
<b>Drop Last Category</b>	Select <b>Yes</b> (the default) to indicate that the last category in the column to encode should be dropped. Otherwise, select <b>No</b> .  For example, if a column to encode "categ" contains three categories ("a", "b", "c") and <b>Yes</b> is selected for this parameter, the output data set

Parameter	Description
	contains only two encoded binary columns: "categ_a" and "categ_b" ("categ_c" is dropped).
<b>Output Column Prefix</b>	(Optional) Specify a string to prepend to all the output encoded column names. This option can be useful if you want to select all of the encoded columns in a subsequent operator, because they all start with the same prefix, which simplifies filtering on the first letters and then selecting them.
<b>Write Rows Removed Due to Null Data to File</b>	<p>Rows with null values (only in the <b>Columns to Encode</b>) are removed from the analysis. Use this parameter to specify that the data with null values is written to a file. The file is written to the same directory as the rest of the output. The filename is appended with the suffix <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> (the default) - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data, but do not count and display in the result UI.</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

- **Output** preview:

msa	pmisa	State_AK	State_AL	State_AR	State_AZ	State_CA	State_CO	State_CT	State_DC	State_DE	State_FL	State_GA
5240	N/A	0	1	0	0	0	0	0	0	0	0	0
5180	N/A	0	1	0	0	0	0	0	0	0	0	0
N/A	N/A	0	1	0	0	0	0	0	0	0	0	0
N/A	N/A	0	1	0	0	0	0	0	0	0	0	0
1000	N/A	0	1	0	0	0	0	0	0	0	0	0
N/A	N/A	0	1	0	0	0	0	0	0	0	0	0
N/A	N/A	0	1	0	0	0	0	0	0	0	0	0
450	N/A	0	1	0	0	0	0	0	0	0	0	0
N/A	N/A	0	1	0	0	0	0	0	0	0	0	0
N/A	N/A	0	1	0	0	0	0	0	0	0	0	0
N/A	N/A	0	1	0	0	0	0	0	0	0	0	0
N/A	N/A	0	1	0	0	0	0	0	0	0	0	0

- **Summary:** parameters selected, null data removed from input and output location:

<b>Output</b> <b>Summary</b>	<b>Parameters selected:</b> Selected Columns for One-Hot Encoding (first 10): State Keep the Selected Columns: No Drop the Last Encoded Column: No Output Column Prefix: Number of Columns Created: 51  Input data size: 3141 rows Input size after removing rows due to null values in 'Columns to Encode': 3141 rows No data removed due to null values in 'Columns to Encode'  The output of the operator is stored at /tmp/tsds_out/bbenchoff/6.4_Acceptance_Criteria_Testing_3491/One_Hot_Encoding_OP22
---------------------------------	--

## Data Output

The data set that contains the encoded columns.

## Additional Notes

### "Semi-terminal" operator

A partial schema can be transmitted to subsequent operators at design time, but you must run the operator for subsequent operators to see the final output schema.

**i Note:** The final output schema of the Correlation Filter operator is cleared if one of the following occurs.

- The user changes the configuration properties of the One-Hot Encoding operator.
- The user changes the input connected to the One-Hot Encoding operator.
- The user clears the step run results of the One-Hot Encoding operator.

In this case, the output schema transmitted to subsequent operators again becomes the partial schema defined at design time (hence, subsequent operators can turn invalid), and you must run the One-Hot Encoding operator again to transmit the new output schema.

## Pivot (DB)

Lets you transform the categorical data contained in a column of a table into columns of a new table, by means of subtotals (or other calculations) that might be defined by another column in the same list. The other calculations might be averages and counts.



## Information at a Glance

**i Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

**i Note:** The Pivot (DB) operator is for database data only. For Hadoop data, use the [Pivot \(HD\)](#) operator.

## Algorithm

For typical data entry and storage, data usually appears in flat tables, meaning that it consists of only columns and rows. While such data can contain a lot of information, it can be difficult to get summarized information. A pivot table can help quickly summarize the flat data, giving it depth, and highlighting the desired information.

The usage of a pivot table is extremely broad and depends on the situation. The first question to ask is "What am I looking for?" A pivot table usually consists of row, column, and data (or fact) fields. These fields allow several kinds of aggregations including: sum, average, count, max, min, etc.

The Pivot Column is usually a categorical column, and in the output table it is transformed into multiple columns, one for each category.

- The results are also grouped by a selected column.
- The values in the new columns are aggregates of a third column (or of the presence of the category if no aggregate column is chosen).
- The columns are listed (in the table, file, or array) in descending order of the value of the category they represent.



**Note:** This operator can only connect to subsequent operators in certain situations - see the Output section below for details.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Pivot Column</b>	The column for pivot transformation. On database data sources, you can only choose categorical columns to pivot.
<b>Group By</b>	The column to group by.
<b>Aggregate Column</b>	The pivot column's value column.
<b>Aggregation</b>	The aggregate function. Options: <ul style="list-style-type: none"> <li>• Sum</li> <li>• Average</li> </ul>



Parameter	Description
	<ul style="list-style-type: none"> <li>Count</li> <li>Max</li> <li>Min</li> </ul>
<b>Use Array</b>	<p>The format of the output.</p> <ul style="list-style-type: none"> <li>Check the <b>Use Array</b> checkbox to use an array. This option is non-terminal.</li> <li>Clear the <b>Use Array</b> checkbox to not use an array. This option is terminal.</li> </ul> <p>Default value: false</p>
<b>Output Type</b>	<ul style="list-style-type: none"> <li><b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li><b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	<p>Advanced database settings for the operator output. Available only for <b>TABLE</b> output.</p> <p>See <a href="#">Storage Parameters dialog</a> for more information.</p>
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li><b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li><b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

If **Use Array** is true, pivoted values are stored in an array in one column.

monthly_income_bin	credit_lines_bin
2	{0.027453455348690439,0.044572468162522742,0.075007076139258416}
4	{0.026041666666666668,0.037250151423379771,0.07248611967921037}
5	{0.023662274805055124,0.0342424242424241,0.063737001006373695}
3	{0.029895366218236172,0.041962174940898343,0.063894833384286157}
1	{0.029224051058112193,0.042861365651788351,0.070879120879120877}

The values in the array are arranged in descending order of the value of the category they represent.

If **Use Array** is false or if the operator is operating on a Hadoop data set, pivoted values are each placed in their own column.

monthly_income_bin	credit_lines_bin_3	credit_lines_bin_2	credit_lines_bin_1
2	0.0275	0.0446	0.075
4	0.026	0.0373	0.0725
5	0.0237	0.0342	0.0637
3	0.0299	0.042	0.0639
1	0.0292	0.0429	0.0709

### Data Output

If **Use Array** is true, a data set of the newly created table or view is output to the succeeding operator. Otherwise, no output is sent.

## Example

The raw grocery data shown below can be quickly converted using the Pivot operator into a summary table of the count of beers sold when wine was or was not sold.



## Algorithm

For typical data entry and storage, data usually appears in flat tables, meaning that it consists of only columns and rows. While such data can contain a lot of information, it can be difficult to get summarized information. A pivot table can help quickly summarize the flat data, giving it depth, and highlighting the desired information.

The usage of a pivot table is extremely broad and depends on the situation. The first question to ask is "What am I looking for?" A pivot table usually consists of row, column, and data (or fact) fields. These fields allow several kinds of aggregations including: sum, average, count, max, min, etc.

The Pivot Column is usually a categorical column, and in the output table it is transformed into multiple columns, one for each category.

- The results are also grouped by a selected column.
- The values in the new columns are aggregates of a third column (or of the presence of the category if no aggregate column is chosen).
- The columns are listed (in the table, file, or array) in descending order of the value of the category they represent.

**i Note:** This operator can only connect to subsequent operators in certain situations - see the Output section below for details.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Pivot Column</b>	The column for pivot transformation. You can choose any data type

Parameter	Description
	column. Limit: 1,500 distinct values.
<b>Group By</b>	The column to group by.
<b>Aggregate Column</b>	The pivot column's value column.
<b>Aggregation</b>	The aggregate function. Options: <ul style="list-style-type: none"> <li>• <b>sum</b></li> <li>• <b>average</b></li> <li>• <b>count</b></li> <li>• <b>max</b></li> <li>• <b>min</b></li> </ul>
<b>Store Results?</b>	Specifies whether to store the results. <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	The HDFS directory where the results of the operator are stored. This is the main directory, the subdirectory of which is specified in <b>Results Name</b> . Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.
<b>Results Name</b>	The name of the file in which to store the results.
<b>Overwrite</b>	Specifies whether to delete existing data at that path and file name. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Storage Format</b>	Select the format in which to store the results. The storage format is determined by your type of operator.  Typical formats are <b>Avro</b> , <b>CSV</b> , <b>TSV</b> , or <b>Parquet</b> .

Parameter	Description
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

If **Use Array** is true, pivoted values are stored in an array in one column.

monthly_income_bin	credit_lines_bin
2	{0.027453455348690439,0.044572468162522742,0.075007076139258416}
4	{0.026041666666666668,0.037250151423379771,0.07248611967921037}
5	{0.023662274805055124,0.0342424242424241,0.063737001006373695}
3	{0.029895366218236172,0.041962174940898343,0.063894833384286157}
1	{0.029224051058112193,0.042861365651788351,0.070879120879120877}

The values in the array are arranged in descending order of the value of the category they represent.

If **Use Array** is false or if the operator is operating on a Hadoop dataset, pivoted values are each placed in their own column.

monthly_income_bin	credit_lines_bin_3	credit_lines_bin_2	credit_lines_bin_1
2	0.0275	0.0446	0.075
4	0.026	0.0373	0.0725
5	0.0237	0.0342	0.0637
3	0.0299	0.042	0.0639
1	0.0292	0.0429	0.0709

### Data Output

If **Use Array** is true, no output is sent to the succeeding operator.

## Reorder Columns (DB)

Reorders one or more columns from an input table, and optionally renames them.



### Information at a Glance

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	DB

**i Note:** The Reorder Columns (DB) operator is for database data only. For Hadoop data, use the [Reorder Columns \(HD\)](#) operator.

## Input

A database table.

## Configuration

<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Ordered Columns</b>	Click <b>Define</b> to specify the columns (in order) to become the first columns in the output, and optionally specify a new name for each. See <a href="#">Ordered Columns dialog</a> for more information.
<b>Columns to Keep</b>	Specify any other columns to keep in the output.
<b>Output Type</b>	<ul style="list-style-type: none"> <li>• <b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li>• <b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Outputs

### Visual Output



Output (Preview of the output data set):

<b>Output</b> <b>Summary</b>	id	times_90_late	c_lines	monthly_income
	2	0	4	1,956.59
	6	0	6	3,983.46
	52	0	7	3,595.37
	76	2	3	1,729.96
	84	0	7	2,507.17

Summary:

<b>Output</b> <b>Summary</b>	<b>Parameters</b> Columns Reordered (first 10): id, times90dayslate, credit_lines
	<b>Output</b> The output of the operator is stored at /tmp/tsds_out/emilie/Test_Reorder_3163/Reorder_Columns_OP0

## Data Output

A database table with reordered and (optionally) renamed columns.

## Reorder Columns (HD)

Reorders one or more columns from an input table, and optionally renames them.



## Information at a Glance

Parameter	Description
Category	Transform

Parameter	Description
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark



**Note:** The Reorder Columns (HD) operator is for Hadoop data only. For database data, use the [Reorder Columns \(DB\)](#) operator.

## Input

A tabular data set.

## Configuration

<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Ordered Columns</b>	Click <b>Define</b> to specify the columns (in order) to become the first columns in the output, and optionally specify a new name for each. See <a href="#">Ordered Columns dialog</a> for more information.
<b>Columns to Keep</b>	Specify any other columns to keep in the output.
<b>Storage Format</b>	Select the format in which to store the results. The storage format is determined by your type of operator.  Typical formats are <b>Avro</b> , <b>CSV</b> , <b>TSV</b> , or <b>Parquet</b> .
<b>Compression</b>	Select the type of compression for the output.  Available Parquet compression options.  • <b>GZIP</b>

- **Deflate**
- **Snappy**
- no compression

Available Avro compression options.

- **Deflate**
- **Snappy**
- no compression

<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	Specifies whether to delete existing data at that path. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Outputs

### Visual Output

**Output (Preview of the output data set):**

<b>Output</b> <a href="#">Summary</a>	<b>id</b>	<b>times_90_late</b>	<b>c_lines</b>	<b>monthly_income</b>
	2	0	4	1,956.59
	6	0	6	3,983.46
	52	0	7	3,595.37
	76	2	3	1,729.96
	84	0	7	2,507.17

**Summary:**

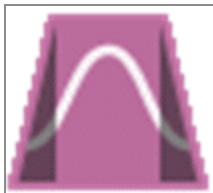
<b>Output Summary</b>	<b>Parameters</b> Columns Reordered (first 10): id, times90dayslate, credit_lines  <b>Output</b> The output of the operator is stored at /tmp/tsds_out/emilie/Test_Reorder_3163/Reorder_Columns_OP0
-----------------------	--

**Data Output**

A tabular data set with reordered and (optionally) renamed columns.

## Replace Outliers (DB)

Reduces the range of values for numeric columns.

**Information at a Glance**

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	DB

For more information about how the Replace Outliers operator works, see [Outliers in Numerical Data](#).

**Note:** The Replace Outliers (DB) operator is for database data only. For Hadoop data, use the [Replace Outliers \(HD\)](#) operator.

## Input

This operator works for tabular data sets. The transformation function can be applied only to numeric columns, and the type of the numeric input columns is preserved in the output.

### Bad or Missing Values

Any row that contains dirty data, such as a string in a numeric column, is removed as the data is read in. After the data is read in, the operator filters out all rows that contain null values in the selected numeric columns. Rows that have null values in any of the columns not selected are not removed. The rows removed are reported in the **Summary** tab. If the value of **Write Null Data to File Parameter** is set to yes, then the rows removed because they have null data are written to an external file (the location of which is reported in the **Summary** tab).

## Restrictions

Any data set with numeric columns can be used. This operator slows down as the number of columns selected and the cardinality of the columns increases.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	The numeric columns to transform.
<b>Lower Boundary (%)</b>	<p>A double that represents the percentage of values in the left tail of the distribution (on the low end of the range in each column) to replace.</p> <p>The lower threshold <math>x</math> is calculated as <math>\frac{\text{rows} \times \text{lower\_boundary}}{100}</math>.</p>

Parameter	Description
<b>Upper Boundary (%)</b>	<p>A double that represents the percentage of values in the right tail of the original distribution for each column (the high end of the range in each column) to replace.</p> <p>The upper threshold <math>y</math> is calculated as <math>\frac{rows * upper\_boundary}{100}</math>.</p>
<b>Output Type</b>	<ul style="list-style-type: none"> <li>• <b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li>• <b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

The operator has two tabs of output. The first is the output data, which is passed on to the next operator. The second is a summary that explains which parameters were selected, how much null data was removed, and where the results were written.

- **Output:** A table with the outlier values replaced, as detailed above.

id	times90dayslate	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslate	srsdlqncy
2	0	0.26	0.324	4	1,956.592205	0	0
6	0	0.25	0.927	6	3,983.456356	0	0
52	0	0.46	0.372	7	3,595.370678	0	0
76	2	0.88	0.154	3	1,729.964479	2	0
84	0	0.14	0.69	7	2,507.165171	0	0
86	0	0.11	0.131	5	2,126.599617	0	0

- **Summary:** A description of the input data and the rows removed due to null data. It also shows where the results are stored.

```

Input data size: 50000 rows
Input size after removing rows due to null values: 50000 rows
No data removed due to null values

The output of the operator is stored at
/tmp/Replace_Outliers_1464042000332

```

## Data Output

The operator outputs the same tabular data set as the input data, but with some of the values in the selected numeric columns replaced. See [Outliers in Numerical Data](#) for more information.

## Replace Outliers (HD)

Reduces the range of values for numeric columns.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD

Parameter	Description
Send output to other operators	Yes
Data processing tool	Spark

For more information about how the Replace Outliers operator works, see [Outliers in Numerical Data](#).



**Note:** The Replace Outliers (HD) operator is for Hadoop data only. For database data, use the [Replace Outliers \(DB\)](#) operator.

## Input

This operator works for tabular data sets on HDFS. The transformation function can be applied only to numeric columns, and the type of the numeric input columns is preserved in the output.

### Bad or Missing Values

Any row that contains dirty data, such as a string in a numeric column, is removed as the data is read in. After the data is read in, the operator filters out all rows that contain null values in the selected numeric columns. Rows that have null values in any of the columns not selected are not removed. The rows removed are reported in the **Summary** tab. If the value of **Write Null Data to File Parameter** is set to yes, then the rows removed because they have null data are written to an external file (the location of which is reported in the **Summary** tab).

## Restrictions

Any data set with numeric columns can be used. This operator slows down as the number of columns selected and the cardinality of the columns increases.



## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	Numeric columns to transform.
<b>Lower Boundary (%)</b>	<p>A double that represents the percentage of values in the left tail of the distribution (on the low end of the range in each column) to replace.</p> <p>The lower threshold <math>x</math> is calculated as <math display="block">\frac{\text{rows} * \text{lower\_boundary}}{100}</math></p>
<b>Upper Boundary (%)</b>	<p>A double that represents the percentage of values in the right tail of the original distribution for each column (the high end of the range in each column) to replace.</p> <p>The upper threshold <math>y</math> is calculated as <math display="block">\frac{\text{rows} * \text{upper\_boundary}}{100}</math></p>
<b>Write Null Data To File</b> *required	<p>Rows with null values are removed from the analysis. This parameter allows you to specify that the data with null values be written to a file.</p> <p>The file is written to the same directory as the rest of the output. The filename is suffixed with <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null-value data, but do not count and display in the result UI.</li> <li>• <b>Do Not Write Null Rows to File</b> - remove null-value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Write Up to 1000 Null Rows to File</b> - remove null-value data and write the first 1,000 rows of that data to the external file.</li> <li>• <b>Write All Null Rows to File</b> - remove null-value data and write all removed rows to an external file.</li> </ul>

Parameter	Description
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

The operator has two tabs of output. The first is the output data, which is passed on to the next operator. The second is a summary that explains which parameters were selected, how much null data was removed, and where the results were written.

- **Output:** A table with the outlier values replaced, as detailed above.

id	times90dayslate	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslate	srsdlqncy
2	0	0.26	0.324	4	1,956.592205	0	0
6	0	0.25	0.927	6	3,983.456356	0	0
52	0	0.46	0.372	7	3,595.370678	0	0
76	2	0.88	0.154	3	1,729.964479	2	0
84	0	0.14	0.69	7	2,507.165171	0	0
86	0	0.11	0.131	5	2,126.599617	0	0

- **Summary:** A description of the input data and the rows removed due to null data. It also shows where the results are stored.

```

Input data size:                    50000 rows
Input size after removing rows due to null values:  50000 rows
No data removed due to null values

The output of the operator is stored at
/tmp/Replace_Outliers_1464042000332

```

## Data Output

The operator outputs the same tabular data set as the input data, but with some of the values in the selected numeric columns replaced. See [Outliers in Numerical Data](#) for more information. This output should work as input for any Hadoop operator that expects tabular data.

## Row Filter (DB)

Sets the criteria for filtering data set rows. Only the rows that meet the criteria remain in the output data set.



## Information at a Glance

**i Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

**i Note:** The Row Filter (DB) operator is for database data only. For Hadoop data, use the [Row Filter \(HD\)](#) operator.

You can specify row filters in the following modes.

- Simple mode: Use a simple template to define the filter, choosing a column, an inequality (for example, ">" or "between"), and a value (for example, a literal value or a column expression).
- Script mode: Enter almost any set of filters by using a SQL script.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
Notes	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the

Parameter	Description
	operator.
<b>Filter</b>	The filters for the operator. See <a href="#">Define Filter dialog</a> for more information.
<b>Output Type</b>	<ul style="list-style-type: none"> <li>• <b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li>• <b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	<p>Advanced database settings for the operator output. Available only for <b>TABLE</b> output.</p> <p>See <a href="#">Storage Parameters dialog</a> for more information.</p>
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

The data rows of the output table or view displayed (up to 200 rows of the data).

### Data Output

Either a newly created table or a view.

## Row Filter (HD)

Sets the criteria for filtering data set rows. Only the rows that meet the criteria remain in the output data set.



### Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Pig



**Note:** The Row Filter (HD) operator is for Hadoop data only. For database data, use the [Row Filter \(DB\)](#) operator.

You can specify row filters in the following modes.

- Simple mode: Use a simple template to define the filter, choosing a column, an inequality (for example, ">" or "between"), and a value (for example, a literal value or a column expression).
- Script mode: Enter almost any set of filters by using a Pig script.

### Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Filter</b>	The filters for the operator. See <a href="#">Define Filter dialog</a> for more information.
<b>Use Row Limit?</b>	Specify whether to use a row limit to limit filtering to specified rows. Default: <b>false</b> .
<b>Row Limit Amount</b>	If <b>Use Row Limit?</b> is set to <b>true</b> , set this field to the number of rows to which to limit the filtering.
<b>Store Results?</b>	Specifies whether to store the results. <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	The HDFS directory where the results of the operator are stored. This is the main directory, the sub-directory of which is specified in <b>Results Name</b> . Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.
<b>Results Name</b>	The name of the file in which to store the results.
<b>Overwrite</b>	Specifies whether to delete existing data at that path and file name. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Storage Format</b>	Select the format in which to store the results. The storage format is determined by your type of operator.  Typical formats are <b>Avro</b> , <b>CSV</b> , <b>TSV</b> , or <b>Parquet</b> .
<b>Compression</b>	Select the type of compression for the output.

Parameter	Description
	<p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

The data rows of the output table or view displayed (up to 200 rows of the data).

### Data Output

Either a newly created table or a new file.

## Sessionization

Enables the application of sessionization on time-series data to create a **session\_id** column that, for each row (and user ID), gives the session the action belongs to.





## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

A session can be described as an ordered list of a user's actions in completing a task. For information about how the boundaries of a session are defined, see the **Session Boundaries** parameter in the table below.

**Note:** The first session for each user ID starts at 0.

Sessionization is most commonly used in web analytics for log/clickstream analysis, but is also popular in other areas such as predictive maintenance and IoT.

## Input

This operator requires a single tabular input that contains at least a datetime column.

### Bad or Missing Values

- **Dirty data:** When parsing delimited data, the Sessionization operator removes dirty data (such as strings in numeric columns, doubles in integer columns, or rows with the incorrect number of values) as it parses. These rows are silently removed because Spark is incapable of handling them.
- **Null values:** Before applying sessionization, the operator filters any rows that

contain null values either in the **Timestamp** column or the **Status** column. The operator then processes these rows with null values according to the value of the **Write Rows Removed Due to Null Data To File** parameter. The number of rows removed due to null data is reported in the **Summary** tab of the visual output.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Session Boundaries</b>	Select the method to define session boundaries: <ul style="list-style-type: none"> <li>• <b>Time Interval Threshold</b> - define a new user session if this user is inactive for a certain amount of time.</li> <li>• <b>Change of Status</b> - define a new user session when there is a change of assignment.</li> </ul>
<b>Timestamp Column</b>	Select the datetime column that contains the timestamp for each action in the data set.
<b>Time Interval Threshold (seconds)</b>	Required only if <b>Session Boundaries</b> is set to <b>Time Interval Threshold</b> .  Enter the threshold of inactivity, in seconds, that is used to define a new session.
<b>Status Column</b>	Required only if <b>Session Boundaries</b> is set to <b>Change of Status</b> .  Specify the column to use to detect the change of assignment to define a new session.
<b>User ID Column(s)</b>	Select the user ID column(s) to use to partition the input data set and create session IDs for each distinct user.
<b>Columns to Keep</b>	Select the input columns to keep in the output.
<b>Write Rows</b>	Rows with null values are removed from the analysis. Use this

Parameter	Description
<b>Removed Due to Null Data To File</b>	<p>parameter to specify that the data with null values is written to a file. The file is written to the same directory as the rest of the output. The filename is appended with the suffix <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data, but do not count and display in the result UI.</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	Specifies whether to delete existing data at that path.

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Outputs

### Visual Output

Output data set with **new\_session** and **session\_id** columns.

Output Summary	machine_uid	timestamp	meas_type	product	meas1	meas2	meas3	new_session	session_id
	uid_1	2015-01-01 01:07:00	PRESSURE	productA	19.98	19.7	23.2	0	0
	uid_1	2015-01-01 01:09:54	PRESSURE	productA	20.21	19.7	23.2	1	1
	uid_1	2015-01-01 01:10:50	HUMIDITY	productA	70.21	70.22	80.12	1	2
	uid_1	2015-01-01 01:10:56	PRESSURE	productA	20.22	19.8	2.2	0	2
	uid_1	2015-01-01 02:07:00	PRESSURE	productB	19.93	19.72	20.2	1	3
	uid_1	2015-01-01 02:09:21	PRESSURE	productB	20.25	20.7	21.2	1	4
	uid_1	2015-01-01 02:10:20	HUMIDITY	productB	70.23	70.26	79.09	1	5
	uid_1	2015-01-01 02:11:45	PRESSURE	productB	20.22	19.8	2.2	1	6
	uid_2	2015-01-01 01:09:34	HUMIDITY	productA	70.21	70.22	80.12	0	0
	uid_2	2015-01-01 01:09:51	PRESSURE	productA	20.25	19.6	23.2	1	1

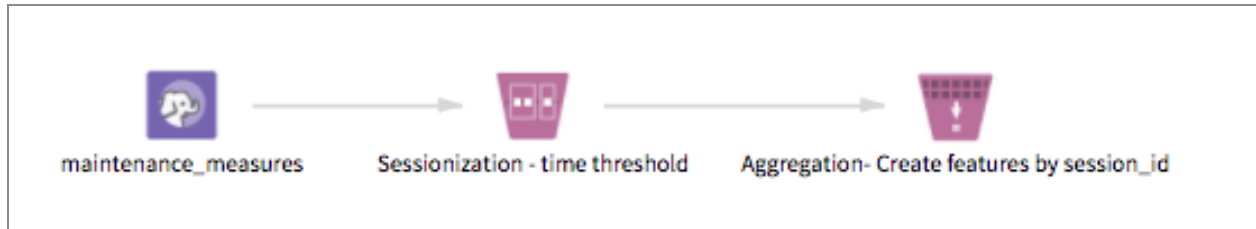
Summary of the user parameters and output location.

Output Summary	<p><b>Parameters selected</b></p> <p>Session Boundaries      Time Interval Threshold</p> <p>TimeStamp Column:      timestamp</p> <p>Time Interval Threshold (s): 10.0</p> <p>User ID Column(s):      machine_uid</p> <p><b>Output</b></p> <p>Input data size: 16 rows</p> <p>Input size after removing rows due to null values in 'TimeStamp Column' or 'Status Column': 16 rows</p> <p>No data removed due to null values in 'TimeStamp Column' or 'Status Column'</p> <p>The output of the operator is stored at</p> <p>/tmp/alpine_out/edelongeau/Sessionization_Example/Sessionization__time_threshold_OP0</p>
----------------	--

## Data Output

Connect this operator to succeeding operators.

## Example



## Set Operations (DB)

Combines results from merging two or more queries into a single result set.



## Information at a Glance

**i Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

**i Note:** The Set Operations (DB) operator is for database data only. For Hadoop data, use the [Set Operations \(HD\)](#) operator.

## Input

Two or more databases.

## Restrictions

- The number and the order of the columns must be the same in all queries.
- The data types must be compatible.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Sets</b>	Click <b>Define Sets</b> to display the Define Sets dialog. For more information, see the <a href="#">Define Sets dialog</a> .
<b>Output Type</b>	<ul style="list-style-type: none"> <li>• <b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li>• <b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	Advanced database settings for the operator output. Available only for <b>TABLE</b> output.

Parameter	Description
	See <a href="#">Storage Parameters dialog</a> for more information.
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

The data rows of the output table or view displayed (up to 200 rows).

### Data Output

A data set of the output table or view of the joined data sets. This operator always creates a CSV output.

## Set Operations (HD)

Combines results from merging two or more queries into a single result set.



### Information at a Glance

Parameter	Description
Category	Transform
Data source type	DB

Parameter	Description
Send output to other operators	Yes
Data processing tool	MapReduce / Spark



**Note:** The Set Operations (HD) operator is for Hadoop data only. For database data, use the [Set Operations \(DB\)](#) operator.

## Input

Two or more databases.

## Restrictions

- The number and the order of the columns must be the same in all queries.
- The data types must be compatible.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Sets</b>	Click <b>Define Sets</b> to display the Define Sets dialog. For more information, see the <a href="#">Define Sets dialog</a> .
<b>Store Results?</b>	Specifies whether to store the results. <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results</b>	The HDFS directory where the results of the operator are stored. This is the



Parameter	Description
<b>Location</b>	main directory, the sub-directory of which is specified in <b>Results Name</b> . Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.
<b>Results Name</b>	The name of the file in which to store the results.
<b>Overwrite</b>	Specifies whether to delete existing data at that path and file name. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Storage Format</b>	Select the format in which to store the results. The storage format is determined by your type of operator.  Typical formats are <b>Avro</b> , <b>CSV</b> , <b>TSV</b> , or <b>Parquet</b> .
<b>Compression</b>	Select the type of compression for the output.  Available Parquet compression options. <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> Available Avro compression options. <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

The data rows of the output table/view displayed (up to 200 rows).

### Data Output

A data set of the joined data sets. This operator always creates a CSV output.

## Sort By Multiple Columns

Allows you to choose up to three columns to sort by and returns a data set sorted by the selected column(s), adding a column called `row_index` that enables you to filter the output based on the sorting results.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

## Input

A tabular data set from HDFS.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Primary Sort Column</b>	First column to sort by. While <b>Secondary Sort Column</b> and <b>Tertiary Sort Column</b> can be left blank, this column is required.
<b>Primary Column Sort Order</b>	Order by which to sort the first column: <b>Ascending</b> (the default) or <b>Descending</b> .
<b>Secondary Sort Column</b>	Second column to sort by. To sort by one column only, leave this column and the <b>Tertiary Sort Column</b> blank.
<b>Secondary Column Sort Order</b>	Order by which to sort the second column: <b>Ascending</b> (the default) or <b>Descending</b> .
<b>Tertiary Sort Column</b>	Third column to sort by. To sort two columns only, leave this one blank.
<b>Tertiary Column Sort Order</b>	Order by which to sort the third column: <b>Ascending</b> (the default) or <b>Descending</b> .
<b>Create 'row_index' Column</b>	Specify whether to add the row_index column, which adds an extra column to the data set that shows the sort index.  Default value: No.
<b>Write Rows Removed Due to Null Data to File</b>	<p>Rows with null values (only in the columns selected to sort by) are removed from the analysis. This parameter allows you to specify that the data with null values be written to a file.</p> <p>The file is written to the same directory as the rest of the output. The file name is suffixed with _baddata.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> (the default) - remove null value data and display in the result UI, but do not write to an external</li> </ul>

Parameter	Description
	<p>file.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI.</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> </ul>

Parameter	Description
<b>Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

The following example is sorted by age, then income.

age	income	id	location
10	-98,849	7,419	Belmont
10	-91,121	7,956	N. Dartmouth
10	-89,737	586	N. Dartmouth
10	-86,803	5,035	San Francisco
10	-84,426	7,698	Belmont
10	-82,900	5,699	N. Dartmouth
10	-82,674	6,341	Clearwater

### Data Output

A data set that contains the sorted columns and the extra row\_index column if selected.

## Time Series SAX Encoder

Produces a new data set with one or more columns that contains the time series ID and discretized string representation of the original time series.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark SQL

The operator takes each time series in a row of the input table and creates a user-requested compressed representation of each input time series. Null values are dropped, and the time series is z-normalized if the user requests it.

The time series is then binned into a user-requested number of bins (specified in the **SAX String Length** parameter). If the length of the time series is not exactly divisible by the requested number of bins, the operator uses a partial contributions approach to determine the number of data points to include in each bin.

For example, if the time series has 10 data points and the user requests a bin size of three, the bin divisions are as follows.

- The first bin gets the first three points and 1/3rd of the fourth point.
- The second bin gets two-thirds of the fourth point, plus the fifth and sixth points, plus two-thirds of the seventh point.
- The third bin gets one-third of the seventh point, plus the last three points.

Once the time series is binned, the values within each bin are aggregated according to user selection (specified in the **Aggregation Method** parameter). If the user requests aggregate output, the values are returned; otherwise, the aggregated value is compared to the standard normal distribution, and the corresponding cut of the distribution is returned as the output.

**i Note:** The standard normal distribution is binned according to user request (specified in the **SAX Alphabet Size** parameter), and each bin is assigned an alphabet from the lower tail to the upper tail.

## Input

A single tabular data set.

### Bad or Missing Values

Null values in a series are dropped, and a time series with all null values returns a null string or is dropped if alphabet or aggregate output is selected.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Time Series Columns Range</b>	Specify the column number range that contains the time series, one series per row.
<b>Aggregation Method</b>	Specify the aggregation method to use in SAX encoding - <b>Average</b> (the default), <b>Maximum</b> , <b>Median</b> , or <b>Minimum</b> .
<b>SAX String Length</b>	Specify the number of bins into which to discretize the time series.
<b>SAX Alphabet Size</b>	Specify the number of intervals into which to divide the z-normal distribution.
<b>Time Series ID Column</b>	An optional column name that contains the ID of the time series, for output clarity. If a name is not specified, TIBCO Data Science - Team Studio generates an ID column with row IDs in the output.
<b>Columns to Keep</b>	Click the <b>Select Columns</b> button to select columns from the input data set to append to the output.
<b>Output Format</b>	Defines the output format. <ul style="list-style-type: none"> <li>• <b>SAX Aggregate</b> - the time series is binned, aggregated within bin and each aggregated value is returned</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>SAX Alphabets</b> - same as string output, except the individual alphabets are returned without concatenating</li> <li>• <b>SAX String</b> (the default) - the time series is normalized, binned, aggregated within bin and the bin values converted to alphabets and concatenated to form the string</li> </ul>
<b>Z Normalize Input</b>	Specify whether the input time series should be standardized. <b>Yes</b> (the default) or <b>No</b> .
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>



Parameter	Description
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

A tabular preview of the output data set, which includes **Output** and **Summary** tabs.

### Output

A single tabular data set that displays the SAX-encoded strings.

Results - Time Series SAX Encoder		
Output Summary	Batch	
	SAX_String	
	1574_24.11	bbhjbbdghhhhggfb
	1582_21.11	aaiaaggghhhhggfb
	1666_06.11	aaiaabggghhhhggga
	1674_03.11	aahjaaghhhhhgggb
	1715_03.11	bbdjcbfhhhhhgggb
	1719_18.11	aagjabggghhhhgggb
	1732_06.11	aaigabghhhhggga
	1859_04.11	aaifbbggghhhhggga
	1861_14.11	aaigabghhhhggga
	1551_08.11	bbgjdkbeghhhhggeb
	1569_07.11	bbhjbbfggghhhgggb
	1678_06.11	bbdjdbeghhhhhgggb
	1554_02.11	bbgjbbeghhhhhggfb
	1571_05.11	aaiaabggghhhhggga
	1638_01.11	adjdaeggggggffa
	1672_07.11	aaiaabggghhhhggfa

Summary

The default summary, which includes selected parameters, input data size, and output location.

Results - Time Series SAX Encoder

Output Summary

Parameters selected

Time Series Columns Range: 2-\*

Aggregation Method: Average

SAX String Length: 16

SAX Alphabet Size: 10

Time Series ID Column: Batch

Output

Number of Encoded Time Series: 1194

The output of the operator is stored at

/tmp/tsds\_out/siramali\_tibco.com\_5/sax/Time\_Series\_SAX\_Encoder\_OP28

Transpose

Allows you to rearrange data so that rows and columns are switched.



Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD

Parameter	Description
Send output to other operators	Yes <sup>1</sup>
Data processing tool	Spark

You can choose which input column should be used to define the new header. If the input has X columns and Y rows, the output has Y rows and X columns.

In the following example, the **Name** column is selected to be the output header.

Age	Name	Grade
12	Jenny	A
14	Mary	A
13	Emily	B

After Transpose is run with **Name** as the header column, the data set looks like the following example.

Name	Jenny	Mary	Emily
Age	12	14	13
Grade	A	A	B

## Input

A data set from HDFS to this operator. At least one categorical column is necessary to define the new header.

---

<sup>1</sup>The full output schema is not available until you step run the operator. After you run this operator, the output schema automatically updates, and subsequent operators either validate or turn red, depending on the structure of the output data.

## Bad or Missing Values

Missing values are kept only if they are not in the column selected to define the new header. In this case, the job fails at runtime.

## Restrictions

This operator cannot transpose an input larger than 5,000 rows. If the input has a single column and you select this column to be the new header, an error occurs while the operator is being configured.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Column for New Header</b>	<p>You can, as an option, select a categorical (chararray) column whose name and values define the new header. If no column is selected, the header in the output is default (Column1, Column2...ColumnX)</p> <div> <p><b>Note:</b> If the selected column contains null or duplicate values, the job fails at runtime with a meaningful error message.</p> <p>If some values contain non-alphanumeric characters, they are replaced by an underscore in the new header.</p> <p>If some values start with a non-letter character, the letter "a" is prepended to match the column name regex <code>^[A-Za-z]+ \\ w*\$</code>.</p> </div>
<b>New Name for First Column</b>	<p>Optional new name for the first column in the output, matching the regular expression <code>^[A-Za-z]+ \\ w*\$</code>.</p> <p>If you do not want to specify a name, keep the default empty box.</p>

Parameter	Description
	<p><b>Note:</b> If you specify a value here, it overrides the first column name of the output: either the column name of the column selected for new header (previous parameter), or the default value Column1 from the default header if no input column was selected to define the output header.</p>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> </ul>

Parameter	Description
<b>Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

Operation created output with 13 columns and 3 rows.

Metric	jan	feb	march	april	may	june	july	august	september	october	november	december
val1	0.93766347	0.94668297	0.94202005	0.86770206	0.63411418	0.26033171	0.96875414	0.38867835	0.94475105	0.91123432	0.96219359	0.95877206
val2	0.34194661	0.9228087	0.22991311	0.70792827	N/A	0.00662886	0.71413567	0.13727108	0.6894632	0.99027129	0.62652286	0.39567037
val3	0.56299834	0.76873349	0.93251239	0.53209337	0.96758924	0.10189652	0.14254723	0.57676273	N/A	0.14433212	0.68374501	0.14312665

### Data Output

This is a semi-terminal operator that can be connected to any subsequent operator at design time, but does not transmit the full output schema until the user runs the operator. The partial output schema at design time is only be the first column of the output. After running it, the output schema is automatically updated and subsequent operators turn red in case the UI parameters selection is not valid anymore.



**Note:** The final output schema of the Transpose operator is cleared if one of the following events occurs.

- The user changes the configuration properties of the Transpose operator.
- The user changes the input connected to the Transpose operator.
- The user clears the step run results of the Transpose operator.

In this case, the output schema transmitted to subsequent operators again becomes the partial schema defined at design time (hence, subsequent operators can turn invalid), and the user must run the Transpose operator again to transmit the new output schema.

The first column of the output is always chararray (because it is created from input header). All of the other columns are either double if all input columns (except the column chosen to define the new header if the user specified it) are numeric, or chararray otherwise.

## Unpivot (DB)

Unpivots one or more columns.



### Information at a Glance

**i Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	SQL

**i Note:** The Unpivot (DB) operator is for database data only. For Hadoop data, use the [Unpivot \(HD\)](#) operator.

The columns selected are removed from the input and flattened into the following two new columns at the end of the output data set.

- The first column, whose values are the names of the chosen columns.
- The second column, whose values are the corresponding values in the chosen columns.

## Input

A data set from a database.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	The columns to unpivot. All data types are supported.
<b>Name of Variable Column</b>	<p>The name of the first new column. This contains the names of the columns to unpivot.</p> <p><b>Note:</b> The value must be alphanumeric. (regular expression to match is : <code>"^[A-Za-z]+ \\ w*\$"</code>)</p>
<b>Name of Value Column</b>	<p>The name of the second new column. This contains the values of the columns to unpivot.</p> <p><b>Note:</b> The value must be alphanumeric. (regular expression to match is : <code>"^[A-Za-z]+ \\ w*\$"</code>)</p>
<b>Output Type</b>	<ul style="list-style-type: none"> <li>• <b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li>• <b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Drop If</b>	Specifies whether to overwrite an existing table.



Parameter	Description
<b>Exists</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

If you select X columns to unpivot from an input with Y columns and N rows, the output data set has (Y-X+2) columns and (X \* N) rows.

### Data Output

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Variable_Column	Value_Column
50	21	77	0	28	0	27	48	22	Column10	2
55	0	92	0	0	26	36	92	56	Column10	4
53	0	82	0	52	-5	29	30	2	Column10	1
37	0	76	0	28	18	40	48	8	Column10	1
37	0	79	0	34	-26	43	46	2	Column10	1
85	0	88	-4	6	1	3	83	80	Column10	5
56	0	81	0	-4	11	25	86	62	Column10	4



#### Note:

- The New Variable column contains the names of the unpivoted values in chararray format.
- For the New Value column:
  - If all columns selected to unpivot are numeric, the resulting value column is double.
  - If all columns selected to unpivot are datetime with the exact same format, the resulting value column is datetime with this same format.
  - For all other cases, the resulting value column is chararray.
- All null values are kept in the output.

## Example

Name	Mathematics	Science	English
John	90	70	50
Matt	60	40	80

After you select the Mathematics, Science, and English columns to unpivot, and specify new columns named Subject and Grade, the result is as follows:

Name	Subject	Grade
John	Mathematics	90
John	Science	70
John	English	50
Matt	Mathematics	60
Matt	Science	40
Matt	English	80

## Unpivot (HD)

Unpivots one or more columns.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark



**Note:** The Unpivot (HD) operator is for Hadoop data only. For database data, use the [Unpivot \(DB\)](#) operator.

The columns selected are removed from the input and flattened into the following two new columns at the end of the output data set.

- The first column, whose values are the names of the chosen columns.
- The second column, whose values are the corresponding values in the chosen columns.

## Input

A data set from HDFS.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	The columns to unpivot. All data types are supported.

Parameter	Description
<b>Name of Variable Column</b>	<p>The name of the first new column. This contains the names of the columns to unpivot.</p> <p><b>Note:</b> The value must be alphanumeric. (regular expression to match is : <code>"^[A-Za-z]+ \\ w*\$"</code>)</p>
<b>Name of Value Column</b>	<p>The name of the second new column. This contains the values of the columns to unpivot.</p> <p><b>Note:</b> The value must be alphanumeric. (regular expression to match is : <code>"^[A-Za-z]+ \\ w*\$"</code>)</p>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	Specifies whether to delete existing data at that path.

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

If you select X columns to unpivot from an input with Y columns and N rows, the output data set has (Y-X+2) columns and (X \* N) rows.

### Data Output

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Variable_Column	Value_Column
50	21	77	0	28	0	27	48	22	Column10	2
55	0	92	0	0	26	36	92	56	Column10	4
53	0	82	0	52	-5	29	30	2	Column10	1
37	0	76	0	28	18	40	48	8	Column10	1
37	0	79	0	34	-26	43	46	2	Column10	1
85	0	88	-4	6	1	3	83	80	Column10	5
56	0	81	0	-4	11	25	86	62	Column10	4

#### Note:

- The New Variable column contains the names of the unpivoted values in chararray format.
- For the New Value column:
  - If all columns selected to unpivot are numeric, the resulting value column is double.
  - If all columns selected to unpivot are datetime with the exact same format, the resulting value column is datetime with this same format.
  - For all other cases, the resulting value column is chararray.
- All null values are kept in the output.

## Example

Name	Mathematics	Science	English
John	90	70	50
Matt	60	40	80

After you select the Mathematics, Science, and English columns to unpivot, and specify new columns named Subject and Grade, the result is as follows:

Name	Subject	Grade
John	Mathematics	90
John	Science	70
John	English	50
Matt	Mathematics	60
Matt	Science	40
Matt	English	80

## Unstack

Takes an HDFS data set in stacked format and produces an unstacked (wide) HDFS data set using user-specified grouping and pivot columns.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark SQL

The operation of Unstack is similar to a pivot operation, except that it is capable of operating on text data as well as numeric data, whereas a pivot operation works on numeric data only. Unstack takes multiple Columns to Keep (group by), a single column that contains the column names (pivot column), and a single column that contains the values of the new columns in the output data set (aggregate column) as inputs.

The maximum number of new columns that can be produced is 10,000. If the pivot column contains more than 10,000 distinct values, an error results.

Missing data appears as null values in the output data set. Unstack handles all column types as inputs except date/time and Boolean.

## Input

A single tabular data set.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>New Column Names</b>	A column name that contains the new column header.

Parameter	Description
<b>New Column Values</b>	A column name that contains the values.
<b>Aggregation Method</b>	The method that aggregates the new values.
<b>Columns to Keep</b>	Any number of non-date/time and Boolean columns.
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit</b></li> </ul>



Parameter	Description
<b>Optimization</b>	<b>Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.

## Output

### Visual Output

A tabular preview of the output data set. Summary output includes parameters selected, input data size, and output location.

### Data Output

A single tabular data set.

**i** **Note:** Unstack is a 'semi-terminal' operator. A partial schema can be transmitted to subsequent operators at design time, but you must run the operator for subsequent operators to see the final output schema.

## Example

In the following illustration, **name** and **date** are the row identifiers, while **header** and **value** are the columns to pivot. To complete the operation, Unstack pivots the text values (text value 1 and text value 2), and then places the values in the proper row. Where **name** equals **cc**, there is only one date (**date4**) and two headers. This results in one row. Where **name** equals **bb**, there are three dates, each with one header. This results in three rows. Columns in which the header values do not exist for the dates are left empty.

name	date	header	value		name	date	text value 1	numerical value 1	text value 2
aa	date1	text value 1	xx		aa	date1	xx		
bb	date1	text value 1	yy		bb	date1	yy		
bb	date2	numerical value 1	500		bb	date2		500	
bb	date3	text value 2	pp		bb	date3			pp
cc	date4	text value 1	zz		cc	date4	zz	7010	
cc	date4	numerical value 1	7010						

## Variable (DB)

Use to define variables created from data fields of the input data set, forming a new table or view.



### Information at a Glance

Parameter	Description
Category	Transform
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

**i Note:** The Variable (DB) operator is for database data only. For Hadoop data, use the [Variable \(HD\)](#) operator.

**! Important:** The created variables are static in nature. They cannot dynamically change during runtime.

The Variable operator also allows users to divide the data rows into quantiles, adding quantile variables to the data. Dividing the data into such smaller and smaller divisions (quantiles) provides an understanding of the overall data distribution patterns.

### Input

An operator that can output a data set.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Variables</b>	Define the expression(s) to create the new Variable column(s).  For details, see <a href="#">Define Variables dialog</a> .
<b>Quantile Variables</b>	<p>If the new variable to create is a quantile variable, select the required column(s) to use for deriving the quantiles.</p> <p>The possible quantile types are <b>Average Ascend</b> (which automatically creates the bins) and <b>Customize</b> (which manually defines the variable bins).</p> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p><b>Note:</b> Automated variable binning can run on only databases that support NTILE, such as Greenplum, PostgreSQL, Oracle, and SQL Server. It is not supported on databases such as Teradata, MySQL, and MariaDB that do not currently support NTILE.</p> </div> <p>See <a href="#">Define Quantile Variables dialog</a> for more information.</p>
<b>Columns</b>	See <a href="#">Select Columns dialog</a> .
<b>Output Type</b>	<ul style="list-style-type: none"> <li>• <b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li>• <b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage</b>	Advanced database settings for the operator output. Available only for

Parameter	Description
<b>Parameters</b>	<b>TABLE</b> output.  See <a href="#">Storage Parameters dialog</a> for more information.
<b>Drop If Exists</b>	Specifies whether to overwrite an existing table. <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

The data rows of the output table or view displayed. The new Variable columns, such as **morethan3k** in the example below, are displayed.

To see all of the data rows in addition to the derived variables, select all columns for the **Columns** parameter.

Results - Variable	
Operation created output with 2 columns and 50,000 rows. Visualization of rows limited to 15. Please refer to output table for full results.	
morethan3k	monthly_income
0	1,956.5922
1	3,983.4564
1	3,595.3707
0	1,729.9645
0	2,507.1652
0	2,126.5996

### Data Output

A data set of the newly created table or view.

## Additional Notes

The Variable operator also provides the following useful functions.

- Parses data fields stored in a key-value pair format, such as JSON, dictionaries, and database STRUCT formats. For details, see [Key-Value Pairs Parsing Example using the Variable Operator](#).
- Converts datetime formats. For details, see [datetime Format Conversion Examples](#).

## Variable (HD)

Use to define variables created from data fields of the input data set, forming a new table or view.



### Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Pig

**i Note:** The Variable (HD) operator is for Hadoop data only. For database data, use the [Variable \(DB\)](#) operator.

**! Important:** The created variables are static in nature. They cannot dynamically change during runtime.

The Variable operator also allows users to divide the data rows into quantiles, adding quantile variables to the data. Dividing the data into such smaller and smaller divisions (quantiles) provides an understanding of the overall data distribution patterns.

## Input

An operator that can output a data set.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Variables</b>	Define the expression(s) to create the new Variable column(s).  For details, see <a href="#">Define Variables dialog</a> and <a href="#">Define Quantile Variables dialog</a> .
<b>Quantile Variables</b>	If the new variable to create is a quantile variable, select the required column(s) to use for deriving the quantiles.  The possible quantile types are <b>Average Ascend</b> (which automatically creates the bins) and <b>Customize</b> (which manually defines the variable bins).
<b>Columns</b>	See <a href="#">Select Columns dialog</a> .
<b>Store Results?</b>	Specifies whether to store the results. <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	The HDFS directory where the results of the operator are stored. This is the main directory, the subdirectory of which is specified in <b>Results Name</b> . Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.

Parameter	Description
<b>Results Name</b>	The name of the file in which to store the results.
<b>Overwrite</b>	<p>Specifies whether to delete existing data at that path and file name.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

The data rows of the output table or view displayed. The new Variable columns, such as **morethan3k** in the example below, are displayed.

To see all of the data rows in addition to the derived variables, select all columns for the **Columns** parameter.

Results - Variable	
Operation created output with 2 columns and 50,000 rows. Visualization of rows limited to 15. Please refer to output table for full results.	
morethan3k	monthly_income
0	1,956.5922
1	3,983.4564
1	3,595.3707
0	1,729.9645
0	2,507.1652
0	2,126.5996

## Data Output

A data set of the newly created table or view.

## Additional Notes

The Variable operator also provides the following useful functions.

- Parse data fields stored in a key-value pair format, such as JSON, dictionaries, and database STRUCT formats. For details, see [Key-Value Pairs Parsing Example using the Variable Operator](#).
- Convert datetime formats. For more information, see [datetime Format Conversion Examples](#).

## Wide Data Variable Selector - Chi Square / Anova

From a very large data set (that is, one whose variables number in the thousands or millions), produces a new data set with correlations and significance statistics for each predictor (X) variable against a user-specified dependent (Y) variable.





## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark SQL

## Algorithm

For each predictor (X) variable, the operator computes the correlation against the dependent (Y) variable. If categorical predictors exist, they are converted to continuous predictors using impact coding before the correlations are calculated. The algorithm does two passes through the data, one to collect the dependent values and another to calculate the correlations.

**i Note:** For this operator, the dependent variable must be categorical. If your dependent variable is continuous, then use the operator [Wide Data Variable Selector - Correlations](#)

The  $t$  statistic and corresponding  $p$  value calculations use the following formula.

$$\text{Test statistic: } t^* = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}}$$

$$\text{P Value: } p = 2.0 * (1.0 - \text{StudentsT}(n-2).cdf(abs(t^*)))$$

Scalability should not be limited by anything other than available cluster resources. The algorithm makes two passes through the data: one to collect the dependent values, and another to calculate the correlations.

## Input

A single tabular data set that contains key-value pairs of variables and values in stacked format, with `variable_names`, `continuous_values`, and `categorical_values`, and `row_id` columns.

### Bad or Missing Data

Missing data is not present in the input table. There is a minimum of two values for each predictor and dependent variable. Missing data is casewise deleted.

### Error and Exception Handling

The operation checks for validity of the dependent variable specification. See the **Algorithm** section for more information.

- If the dependent variable is categorical, then it should be in a categorical values column and have discrete values (string, long, int).
- If the dependent variable is continuous, then use the operator [Wide Data Variable Selector - Correlations](#).

If there are not enough cases to calculate correlation for a variable (at least 2), then the operation returns NaN.

If there are not enough cases to calculate  $t$  statistic and  $p$  value (at least 3), then the operation returns 0 and 1, respectively.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Variable Name</b>	The name of the dependent variable against which the correlation is computed. The dependent variable must be categorical. If it is continuous, then use the operator <a href="#">Wide Data Variable Selector - Correlations</a> .  Required.

Parameter	Description
<b>Variables Column</b>	The name of the column that contains the dependent variable.
<b>Continuous Values Column</b>	The name of the column that contains continuous predictor values.
<b>Categorical Values Column</b>	The name of the column that contains the categorical predictor values. Required.
<b>Row ID Column</b>	The name of the column that contains the row ID numbers. Required.
<b>Number of Bins</b>	The number of bins used for the correlation. The default is 10.
<b>Chi Square Output</b>	Can be one of the following: <ul style="list-style-type: none"> <li>• <b>Anova</b></li> <li>• <b>Chi-Square</b></li> <li>• <b>Chi-Square and p values</b></li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	Specifies whether to delete existing data at that path. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Storage Format</b>	Select the format in which to store the results. The storage format is determined by your type of operator.  Typical formats are <b>Avro</b> , <b>CSV</b> , <b>TSV</b> , or <b>Parquet</b> .
<b>Compression</b>	Select the type of compression for the output.  Available Parquet compression options.

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

A tabular preview of the output data set, which includes **Output** and **Summary** tabs.

#### Output

A single tabular data set containing correlations for each predictor along with significance statistics.

#### Summary

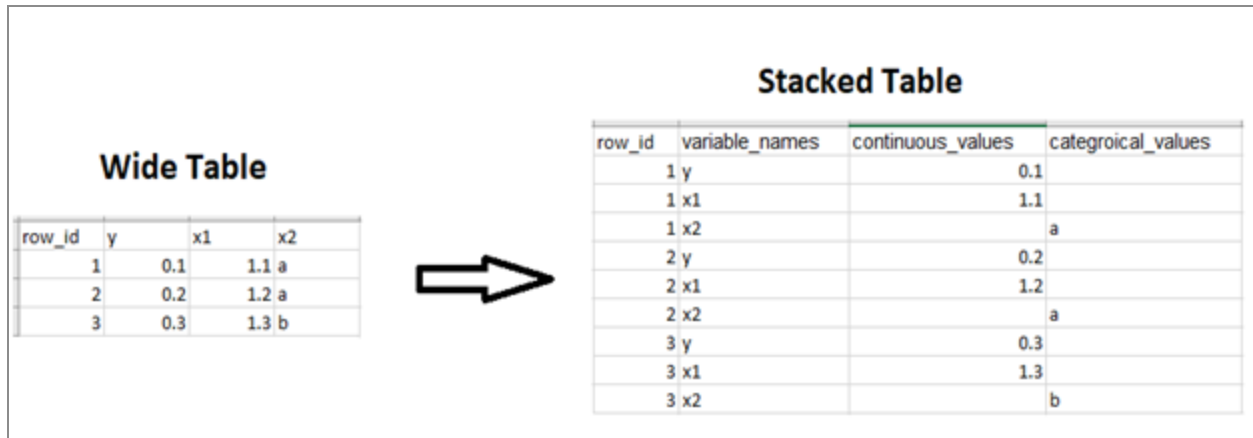
The default summary, which includes parameters selected, input data size, and output location.

### Data Output

A single tabular data set that contains s for each predictor, along with significance statistics.

## Example

The following example shows the relationship between a wide table and the stacked table input the operator requires.



## Wide Data Variable Selector - Correlations

From a very large data set (that is, one whose variables number in the thousands or millions), produces a new data set with correlations and significance statistics for each predictor (X) variable against a user-specified dependent (Y) variable.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark SQL

## Algorithm

For each predictor (X) variable, the operator computes the correlation against the dependent (Y) variable. If categorical predictors exist, they are converted to continuous predictors using impact coding before the correlations are calculated. The algorithm does two passes through the data, one to collect the dependent values and another to calculate the correlations.

**i Note:** For this operator, the dependent variable must be continuous. If your dependent variable is categorical, then use the operator [Wide Data Variable Selector - Chi Square / Anova](#).

The  $t$  statistic and corresponding  $p$  value calculations use the following formula.

$$\text{Test statistic: } t^* = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}}$$

$$\text{P Value: } p = 2.0 * (1.0 - \text{StudentsT}(n-2).cdf(abs(t^*)))$$

Scalability should not be limited by anything other than available cluster resources. The algorithm makes two passes through the data: one to collect the dependent values, and another to calculate the correlations.

## Input

A single tabular data set that contains key-value pairs of variables and values in stacked format, with `variable_names`, `continuous_values`, and `categorical_values`, and `row_id` columns.

### Bad or Missing Data

Missing data is not present in the input table. There is a minimum of two values for each predictor and dependent variable. Missing data is casewise deleted.

### Error and Exception Handling

The operation checks for validity of the dependent variable specification.

- If the dependent variable is continuous, then it should be in a continuous values

column and have numeric values (double, float, long, int).

- If the dependent variable is categorical, then use the operator [Wide Data Variable Selector - Chi Square / Anova](#).

If there are not enough cases to calculate correlation for a variable (at least 2), then the operation returns NaN.

If there are not enough cases to calculate  $t$  statistic and  $p$  value (at least 3), then the operation returns 0 and 1, respectively.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Variable Name</b>	The name of the dependent variable against which the correlation is computed. Required.
<b>Variables Column</b>	The name of the column that contains the continuous dependent variable.
<b>Continuous Values Column</b>	The name of the column that contains the continuous predictor values. If the <b>Dependent Variable Name</b> is specified as continuous, then this value is required.
<b>Categorical Values Column</b>	The name of the column that contains the categorical predictor values. If the <b>Dependent Variable Name</b> is specified as continuous, then this value is required.
<b>Row ID Column</b>	The name of the column that contains the row ID numbers. Required.
<b>Number of Folds</b>	The number of folds used in cross-validated impact coding. Range of 2 - 98.
<b>Threshold for Grand Mean Replacement</b>	An integer threshold below which the dependent's mean is used as an impact coding value. Range of 0 to the max integer value.

Parameter	Description
<b>Random Seed</b>	An integer value to use as the seed for random number generation when splitting the data into folds. Range of 0 to max integer value.
<b>Correlation Computation</b>	Specify the method to use to compute the correlation. Can be either <b>Spark SQL</b> (the default) or <b>TDS</b> .
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	Specifies whether to delete existing data at that path. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Storage Format</b>	Select the format in which to store the results. The storage format is determined by your type of operator.  Typical formats are <b>Avro</b> , <b>CSV</b> , <b>TSV</b> , or <b>Parquet</b> .
<b>Compression</b>	Select the type of compression for the output.  Available Parquet compression options. <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> Available Avro compression options. <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Advanced Spark Settings Automatic</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit</b></li> </ul>



Parameter	Description
<b>Optimization</b>	<b>Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.

## Output

### Visual Output

A tabular preview of the output data set, which includes **Output** and **Summary** tabs.

#### Output

A single tabular data set containing correlations for each predictor along with significance statistics.

#### Summary

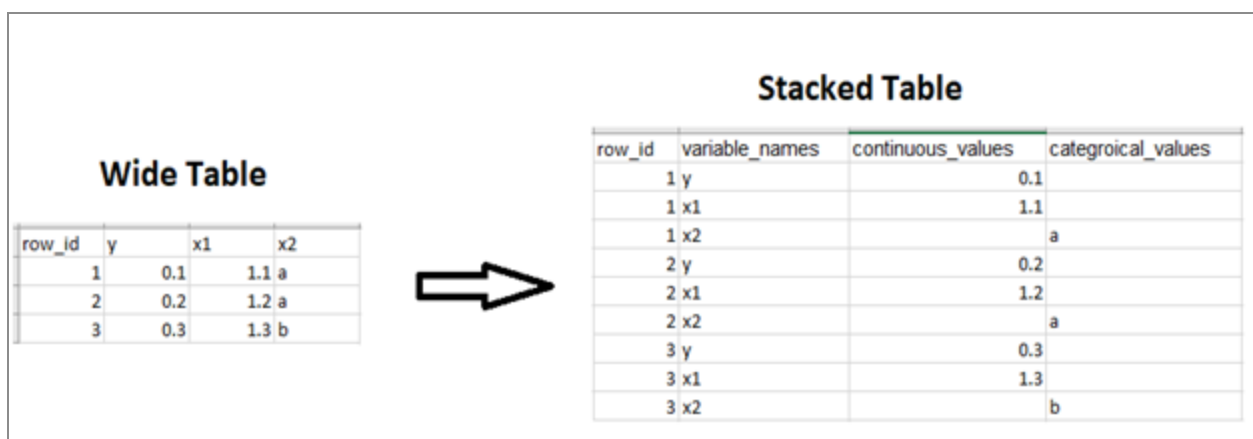
The default summary, which includes parameters selected, input data size, and output location.

### Data Output

A single tabular data set that contains s for each predictor, along with significance statistics.

## Example

The following example shows the relationship between a wide table and the stacked table input the operator requires.



## Window Functions - Aggregate

Unlike regular aggregate functions calls, allows you to create aggregate variables for each input row, based on the specified frame (with an optional order).



### Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

The Aggregate operator supports the count, max, min, sum, mean, first\_value, and last\_value functions. A potential use case might be determining the average number of sales per department store, averaging the amount over the last 30 days or just for the last 10 transactions.

The window functions operators allow you to compute database-like window functions on top of Hadoop by leveraging Spark SQL. To learn more about how window functions are implemented in Spark SQL, see [this tutorial](#).

A window function calculates a return value for every input row of an input based on a specific group of user-defined rows called the frame. Every input row has a unique frame associated with it.

TIBCO Data Science - Team Studio provides two other distinct window functions operators: [Window Functions - Rank](#) and [Window Functions - Lag/Lead](#).

**i Note:** Each operator can compute several window functions on several columns at once, but for a specific ordered partition. If you need to compute the same window functions on a different (ordered) partition, you can copy the operator and modify the partition/frame/order-related parameters on the copy.

## Input

One data set from HDFS. The data set must have some numeric columns on which to compute numeric aggregations, and might include partition by column(s) and order by columns of any type. Datetime columns must have a format specified in the input (for example, Datetime 'MM/dd/yy').

## Bad or Missing Values

**Dirty data:** When parsing delimited data, the window functions operators remove dirty data (such as strings in numeric columns, doubles in integer columns, or rows with the incorrect number of values) as it parses. These rows are silently removed because Spark is incapable of handling them.

**Null values:** Before calculating any of the window functions, the operator filters any rows that contain null values in the **Order By** column selected. The operator then processes these rows with null values according to the value of the **Write Rows Removed Due to Null Data To File** parameter. The number of rows removed due to null data is reported in the **Summary** tab of the visual output (if you do not select **Do Not Write or Count Null Rows (Fastest)**).

## Restrictions

**Wide data:** This operator works quickly on long data, but performance might slow down dramatically if window functions are calculated on thousands of columns. Increasing Spark's executor memory might improve performance.

**Datetime columns:** Input datetime columns must have a format specified in the input (for example, Datetime 'MM/dd/yy'); otherwise, the operator returns null values for the whole column.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Partition By</b>	The column(s) to partition by. You must enter a value for <b>Partition By</b> and/or <b>Order By</b> .
<b>Order By</b>	<p>The column to order each partition by (all data types are supported).</p> <ul style="list-style-type: none"> <li>• If the <b>Range</b> option is selected in <b>Window Frame Boundaries</b>, the <b>Order By</b> column must be numeric.</li> <li>• The default ordering is <b>Ascending</b>.</li> <li>• If no <b>Order By</b> column is specified, the frame must be unbounded (no values entered for <b>Frame Start</b> and <b>Frame End</b>).</li> </ul> <p>You must enter a value for <b>Partition By</b> and/or <b>Order By</b>.</p>
<b>Window Frame Boundaries</b>	<p>Specify frame boundaries type of <b>Rows</b> or <b>Range</b>.</p> <p>If <b>Rows</b> (the default) is selected, the specified <b>Frame Start/Frame End</b> value refers to the number of rows before/after the current row.</p> <p>If <b>Range</b> is selected, the specified <b>Frame Start/Frame End</b> value refers to the number of units off (in <b>Order By</b> column) before/after the current row.</p> <p>If <b>Range</b> is selected, the <b>Order By</b> column must be numeric.</p>
<b>Frame Start (inclusive)</b>	<p>Where the frame starts (rows or units off from the current row).</p> <ul style="list-style-type: none"> <li>• A negative integer refers to rows or range before the current row.</li> <li>• A positive integer refers to rows or range after the current row.</li> <li>• If no value is entered (the default), the frame starts from the beginning of the partition (that is, behaves like the SQL expression UNBOUNDED PRECEDING).</li> </ul>

Parameter	Description
<b>Frame End (inclusive)</b>	<p>Where the frame ends (rows or units off from the current row).</p> <p>Default value is 0 (= current row).</p> <div> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• A negative integer refers to rows or range units before the current row.</li> <li>• A positive integer refers to rows or range units after the current row.</li> <li>• If no value is entered, the frame ends at the ending of the partition (that is, behaves like the SQL expression UNBOUNDED FOLLOWING).</li> <li>• If both <b>Frame Start</b> and <b>Frame End</b> are specified, <b>Frame End</b> must be greater than <b>Frame Start</b>.</li> </ul> </div>
<b>Calculate Count</b>	<p>Specify whether the operator reports the number of rows in the frame selected from the current row.</p> <p>Default value: <b>No</b>.</p>
<b>Find Maximum</b>	The maximum value for each of these columns (within each partition and for the frame selected).
<b>Find Minimum</b>	The minimum value for each of these columns (within each partition and for the frame selected).
<b>Calculate Sum</b>	The sum for each of these columns (within each partition and for the frame selected).
<b>Calculate Mean</b>	The mean for each of these columns (within each partition and for the frame selected).
<b>Find First Value</b>	<p>The first value for each of these columns (within each partition and for the frame selected).</p> <div> <p><b>Note:</b> The output column(s) have the same data type as the input column(s) selected.</p> </div>

Parameter	Description
<b>Find Last Value</b>	<p>The last value for each of these columns (within each partition and for the frame selected).</p> <p><b>Note:</b> The output column(s) have the same data type as the input column(s) selected.</p>
<b>Suffix for New Columns</b>	A suffix to append to new columns created (optional).
<b>Columns to Keep</b>	The columns to keep in the output.
<b>Write Rows Removed Due to Null Data To File</b>  *required	<p>Rows with null values in the selected <b>Order By</b> column are removed from the analysis. This parameter allows you to specify whether the data with null values is written to a file.</p> <p>The file is written to the same directory as the rest of the output. The file name is suffixed with <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - Remove null-value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - Remove null-value data but do not count and display in the result UI.</li> <li>• <b>Write All Null Rows to File</b> - Remove null-value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

Each operator returns visual output with two tabs: **Output** and **Summary**.

- **Output** - A preview of the data output (see Example section below).

The following image shows an output example.

routing_number	account_number	date	transaction_type	amount	count_last10	amount_max_last10	amount_min_last10	amount_mean_last10	amount_first_val_last10	date_last_val_last10	transaction_type_last_val_last10
1.11111127e+8	2.22e+11	04/01/14	written_check	617	1	617	617	617	617	04/01/14	written_check
1.11111127e+8	2.22e+11	04/02/14	written_check	280	2	617	280	448.5	617	04/02/14	written_check
1.11111127e+8	2.22e+11	04/02/14	written_check	3,713	3	3,713	280	1,536.66666667	617	04/02/14	written_check
1.11111127e+8	2.22e+11	04/02/14	written_check	501	4	3,713	280	1,277.75	617	04/02/14	written_check
1.11111127e+8	2.22e+11	04/02/14	return	3,142	5	3,713	280	1,850.6	617	04/02/14	return
1.11111127e+8	2.22e+11	04/02/14	written_check	843	6	3,713	280	1,516	617	04/02/14	written_check
1.11111127e+8	2.22e+11	04/02/14	return	4,265	7	4,265	280	1,908.71428571	617	04/02/14	return
1.11111127e+8	2.22e+11	04/03/14	return	661	8	4,265	280	1,752.75	617	04/03/14	return
1.11111127e+8	2.22e+11	04/03/14	written_check	467	9	4,265	280	1,609.88888889	617	04/03/14	written_check

- **Summary** - A list of the selected parameters, a summary of the number of rows removed due to null data, and a message about where the results were stored.

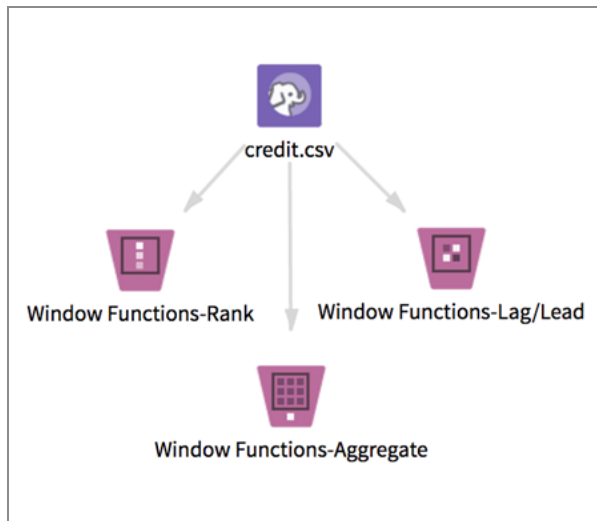
Results - Window Functions (Rolling Aggregations)	
<b>Output</b>	<b>Parameters selected</b>
<b>Summary</b>	Partition by: routing_number,account_number Order by: date Window Frame type: Rows Window Frame start: 10 row(s) preceding Window Frame end: 0 row(s) following Calculate count: Yes Max cols: amount Min cols: amount Sum cols: Mean cols: amount Count Distinct cols: First Value cols: amount Last Value cols: date,transaction_type Suffix for new columns: _last10 Columns to keep (first 50): routing_number, account_number, date, transaction_type, amount
	<b>Output</b>
	Input data size: 5000 rows
	Input size after removing rows due to null values in 'Order by' column: 5000 rows
	No data removed due to null values in 'Order by' column
	The output of the operator is stored at /tmp/alpine_out/emilie/Test_Window_Functions_3_COs_12148/Window_Functions__Rolling_Aggregations__1471398847752

## Data Output

A data set that can be accepted by any TIBCO Data Science - Team Studio operator that accepts HDFS data as input.



## Example



## Window Functions - Lag/Lead

For several columns and offset values ( $n$ ), returns the value of the column that is  $n$  rows before (lag) or after (lead) the current row.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

Lag is used to access data from  $n$  rows prior, and allows the current row to access that data. Lead follows the same logic, but is used to access data from  $n$  rows ahead. An example use case is finding previous and next order dates for customers. Given a data set with a customer ID and an order date, using 1 as the parameter for lag and 1 as the parameter for lead, the result is two additional columns: Lag returns the data prior to the current row, and Lead returns the data from one row ahead. Once this information is known, a potential next step could be determining the number of days between customer orders.

TIBCO Data Science - Team Studio provides two other distinct Window Functions operators: [Window Functions - Aggregate](#) and [Window Functions - Rank](#). The window functions operators allow you to compute database-like window functions on top of Hadoop by leveraging Spark SQL. To learn more about how window functions are implemented in Spark SQL, see [this tutorial](#).

A window function calculates a return value for every input row of an input based on a specific group of user-defined rows called the frame. Every input row has a unique frame associated with it.

**i Note:** Each operator can compute several window functions on several columns at once, but for a specific ordered partition. If you need to compute the same window functions on a different (ordered) partition, you can copy the operator and modify the partition/frame/order-related parameters on the copy.

## Input

This operator takes one data set from HDFS. It must have some numeric columns to compute numeric aggregations on, and might include partition by column(s) and order by column of any type.

Datetime columns must have a format specified in the input (for example, Datetime 'MM/dd/yy').

### Bad or Missing Values

Dirty data: When parsing delimited data, the window functions operators remove dirty data (such as strings in numeric columns, doubles in integer columns, or rows with the incorrect number of values) as it parses. These rows are silently removed because Spark is incapable of handling them.

Null values: Before calculating any of the window functions, the operator filters any rows that contain null values in the **Order By** column selected. The operator then processes these rows with null values according to the value of the **Write Rows Removed Due to Null Data To File** parameter. The number of rows removed due to null data is reported in the **Summary** tab of the visual output (if you do not select Yes, but do not count removed rows (faster)).

## Restrictions

Wide data: This operator works quickly on long data, but performance might slow down dramatically if window functions are calculated on thousands of columns. Increasing Spark's executor memory might improve performance.

Datetime columns: Input datetime columns must have a format specified in the input (for example, Datetime 'MM/dd/yy'); otherwise, the operator returns null values for the whole column.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Partition By</b>	Column(s) to partition by.
<b>Order By</b>	The column by which to order each partition (all data types are supported). Default ordering: <b>Ascending</b> .
<b>Calculate Lag</b>	<p>The lag value for each of these columns within an ordered partition, and for each value in <b>Lag Offset Values</b>.</p> <p>Lag: for an offset value <math>n</math>, the lag function returns the value that is <math>n</math> rows before the current row. For example, an offset of 1 returns the following row at any given point in the window partition (or a null value for the last row of the partition).</p>
<b>Lag Offset</b>	If some columns are selected in <b>Calculate Lag</b> , specify the integer comma-

Parameter	Description
<b>Values</b>	<p>separated list of offsets for which to compute the lag value(s). For example:</p> <p>1,2,4 (= compute lag values of 1, 2, and 4 rows before each column selected in <b>Calculate Lag</b>).</p>
<b>Calculate Lead</b>	<p>The lead value for each of these columns within an ordered partition, and for each value in <b>Lead Offset Values</b>.</p> <p>Lead: For an offset value <math>n</math>, the lead function returns the value that is <math>n</math> rows after the current row. For example, an offset of 1 returns the previous row at any given point in the ordered window partition (or a null value for the first row of the partition).</p>
<b>Lead Offset Values</b>	<p>If some columns are selected in <b>Calculate Lead</b>, specify the integer comma-separated list of offsets for which to compute the lead value(s). For example:</p> <p>1,2,4 (= compute lag values of 1, 2, and 4 rows after each column selected in <b>Calculate Lead</b>).</p>
<b>Columns to Keep</b>	The columns to keep in the output.
<b>Write Rows Removed Due to Null Data To File</b>	<p>Specify whether rows with null values in the <b>Order By</b> column selected are removed from the analysis and written to a file. The file is written to the same directory as the rest of the output. The file name is suffixed with <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - Remove null-value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - Remove null-value data but do not count and display in the result UI.</li> <li>• <b>Write All Null Rows to File</b> - Remove null-value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	Select the format in which to store the results. The storage format is determined by your type of operator.

Parameter	Description
	Typical formats are <b>Avro</b> , <b>CSV</b> , <b>TSV</b> , or <b>Parquet</b> .
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

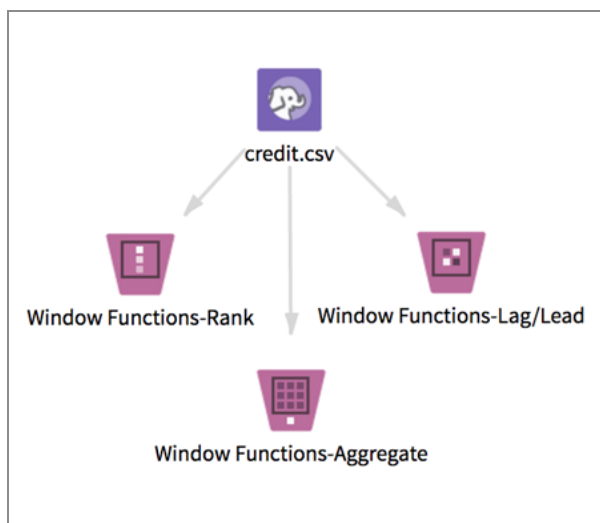
Visual output with two tabs, **Output** and **Summary**.

- **Output** - A preview of the data output.
- **Summary** - A list of the selected parameters, a summary of the number of rows removed due to null data, and a message about where the results were stored.

### Data Output

A data set that can be accepted by any TIBCO Data Science - Team Studio operator that accepts HDFS data as input.

## Example



## Window Functions - Rank

Returns the rank of each row in relation to its windowed partition.



## Information at a Glance

Parameter	Description
Category	Transform
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

The Rank operator supports the `rank`, `dense_rank`, `cumulative distribution`, and `ntile` (`n`: number of quantiles) functions. The core concept of this operator is to compute the rank/order of each row, relative to a defined grouping or partition. An example use case is ranking transactions for individual customers, within a data set that contains unique customers. In this example, the partitioning, or grouping, is the customer ID, while the data to rank/order is the transaction value amount. Transactions within each partition/customer ID are ranked and ordered starting with 1 for the highest, and counting up.

The window functions operators allow you to compute database-like window functions on top of Hadoop by leveraging Spark SQL. To learn more about how window functions are implemented in Spark SQL, see [this tutorial](#).

A window function calculates a return value for every input row of an input based on a specific group of user-defined rows called the frame. Every input row has a unique frame associated with it.

TIBCO Data Science - Team Studio provides two other distinct window functions operators: [Window Functions - Aggregate](#) and [Window Functions - Lag/Lead](#).

**i Note:** Each operator can compute several window functions on several columns at once, but for a specific ordered partition. If you need to compute the same window functions on a different (ordered) partition, you can copy the operator and modify the partition/frame/order-related parameters on the copy.

## Input

This operator takes one data set from HDFS. It must have some numeric columns on which to compute numeric aggregations, and might include partition by column(s) and order by

columns of any type.

### Bad or Missing Values

**Dirty data:** When parsing delimited data, the window functions operators remove dirty data (such as strings in numeric columns, doubles in integer columns, or rows with the incorrect number of values) as it parses. These rows are silently removed because Spark is incapable of handling them.

**Null values:** Before calculating any of the window functions, the operator filters any rows that contain null values in the **Order By** column selected. The operator then processes these rows with null values according to the value of the **Write Rows Removed Due to Null Data To File** parameter. The number of rows removed due to null data is reported in the **Summary** tab of the visual output (if you do not select Yes, but do not count removed rows (faster)).

Datetime columns must have a format specified in the input (for example, Datetime 'MM/dd/yy').

### Restrictions

**Wide data:** This operator works quickly on long data, but performance might slow down dramatically if window functions are calculated on thousands of columns. Increasing Spark's executor memory might improve performance.

**Datetime columns:** Input datetime columns must have a format specified in the input (for example, Datetime 'MM/dd/yy'); otherwise, the operator returns null values for the whole column.

### Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Partition By</b>	Column(s) by which to partition.
<b>Order By</b>	Column by which to order each partition (all data types are supported).



Parameter	Description
<b>Order</b>	Order type for the selected <b>Order By</b> column.
<b>Calculate Row Number</b>	Specifies whether the row number function returns a sequential (and unique) number starting at 1 within an ordered partition. Default value is <b>No</b> .
<b>Calculate Rank</b>	Specifies whether the rank function returns a sequential number starting at 1 within an ordered partition. Default value is <b>No</b> .
<b>Calculate Dense Rank</b>	<p>Specifies whether the dense rank function returns a sequential number starting at 1 within an ordered partition. Default value is <b>No</b>.</p> <p><b>Note:</b> The difference between rank and dense rank is that dense rank leaves no gaps in ranking sequence when ties occur (for example, if three values tie for the second place, all three have a dense rank of 2, and the next value has a dense rank of 3).</p>
<b>Calculate Cumulative Distribution</b>	<p>If <b>Yes</b>, this function returns the cumulative distribution of values within an ordered window partition; that is, the fraction of rows that are below the current row. Default value is <b>No</b>.</p> <p>If <math>N</math> = total number of rows in the partition and <math>V</math> = number of values before (and including) <math>x</math>, the equation would be <math>CUME\_DIST(X)=V/N</math>.</p>
<b>Calculate Quantiles</b>	<p>If <b>Yes</b>, returns the n-tile group ID (from 1 to <b>Number of Quantiles</b> inclusive) in an ordered window partition (equivalent to the NTILE function in SQL). Default value is <b>No</b>.</p> <p>For example, if <math>n = 4</math> (number of quantiles), the first quarter of the rows gets value 1, the second quarter gets 2, the third quarter gets 3, and the last quarter gets 4.</p> <p><b>Note:</b> If <b>Yes</b> is selected, you must specify an integer value for <b>Number of Quantiles</b>.</p>
<b>Number of Quantiles</b>	If <b>Calculate Quantiles</b> is set to <b>Yes</b> , specify the number of quantiles to return for each ordered window partition. The value must be an integer > 0.

Parameter	Description
<b>Columns to Keep</b>	Columns to keep in the output.
<b>Write Rows Removed Due to Null Data To File</b>	<p>Rows with null values in the selected <b>Order By</b> column are removed from the analysis. This parameter allows you to specify whether the data with null values is written to a file.</p> <p>The file is written to the same directory as the rest of the output. The file name is suffixed with <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - Remove null-value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - Remove null-value data but do not count and display in the result UI.</li> <li>• <b>Write All Null Rows to File</b> - Remove null-value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>

Parameter	Description
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

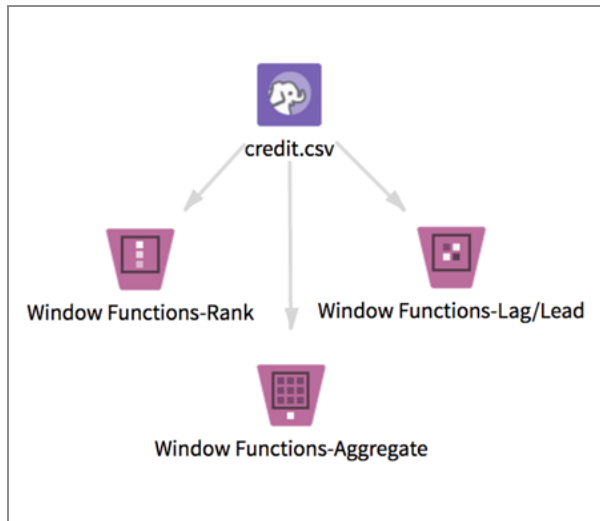
Each operator returns visual output with two tabs: **Output** and **Summary**.

- **Output** - A preview of the data output.
- **Summary** - A list of the selected parameters, a summary of the number of rows removed due to null data, and a message about where the results were stored.

### Data Output

A data set that can be accepted by any TIBCO Data Science - Team Studio operator that accepts HDFS data as input.

## Example



## Sampling Operators

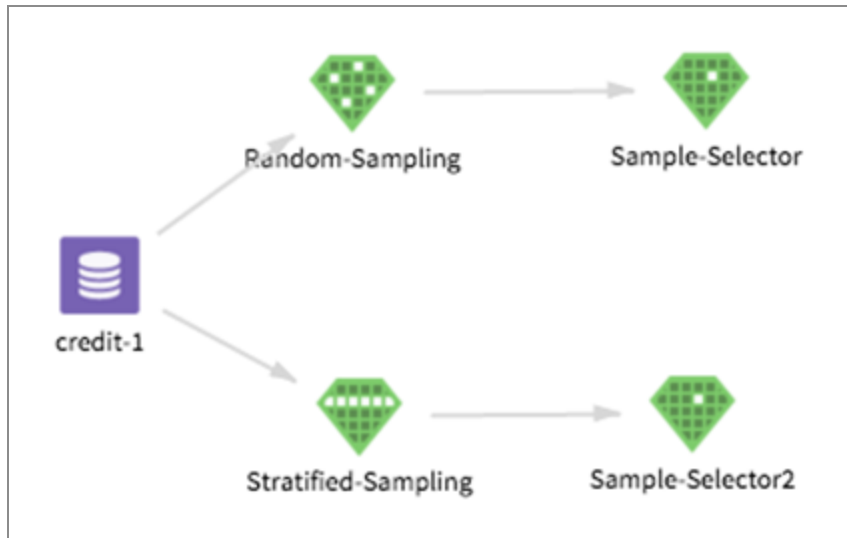
Sampling (Sample) operators provide ways to obtain a sample of a source dataset.

A model is typically created using a training dataset and then tested against a validation dataset. This is achieved in TIBCO Data Science - Team Studio by sampling the source data.

TIBCO Data Science - Team Studio provides the following primary Sampling operators:

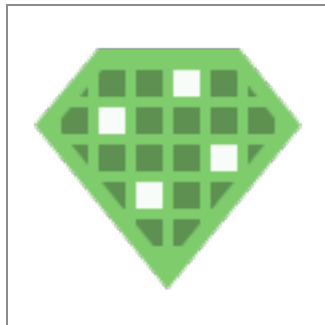
- The Random Sampling Operator
- The Stratified Sampling Operator

The third sampling operator, Sample Selector, follows either the Random Sampling or Stratified Sampling Operator to pass one of the generated sample datasets to succeeding operators in the workflow, as shown below.



## Random Sampling (DB)

Extracts data rows from the input data set and generates sample tables/views according to the sample properties (percentage or row count) the user specifies.



### Information at a Glance

**Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Sample

Parameter	Description
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a



**Note:** The Random Sampling (DB) operator is for database data only. For Hadoop data, use the [Random Sampling \(HD\)](#) operator.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Number of Samples</b>	The number of samples to generate. The samples are in the form of either database tables or views. For example, if the user inputs 3 in this field, 3 sample tables/views are generated.
<b>Sample By</b>	The size of samples by <b>Percentage</b> or by <b>Number of Rows</b> .
<b>Sample Size</b>	<p>The number of rows to generate for each sample data set. This property is interpreted in conjunction with the <b>Sample By</b> property.</p> <ul style="list-style-type: none"> <li>• <b>Percentage</b> - Specify the number of rows to include in the total sample as a percentage of the number of rows in the input data set. For example, if the user inputs 20%, 30%, 40% for three samples and the input data set contains 10,000 rows, each sample data set contains 2000, 3000, 4000 rows, and 9,000 rows is selected in total.</li> </ul>

Parameter	Description
	<p>The total aggregate percentage should be less than 100% if the <b>Disjoint</b> property is <b>true</b>. total. The</p> <ul style="list-style-type: none"> <li>• <b>Row</b> - Specify the exact number of rows to include in each sample data set.</li> </ul> <p>See <a href="#">Define Sample Size dialog</a> dialog help for more information.</p>
<b>Random Seed</b>	<p>The seed used for the pseudo-random row extraction.</p> <ul style="list-style-type: none"> <li>• The seed is the number with which the random sampling algorithm starts to generate the pseudo-random numbers.</li> <li>• The range of this value is from 0 to 1.</li> <li>• A different system-generated seed value is used if no set <b>Random Seed</b> value is specified.</li> </ul>
<b>Consistent</b>	<p>Determines whether the operator always creates the same set of random rows for each sample data generation.</p> <ul style="list-style-type: none"> <li>• <b>true</b> - sample data generation is consistent, provided that the number of samples, sample size, and the value of the Random Seed remain unchanged. If set to <b>true</b>, then <b>Replacement</b> must be <b>false</b>. Must be <b>true</b> to set <b>Key Columns</b>.</li> <li>• <b>false</b> - a different random sample is created each time the operator is run. If set to <b>false</b>, then <b>Random Seed</b> is disabled.</li> </ul> <p>Default value: <b>false</b>.</p>
<b>Replacement</b>	<p>Specifies whether this is sampling with or without replacement.</p> <ul style="list-style-type: none"> <li>• <b>true</b> - sampling with replacement.</li> <li>• <b>false</b> (the default) - sampling without replacement.</li> </ul> <p>If <b>Replacement</b> is selected, both the <b>Consistent</b> and <b>Disjoint</b> properties are set to <b>false</b> and disabled.</p>
<b>Disjoint</b>	<p>Specify whether each sample should be drawn from the entire data set, or from the remaining rows after previous samples are excluded.</p> <ul style="list-style-type: none"> <li>• If you select <b>Disjoint</b>, the same data does not appear in different</li> </ul>

Parameter	Description
	<p>samples.</p> <ul style="list-style-type: none"> <li>If, for <b>Sample by</b>, you specify the <b>Percentage</b> type, the sum of all the sample percentages should not be greater than 100.</li> </ul> <p>If set to <b>true</b>, then <b>Replacement</b> must be <b>false</b>.</p>
<b>Key Columns</b>	<p>Used in conjunction with the <b>Consistent</b> property.</p> <ul style="list-style-type: none"> <li>Click <b>Select Columns</b> to display the Select Columns dialog, which is used for selecting columns to ensure the ordering of the data before generating the pseudo-random sample data set.</li> <li>The Random Sampling operator uses these key columns to guarantee the order of the rows from the input data set, so that the generation of pseudo-random sample data sets is consistent every time.</li> <li>If no key columns are specified, the Random Sampling operator assumes that the row ordering of the input data set is consistent.</li> </ul> <p>See <a href="#">Key Columns dialog</a> for more information.</p>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	<p>Advanced database settings for the operator output. Available only for <b>TABLE</b> output.</p> <p>See <a href="#">Storage Parameters dialog</a> for more information.</p>
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li><b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li><b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>



## Output

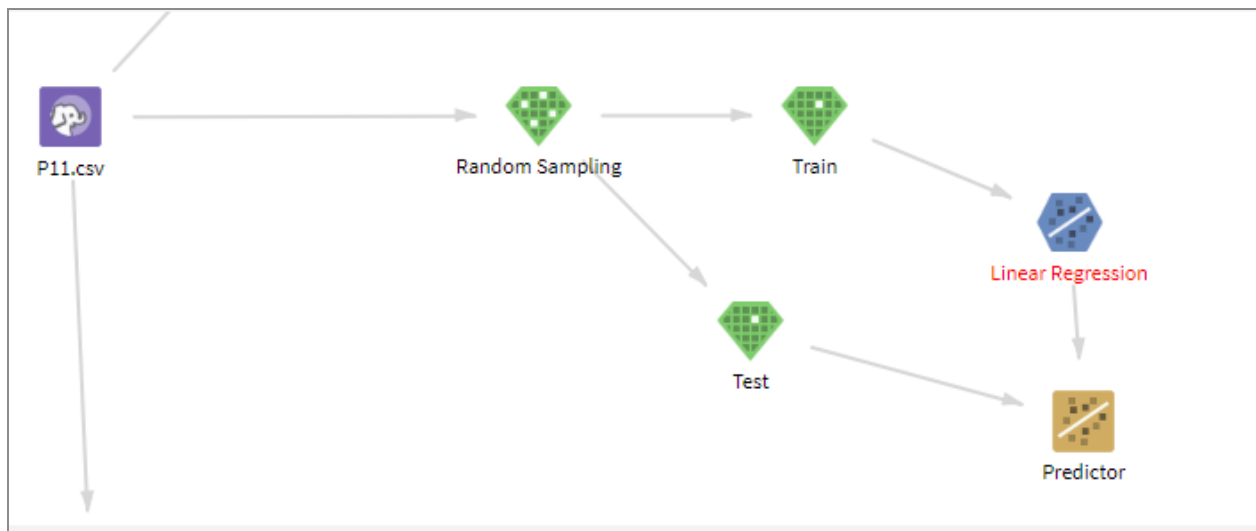
### Visual Output

The data rows of the output table/view of each generated sample displayed (up to 2000 rows of the data).

### Data Output

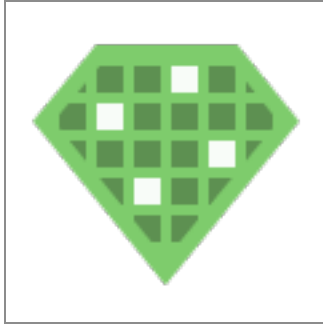
A data set of sample data tables created. Typically, the data set is passed on to a Sample Selector operator, such as Train and Test, to select a sample to use with subsequent operators.

## Example



## Random Sampling (HD)

Extracts data rows from the input data set and generates sample tables/views according to the sample properties (percentage or row count) the user specifies.



## Information at a Glance

Parameter	Description
Category	Sample
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce



### Note:

The Random Sampling (HD) operator is for Hadoop data only. For database data, use the [Random Sampling \(DB\)](#) operator.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on

Parameter	Description
	the operator.
<b>Number of Samples</b>	The number of samples to generate. The samples are in the form of Hadoop files. For example, if the user inputs 3 in this field, 3 sample files are generated.
<b>Sample By</b>	The size of samples by <b>Percentage</b> or by <b>Number of Rows</b> .
<b>Sample Size</b>	<p>The number of rows to generate for each sample data set. This property is interpreted in conjunction with the <b>Sample By</b> property.</p> <ul style="list-style-type: none"> <li>• <b>Percentage</b> - Specify the number of rows to include in the total sample as a percentage of the number of rows in the input data set. For example, if the user inputs 20%, 30%, 40% for three samples and the input data set contains 10,000 rows, each sample data set contains 2000, 3000, 4000 rows, and 9,000 rows is selected in total.</li> </ul> <p>The total aggregate percentage should be less than 100% if the <b>Disjoint</b> property is <b>true</b>. total. The</p> <ul style="list-style-type: none"> <li>• <b>Row</b> - Specify the exact number of rows to include in each sample data set.</li> </ul> <p>See <a href="#">Define Sample Size dialog</a> dialog help for more information.</p>
<b>Random Seed</b>	<p>The seed used for the pseudo-random row extraction.</p> <ul style="list-style-type: none"> <li>• The seed is the number with which the random sampling algorithm starts to generate the pseudo-random numbers.</li> <li>• The range of this value is from 0 to 1.</li> <li>• A different system-generated seed value is used if no set <b>Random Seed</b> value is specified.</li> </ul>
<b>Consistent</b>	<p>Determines whether the operator always creates the same set of random rows for each sample data generation.</p> <ul style="list-style-type: none"> <li>• <b>true</b> - sample data generation is consistent, provided that the number of samples, sample size, and the value of the Random Seed remain unchanged. If set to <b>true</b>, then <b>Replacement</b> must be <b>false</b>. Must be <b>true</b> to set <b>Key Columns</b>.</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>false</b> (the default) - a different random sample is created each time the operator is run. If set to <b>false</b>, then <b>Random Seed</b> is disabled.</li> </ul>
<b>Replacement</b>	<p>Specifies that one row of data can be selected multiple times.</p> <ul style="list-style-type: none"> <li>• <b>true</b> - sampling with replacement.</li> <li>• <b>false</b> (the default) - sampling without replacement, where one row can be selected only once.</li> </ul> <p>If set to <b>true</b>, then both the <b>Consistent</b> and <b>Disjoint</b> properties are set to <b>false</b> and disabled.</p>
<b>Disjoint</b>	<p>Specify whether each sample should be drawn from the entire data set, or from the remaining rows after previous samples are excluded.</p> <ul style="list-style-type: none"> <li>• If you select <b>Disjoint</b>, then the same data does not appear in different samples.</li> <li>• If, for <b>Sample by</b>, you specify the <b>Percentage</b> type, then the sum of all the sample percentages should not be greater than 100.</li> </ul> <p>If set to <b>true</b>, then <b>Replacement</b> must be <b>false</b>.</p>
<b>Store Results?</b>	<p>Specifies whether to store the results.</p> <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	<p>The HDFS directory where the results of the operator are stored. This is the main directory, the subdirectory of which is specified in <b>Results Name</b>. Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.</p>
<b>Results Name</b>	<p>The name of the file in which to store the results.</p>
<b>Overwrite</b>	<p>Specifies whether to delete existing data at that path and file name.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>

Parameter	Description
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options are the following.</p> <ul style="list-style-type: none"><li>• <b>GZIP</b></li><li>• <b>Deflate</b></li><li>• <b>Snappy</b></li><li>• no compression</li></ul> <p>Available Avro compression options are the following.</p> <ul style="list-style-type: none"><li>• <b>Deflate</b></li><li>• <b>Snappy</b></li><li>• no compression</li></ul>

## Output

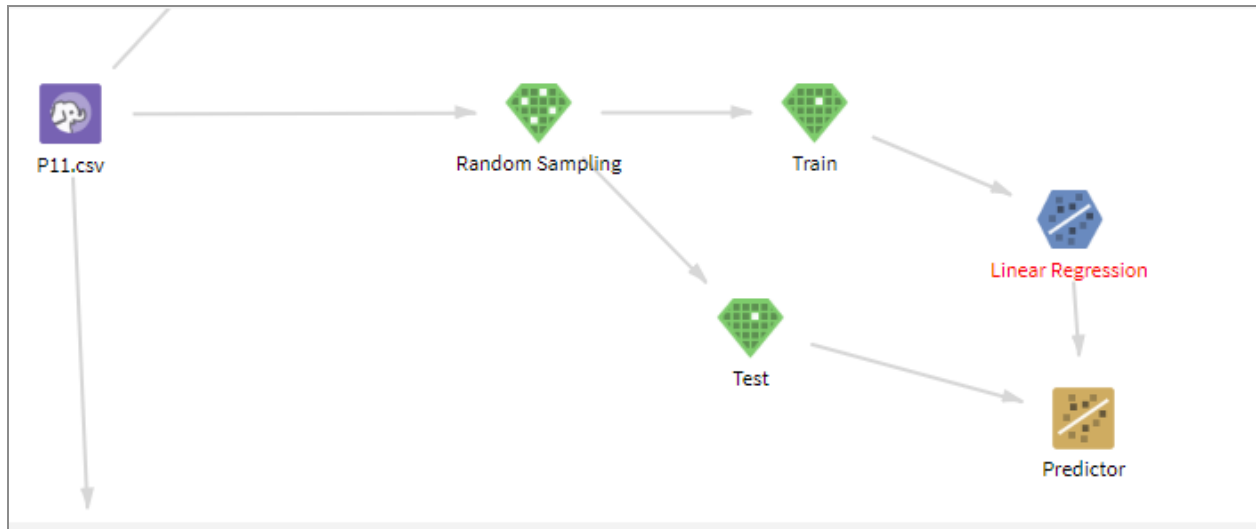
### Visual Output

The data rows of each generated sample displayed (up to 2000 rows of the data).

### Data Output

Data sets of sample files created. Typically, the data set is passed on to a Sample Selector operator, such as Train and Test, to select a sample to use with subsequent operators.

## Example



## Resampling

Changes the distribution of values in a single column. You can use this operator to either balance all values in the selected column or change the proportion of only one value. You can use it to up-sample or down-sample.



## Information at a Glance

Parameter	Description
Category	Sample
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

## Input

A Hadoop file that has at least one categorical column.

### Bad or Missing Values

Rows with null values in the selected **Column to Resample** are removed from the dataset prior to resampling. Null values in other columns do not affect the result.

## Restrictions

Input data must have at least one categorical column with less than 100 distinct values.

## Configuration

Parameter	Description				
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.				
<b>Column to Resample</b>  *required	A categorical column with less than 100 distinct values.				
<b>Balance All Values in Selected Column</b>	<p><b>Yes</b> balances all values in the selected column by up-sampling rows to match the number of rows of the most common value.</p> <p><b>Sample with Replacement</b> must be <b>Yes</b>, and any text entered into <b>Single Value from Selected Column for Resampling</b> is ignored.</p> <p>For example, if a dataset has 3 distinct values in the selected column with the following distribution</p> <table> <tr> <th>Value</th><th>Count</th></tr> <tr> <td>A</td><td>100</td></tr> </table>	Value	Count	A	100
Value	Count				
A	100				

Parameter	Description								
	<table><tr><th>Value</th><th>Count</th></tr><tr><td>B</td><td>75</td></tr><tr><td>C</td><td>50</td></tr></table>	Value	Count	B	75	C	50		
	Value	Count							
	B	75							
	C	50							
	the output has the following distribution.								
	<table><tr><th>Value</th><th>Count</th></tr><tr><td>A</td><td>100</td></tr><tr><td>B</td><td>100</td></tr><tr><td>C</td><td>100</td></tr></table>	Value	Count	A	100	B	100	C	100
	Value	Count							
	A	100							
	B	100							
	C	100							
<p><b>No</b> resamples only one value in the chosen column. The user must enter values for <b>Single Value from Selected Column for Resampling</b> and <b>Multiplier for Up-Sampling or Down-Sampling</b>. Given the same input as above, if the user chooses to resample the value B with a multiplier of 3, output distribution is:</p>									
<table><tr><th>Value</th><th>Count</th></tr><tr><td>A</td><td>100</td></tr><tr><td>B</td><td>225</td></tr><tr><td>C</td><td>50</td></tr></table>	Value	Count	A	100	B	225	C	50	
Value	Count								
A	100								
B	225								
C	50								
<b>Single Value from Selected Column for Resampling</b>	<p>Required when <b>Balance All Values in Selected Column</b> is <b>No</b>.</p> <p>A character string or a numeric value that appears in the column selected in <b>Column to Resample</b>. An error occurs when running the operator if the value does not occur in the column.</p>								



Parameter	Description
<b>Multiplier for Up-Sampling or Down-Sampling</b>	<p>Required when <b>Balance All Values in Selected Column</b> is <b>No</b>.</p> <p>A positive decimal number that is the multiplicative factor by which to resample the selected column and value.</p>
<b>Sample with Replacement</b>	<ul style="list-style-type: none"> <li>For multipliers less than or equal to 1, specify whether samples are with or without replacement.</li> <li>For multipliers greater than 1, click <b>Yes</b> to sample rows with replacement.</li> </ul>
<b>Exact (Slower)</b>	An exact calculation requires an additional pass through the data and results in slower operator execution. For non-exact resampling, the output distribution of values can vary from the expected distribution.
<b>Use Random Seed</b>	<ul style="list-style-type: none"> <li>Click <b>Yes</b> to use the specified random seed and get repeatable results.</li> <li>Click <b>No</b> to use a system-generated seed value.</li> </ul>
<b>Random Seed</b>	An integer value that is used as the seed for the pseudo-random row extraction. Only used if <b>Use Random Seed</b> is <b>Yes</b> .
<b>Write Rows Removed Due to Null Data To File</b>	<p>Rows with at least one null value in the <b>Column to Resample</b> are removed from the dataset prior to resampling. This parameter allows you to specify whether rows with null values are written to a file.</p> <p>The file is written to the same directory as the rest of the output. The filename is given a suffix of <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li><b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li><b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI.</li> <li><b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	Select the format in which to store the results. The storage format is determined by your type of operator.

Parameter	Description
	Typical formats are <b>Avro</b> , <b>CSV</b> , <b>TSV</b> , or <b>Parquet</b> .
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Outputs

### Visual Output

The **Output** tab displays a preview of the output dataset.

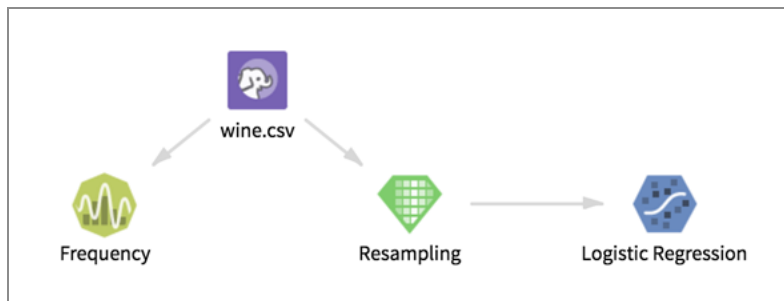
The **Summary** tab displays information about the parameters selected, the output value distribution, and the information about rows removed from the data due to null values in the selected column.

Results - Resampling	
Output Summary	<b>Selected Parameters:</b>
	Select Column for Resampling: Class
	Balance All Values in Selected Column: Yes
	Sample with Replacement: Yes
	Exact (Slower): No
	Use Random Seed: No
	<b>Input Distribution (Top 10):</b>
	Class Name                      Count
	2                                      71
	1                                      59
	3                                      48
	<b>Output Distribution (Top 10):</b>
	Class Name                      Count
	2                                      99
	3                                      97
	1                                      93

## Data Output

The resampled data set.

## Example



## Sample Selector

Connects to a preceding sample-generating operator (for example, the Random Sampling operator) and allows you to specify one of the sample data sets generated from that operator for use in succeeding operators.



## Information at a Glance

**i Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Sample
Data source type	HD
Send output to other operators	Yes
Data processing tool	n/a

## Input

A sample-generating operator - the Random Sampling operator or the Stratified Sampling operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Selected Sample</b>	Select the samples as database tables/views or Hadoop files from the preceding sample-generating operator.

## Output

### Visual Output

The data rows of the selected data sample table/view displayed.

### Data Output

A data set of the selected data sample table/view.

## Stratified Sampling

Extracts data rows from the input data set and generates sample tables/views according to the sample properties specified by users.



## Information at a Glance

Parameter	Description
Category	Sample
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

The user chooses a sample column. The proportion of all distinct values in the sample column remains unchanged in all samples generated.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Sampling Column</b>	<p>The column which the proportion of all distinct values remain unchanged in all generated samples.</p> <p>For example, if a column "gender" is chosen as the sample column and it contains 2 distinct values, "male" and "female," and there are 40% "male" values and 60% "female" values, all the samples generated contain 40% "male" values and 60% "female" values in the "gender" column.</p>
<b>Number of Samples</b>	<p>The number of samples to generate.</p> <p>The samples are in the form of database tables/views. For example, if the number of samples is 3, 3 sample tables/views are generated.</p>
<b>Sample by</b>	<p>The size of samples by <b>Percentage</b> or by <b>Number of Rows</b>.</p> <p>If <b>Sample by</b> is set to <b>Percentage</b>, the sum of all of the sample percentages should not be greater than 100.</p>
<b>Sample Size</b>	<p>Number of rows to generate for each sample data set. This property is interpreted in conjunction with the <b>Sample By</b> property.</p> <ul style="list-style-type: none"> <li>• <b>Percentage</b> - Specify the number of rows to include in the total sample as a percentage of the number of rows in the input data set. For example, if the user inputs 20%, 30%, 40% for three samples and the input data set contains 10,000 rows, each sample data set contains 2000, 3000, 4000 rows, and 9,000 rows is selected in total.</li> </ul> <p>The total aggregate percentage should be less than 100% if the <b>Disjoint</b> property is <b>true</b>.</p> <ul style="list-style-type: none"> <li>• <b>Row</b> - Specify the exact number of rows to include in each sample data set.</li> </ul> <p>See <a href="#">Define Sample Size dialog</a> dialog help for more information.</p>

Parameter	Description
<b>Random Seed</b>	<p>The seed used for the pseudo-random row extraction. The seed is the number with which the Random Sampling algorithm starts to generate the pseudo-random numbers.</p> <p>The range of this value is from 0 to 1.</p> <p>A different system-generated seed value is used if no set <b>Random Seed</b> value is specified.</p>
<b>Consistent</b>	<p>Specify whether the operator always creates the same set of rows for each sample data generation.</p> <ul style="list-style-type: none"> <li>If <b>Consistent</b> is set to <b>true</b>, sample data generation is consistent, provided that the number of samples, sample size, and value of the <b>Random Seed</b> remain unchanged.</li> </ul> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p><b>Note:</b> If a <b>Random Seed</b> value is specified, the <b>Consistent</b> property is set automatically to <b>true</b>.</p> </div> <ul style="list-style-type: none"> <li>If <b>Consistent</b> is set to <b>false</b> (the default), a different random sample is created each time.</li> </ul>
<b>Disjoint</b>	<p>Specify whether each sample should be drawn from the entire data set, or from the remaining rows after previous samples are excluded.</p> <p>If <b>Disjoint</b> is set to <b>true</b>, the same data does not appear in different samples.</p> <p>Default value: <b>false</b>.</p>
<b>Key Columns</b>	<p>Used in conjunction with the <b>Consistent</b> property.</p> <ul style="list-style-type: none"> <li>Click <b>Select Columns</b> to display the Select Columns dialog, which is used for selecting columns to ensure the ordering of the data before generating the pseudo-random sample data set.</li> <li>The Stratified Sampling operator uses these key columns to guarantee the order of the rows from the input data set, so that the generation of pseudo-random sample data sets is consistent every time.</li> <li>If no key columns are specified, the Stratified Sampling operator assumes that the row ordering of the input data set is consistent.</li> </ul>

Parameter	Description
	See <a href="#">Key Columns dialog</a> for more information.
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	Advanced database settings for the operator output. Available only for <b>TABLE</b> output.  See <a href="#">Storage Parameters dialog</a> for more information.
<b>Drop If Exists</b>	Specifies whether to overwrite an existing table. <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

The data rows of the output table/view of each generated sample displayed (up to 2000 rows of the data).

### Data Output

Data sets of sample data tables are created. The output is typically connected to a Sample Selector operator to select a sample to use with further succeeding operators.

## Modeling Operators

Modeling Algorithm (Model) operators define the modeling method, or mathematical calculations, to apply to an input dataset.



TIBCO Data Science - Team Studio algorithms, or modeling approaches, use historical data from the input data to produce a predictive model, known as model training. Each Modeling operator is associated with a Predictor/Classifier operator and, together, they can be applied to other datasets within an analytic workflow in order to predict future results for that dataset.

## Alpine Forest - MADlib

Uses the MADlib built-in function, `forest_train()`, to generate multiple decision trees, the combination of which is used to make a prediction based on several independent columns.



### Information at a Glance

Parameter	Description
Category	Model
Data source type	DB
Send output to other operators	Yes
Data processing tool	MADlib

**i Note:** This operator works only with MADlib 1.8 or higher.

Each decision tree is generated based on bootstrapped sampling and a random subset of the feature columns. The destination of the output of this operator must be an Alpine Forest Predictor (MADlib) operator. MADlib 1.8 or higher must be installed on the database. For more information, see the [official MADlib documentation](#).

## Input

The input table must have a single, categorical (string or integer) or regression (floating point) column to predict, and one or more independent columns to serve as input.

### Bad or Missing Values

Any rows in the source table that contain NULL values for the predicted or independent columns are ignored.

## Restrictions

This operator works only on databases with MADlib 1.8+ installed. Source data tables must have a numeric ID column that uniquely identifies each row in the source table. The prediction column must be integer or string for classification trees, or floating point for regression trees.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>MADlib Schema</b>	The name of schema where MADlib is installed. By default, the schema name is madlib.
<b>Model Output Schema</b>	The name of schema to use for MADlib-generated output tables.
<b>Model Output Table</b>	<p>The name of MADlib-generated output table. This table is generated by the forest trainer. The following additional tables are also generated.</p> <ul style="list-style-type: none"> <li>• A table with the same name, appended with <code>_summary</code>.</li> <li>• A table with the same name, appended with <code>_group</code>.</li> </ul>
<b>Drop If Exists</b>	<ul style="list-style-type: none"> <li>• If <b>Yes</b> (the default), drop the existing table of the same name and create a new one.</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>If <b>No</b>, stop the flow and alert the user that an error has occurred.</li> </ul>
<b>ID Column</b>	All source tables must have a numeric ID column to uniquely identify each row.
<b>Dependent Variable</b>	The name of the column to predict. If the data type of the column is floating point, the generated trees are regression trees. Otherwise, the generated trees are classification trees.
<b>Feature List</b>	The selection of one or more columns to use as independent variables to predict the dependent variable. Note that the execution time increases as more columns are selected.
<b>Number of Trees</b>	The maximum number of trees to generate. MADlib usually generates this number of trees, but the actual number may be slightly less. The default number is <b>100</b> . Note that the execution time increases as more trees are generated.
<b>Number of Random Features</b>	The number of random features to select at each split. If none is specified, the default is $\sqrt{n}$ for classification trees or $\frac{n}{3}$ for regression trees, where $n$ is the maximum number of trees.
<b>Calculate Variable Importance</b>	Whether or not to calculate variable importance. The default is <b>true</b> . If <b>false</b> , the execution time decreases.
<b>Number of Permutations for Each Feature (for Var. Importance)</b>	Variable importance is calculated by permuting the variable with random variables and computing the drop in predictive accuracy. The default value is <b>1</b> , and higher values lead to longer execution times. Usually 1 is sufficient.
<b>Maximum Tree Depth</b>	The generated trees do not exceed this depth, where the root node is at depth 0. If not specified, the default is <b>10</b> . Longer tree depths might lead to longer execution times.
<b>Minimum Observations Before Splitting</b>	The number of observations that must occur at a particular node before considering a split. If not specified, the default is <b>20</b> .

Parameter	Description
<b>Minimum Observations in Terminal Nodes</b>	The minimum number of observations in any terminal node. If not specified, the default is $\frac{n}{3}$ , where $n$ is the minimum observations before splitting.
<b>Number of Bins for Split Boundaries</b>	For continuous value features, values are quantized into bins for split boundaries. If not specified, the default is <b>100</b> . Higher values result in longer execution times.
<b>Sampling Ratio</b>	Specify only a fraction of the input for training. This must be specified as an integer value between 1 and 100. Smaller values result in faster execution times because less data is sampled. The default is <b>100</b> , where all input rows are sampled.

## Output

### Visual Output

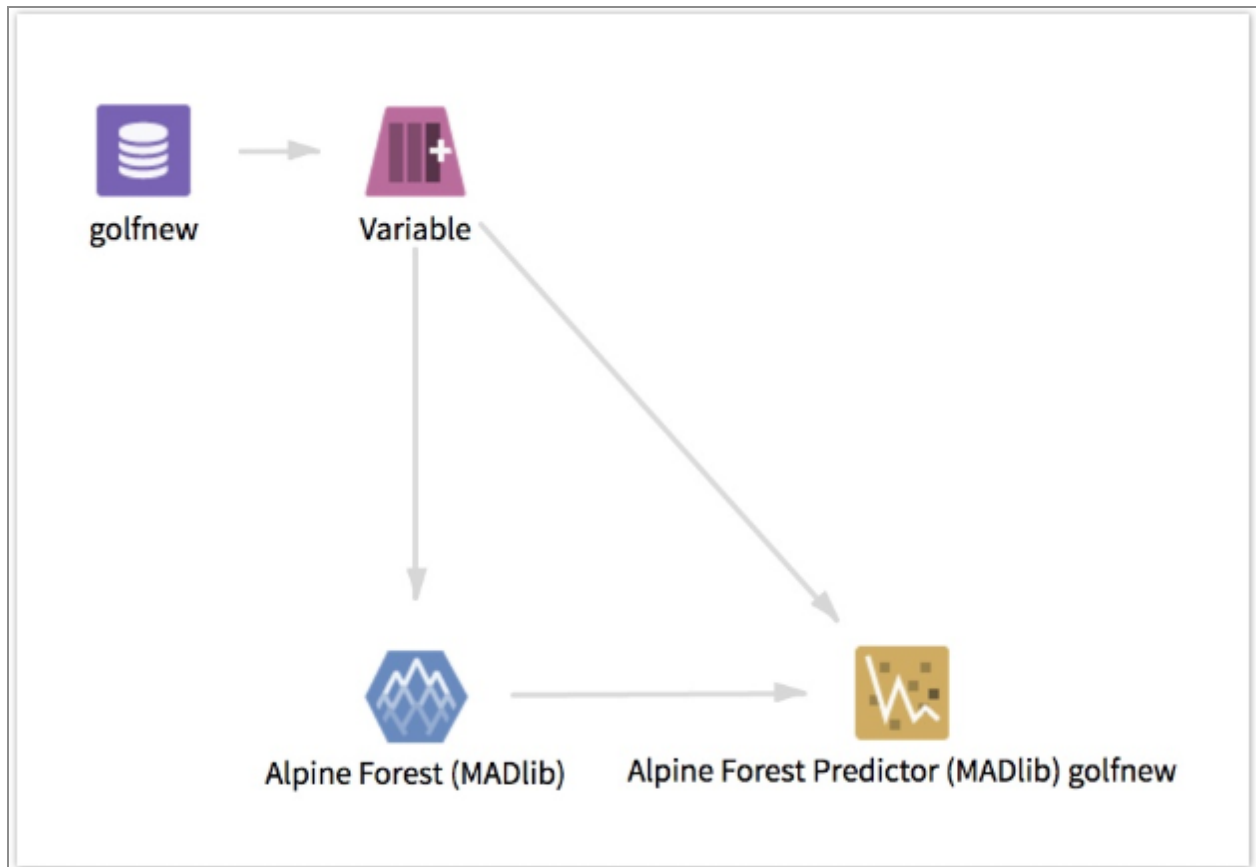
This operator has three sets of output tabs.

- The first set of tabs contains a text representation of each of the generated decision trees.
- The second tab contains DOT notation of each of the generated decision trees. DOT notation can be exported to third-party tools such as GraphViz.
- The third tab contains the raw output tables generated by MADlib.
  - The first is the model output table. The gid column represents the group ID. Grouping is not supported at this time, so this value is always 1. The sample\_id represents the tree ID. The tree column encodes each generated decision tree in binary format.
  - The second is the output summary table, which contains information about how the trees were generated. Many parameters passed to the MADlib training function appear here as columns.
  - The third tab is the grouping table, which contains one row for each set of values by which we are grouping. Because grouping is not supported, it has only one row.

## Data Output

The output of this operator must be sent to an Alpine Forest Predictor operator.

## Example



## Alpine Forest Classification

An Alpine Forest Classification model is an ensemble classification method of creating a collection of decision trees with controlled variation. Ensemble modeling is the application of many models, each operating on a subset of the data.



## Information at a Glance

Parameter	Description
Category	Modeling
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce, Spark

You usually do not need to change the default configuration settings for this operator. The main properties that are specific to Alpine Forest Classification modeling are the **Number of Features per Node**, **Number of Trees**, **Sample With Replacement**, and **Sample Percentage**.

For information about the advantages of using this model, see [Ensemble Decision Tree Modeling with Alpine Forest](#).

## Algorithm

The Alpine Forest Classification operator implements the algorithm by building multiple decision trees, each one starting by randomly selecting a subset,  $m$ , of the available attributes and then selecting a subset of data observations (rows). The final "ensemble" classification is then done by querying each tree in the "Alpine Forest" for its prediction (essentially a "vote"). The overall prediction of the Alpine Forest Classification model is the mode of the votes - that is the most common prediction from individual tree models.

The Alpine Forest of trees is grown on a dataset consisting of  $n$  data observations and  $m$  classifiers (independent variables). For each tree of the forest:

- The value of  $n$  is specified by the **Number of Trees** configuration property. This is the number of decision trees to create, each with its own randomly selected subset of data rows.
- For each individual decision tree,  $m$  out of total available Independent Variables are randomly chosen from which to determine the optimum decision tree node split. This is referred to as the random input selection methodology. The value of  $m$  is specified by the **Number of Features per Node** configuration property.

- For Alpine Forest Classification, there is a with or without replacement option. This option controls whether a data row can be chosen more than once (that is, be replaced) in the dataset used for each of the  $n$  decision trees created. This is specified by the **Sampling with Replacement** configuration property.
- The remaining data rows not included in the  $n$  decision tree dataset samples are used for automatically generated cross-validation error estimates of the model. Note: These are called the OOB (Out of Bag) Error Estimates.
- Each individual decision tree is grown according to the specified tree growth configuration parameters set for the Alpine Forest Classification operator.

In summary, the Alpine Forest Classification algorithm combines an ensemble classification or "bagging" approach with the random input selection of features, in order to construct a collection of CART decision trees with controlled variation. The individual models are combined by voting for the final classification or prediction.

## Input

A dataset that contains the dependent and independent variables for modeling.

## Configuration

### Minimal Configuration

- **Dependent Column:** the property in the dataset to be the predicted Dependent Variable. For classification models, the dependent variable must be categorical.
- **Columns:** the expected Independent Variable data columns, or properties, to use for model training.
- **Sampling Percentage:** the fraction of the data rows to be used as randomly-selected datasets for each decision tree.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent</b>	The quantity to model or predict.

Parameter	Description
<b>Column</b>	<p>Select the data column to be considered the dependent variable. Alpine Forest Classification on Hadoop only supports classification models.</p> <p>The dependent column must be a categorical variable.</p> <p>To perform a regression instead, use the <a href="#">Alpine Forest Regression</a> operator.</p>
<b>Columns</b>	<p>Select the Independent Variable column(s) to include for the decision tree training.</p> <p>At least one column must be specified.</p> <p>Click <b>Select Columns</b> to open the dialog.</p> <p>See <a href="#">Select Columns dialog</a> for more details.</p>
<b>Number of Trees</b>	<p>This specifies how many individual decision trees to train in the Alpine Forest.</p> <div> <p><b>Note:</b> Increasing the number of trees created generally increases the accuracy of the model. However, as long as enough trees are created, the Alpine Forest Classification model is not very sensitive to changing this property.</p> </div> <p>The user interface displays only a maximum of 20 tree results, even if more are generated.</p> <p>Default value: <b>10</b>.</p>
<b>Use Automatic Configuration</b>	<p>Specifies that TIBCO Data Science - Team Studio should determine all the required Alpine Forest Classification configuration properties except for the <b>Number of Trees</b> property.</p> <p>Default value: <b>true</b>.</p>
<b>Number of Features Function</b>	<p>Automatically sets a value for <b>Number of Features per Node</b>.</p> <ul style="list-style-type: none"> <li>• <b>Square Root</b> - The number of features per node is set to the square root of the number of columns (truncated to an integer), or at least to 1.</li> </ul>



Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>1/3</b> - The number of features per node is set to the <i>(number of columns)/3 (truncated to an integer)</i>, or at least to 1.</li> <li>• <b>All</b> - The number of features per node is set to the number of columns.</li> <li>• <b>User Defined</b> - Set the number of features per node value in <b>Number of Features per Node</b>.</li> </ul> <p>Default value: <b>Square Root</b>.</p>
<b>Number of Features per Node</b>	<p>Specifies <math>m</math>, the number of predictors to consider at each node during the tree building process. The Alpine Forest Classification algorithm calculates the best split for the tree based on these <math>m</math> variables that are selected randomly from the training set.</p> <ul style="list-style-type: none"> <li>• The Number of Features per Node should be much less than the number of columns specified for the <b>Columns</b> property.</li> </ul> <p>The number of features per node is the configuration property that an Alpine Forest Classification model is most sensitive to. Increasing the number per split makes each of the decision trees bigger, providing more information at each node. However, it also becomes harder to interpret for the modeler.</p> <p>Default value: <b>1</b>.</p>
<b>Sampling with Replacement</b>	<p>Specifies whether to use replacement when selecting training variable data row samples from the input dataset. This property controls whether a data row can be reused for each of the <math>n</math> training data samples collected from the available dataset rows.</p> <ul style="list-style-type: none"> <li>• Setting this value to <b>true</b> increases the training performance time because there are more possible random dataset combinations.</li> <li>• Setting this value to <b>false</b> specifies that the system does not choose a data row more than once for each of the decision trees. This setting is appropriate for a small sample of <math>n</math> data rows from a large data set. In such a case, sampling without replacement is approximately the same as sampling with replacement (where the odds of randomly choosing the same data point twice is low).</li> </ul>

Parameter	Description
	Default value: <b>true</b>
<b>Sampling Percentage</b>	<p>Specifies <math>m</math>, the number of predictors to consider at each node during the tree building process. The Alpine Forest Classification algorithm calculates the best split for the tree based on these <math>m</math> variables that are selected randomly from the training set.</p> <ul style="list-style-type: none"> <li>The Number of Features per Node should be much less than the number of columns specified for the <b>Columns</b> property.</li> </ul> <p>The number of features per node is the configuration property that an Alpine Forest Classification model is most sensitive to. Increasing the number per split makes each of the decision trees bigger, providing more information at each node. However, it also becomes harder to interpret for the modeler.</p> <p>Default value: <b>1</b>.</p>
<b>Maximum Depth</b>	<p>Specifies the "depth" of the tree, or the maximum number of decision nodes it can branch out to beneath the root node. A tree stops growing any deeper either if a node becomes empty (that is, there are no more examples to split in the current node), or if the depth of the tree exceeds this limit.</p> <ul style="list-style-type: none"> <li><b>Maximum Depth</b> is used during the tree-growth phase.</li> <li>The range of possible values is between -1 and any integer greater than 0. A value of -1 represents "no bound" - the tree can take on any size or an unlimited number of decision nodes until the nodes become empty.</li> </ul> <p>Default value: <b>-1</b></p>
<b>Minimum Size For Split</b>	<p>Specifies the minimum size (or number of members) of a node in the decision tree to allow a further split. If the node has fewer data members than the minimum size for split, then it must become a leaf or end node in the tree. When individual trees are being trained, this is a criteria for stopping tree training.</p> <ul style="list-style-type: none"> <li>The range of possible values is any integer <math>\geq 2</math>.</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>Minimum Size for Split is referenced during the pre-pruning phase.</li> </ul> <p>Default value: <b>2</b>.</p>
<b>Minimum Leaf Size</b>	<p>Specifies the fewest number of data instances that can exist within a terminal leaf node of a decision tree. This property pre-prunes to constrain the tree leaf to at least this number of training samples.</p> <ul style="list-style-type: none"> <li>The range of possible values is any integer value <math>\geq 1</math>.</li> <li>This property limits the tree depth, based on the size of the leaf nodes, ensuring enough data makes it to each part of the tree.</li> </ul> <p>This setting is useful when the model construction is taking too long, or when the model shows very good ROC on training data but not nearly as good performance on hold-out or cross-validation data (due to over-fitting). For example, if the minimum leaf size is 2, then each terminal leaf node must contain at least 2 training data points.</p> <p>Default value: <b>1</b>.</p>
<b>Maximum JVM Heap Size</b>	<p>Determines the amount of virtual memory assigned to an individual tree trainer. The number of training samples for a single tree is limited by this.</p> <ul style="list-style-type: none"> <li>A value of -1 automatically sets the <b>Maximum JVM Heap Size</b> to avoid out-of-memory issues.</li> </ul> <p>Default value: 1024</p>
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li><b>Yes</b> specifies using the default Spark optimization settings.</li> <li><b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Outputs

### Visual Output

When run against Hadoop, the Alpine Forest Classification operator does not display the individual decision trees generated within the model because it is assumed that with large datasets the trees are much larger that can be displayed visually. Instead, summarizing statistics are presented.

- **Variable Importance** - Results for Hadoop data sources display the **Variable Importance** value, which provides a way of measuring the impact each variable has on the model.

At each split, we calculate how much this split reduces node impurity (purity gain). Then for each variable, we sum up over all splits where it is used (weighted by the number of samples used in the node), over all trees. We then find the variable that has the maximum purity gain and divide by this value across all variables.

For Alpine Forest Classification, we use Information Gain as the impurity function.

**Note:** The variable importance values are also stored as a CSV file in the following HDFS directory:

```
@default_tempdir/tsds_model/@user_name/@flow_name/AlpineForest_<uniqueFlowRunID>/varImp.csv
```

ALPINE FOREST (?)		
Results - Alpine Forest (Class. Hadoop)		
Variable Importance <a href="#">Individual Tree Statistics (Shows Max 20 Trees)</a> <a href="#">Average Tree Statistics</a>	Column Name	Variable Importance
	Column4	1
	Column3	0.73021
	Column6	0.728051
	Column1	0.725771
	Column2	0.592978
	Column5	0.252155
	Column7	0.096691

- **Individual Tree Statistics** - The Individual Tree Statistics (for up to 20 trees in the model) are displayed providing the number of training samples, dropped training samples, non-leaf nodes and number of leaves for each tree.

Results - Alpine Forest (Class. Hadoop)										
Statistics	Tree0	Tree1	Tree2	Tree3	Tree4	Tree5	Tree6	Tree7	Tree8	Tree9
Number of Training Samples	49862	50056	49646	50018	49912	50269	50304	49740	49906	50136
Number of Dropped Training Samples	0	0	0	0	0	0	0	0	0	0
Number of Non-Leaf Nodes	1863	2215	2005	1818	2106	2406	2319	2316	2003	2328
Number of Leaves	1864	2216	2006	1819	2107	2407	2320	2317	2004	2329

- **Average Tree Statistics** - Displays the average statistical values across all the individual trees in the model, which provides a sense of the overall size of the decision trees in the model.

Results - Alpine Forest (Class. Hadoop)															
<b>Variable Importance</b> <b>Individual Tree Statistics (Shows Max 20 Trees)</b> <b>Average Tree Statistics</b>	<table> <tr> <th>Statistics</th><th>Average over All Trees</th></tr> <tr> <td>Number of Trees</td><td>10</td></tr> <tr> <td>Number of Training Samples</td><td>49984.9</td></tr> <tr> <td>Number of Dropped Training Samples</td><td>0</td></tr> <tr> <td>Number of Non-Leaf Nodes</td><td>2137.9</td></tr> <tr> <td>Number of Leaves</td><td>2138.9</td></tr> <tr> <td>Impurity Function</td><td>InformationGain</td></tr> </table>	Statistics	Average over All Trees	Number of Trees	10	Number of Training Samples	49984.9	Number of Dropped Training Samples	0	Number of Non-Leaf Nodes	2137.9	Number of Leaves	2138.9	Impurity Function	InformationGain
Statistics	Average over All Trees														
Number of Trees	10														
Number of Training Samples	49984.9														
Number of Dropped Training Samples	0														
Number of Non-Leaf Nodes	2137.9														
Number of Leaves	2138.9														
Impurity Function	InformationGain														

## Output to Succeeding Operators

Connect this operator to succeeding operators.

## Alpine Forest Predictor - MADlib

Uses the model trained by Alpine Forest (MADlib) and scores the results. It must be connected to an Alpine Forest (MADlib) operator.



## Information at a Glance

Parameter	Description
Category	Model
Data source type	DB
Send output to other operators	Yes
Data processing tool	MADlib

The [Alpine Forest - MADlib](#) operator uses the MADlib built-in function, `forest_train()`, to generate multiple decision trees, the combination of which is used to make a prediction based on several independent columns. Each decision tree is generated based on bootstrapped sampling and a random subset of the feature columns. The destination of the output of this operator must be an Alpine Forest Predictor (MADlib) operator. MADlib 1.8 or higher must be installed on the database.

This operator must be connected to Alpine Forest (MADlib) and an operator that produces a database table output. For more information, see the [official MADlib documentation](#).



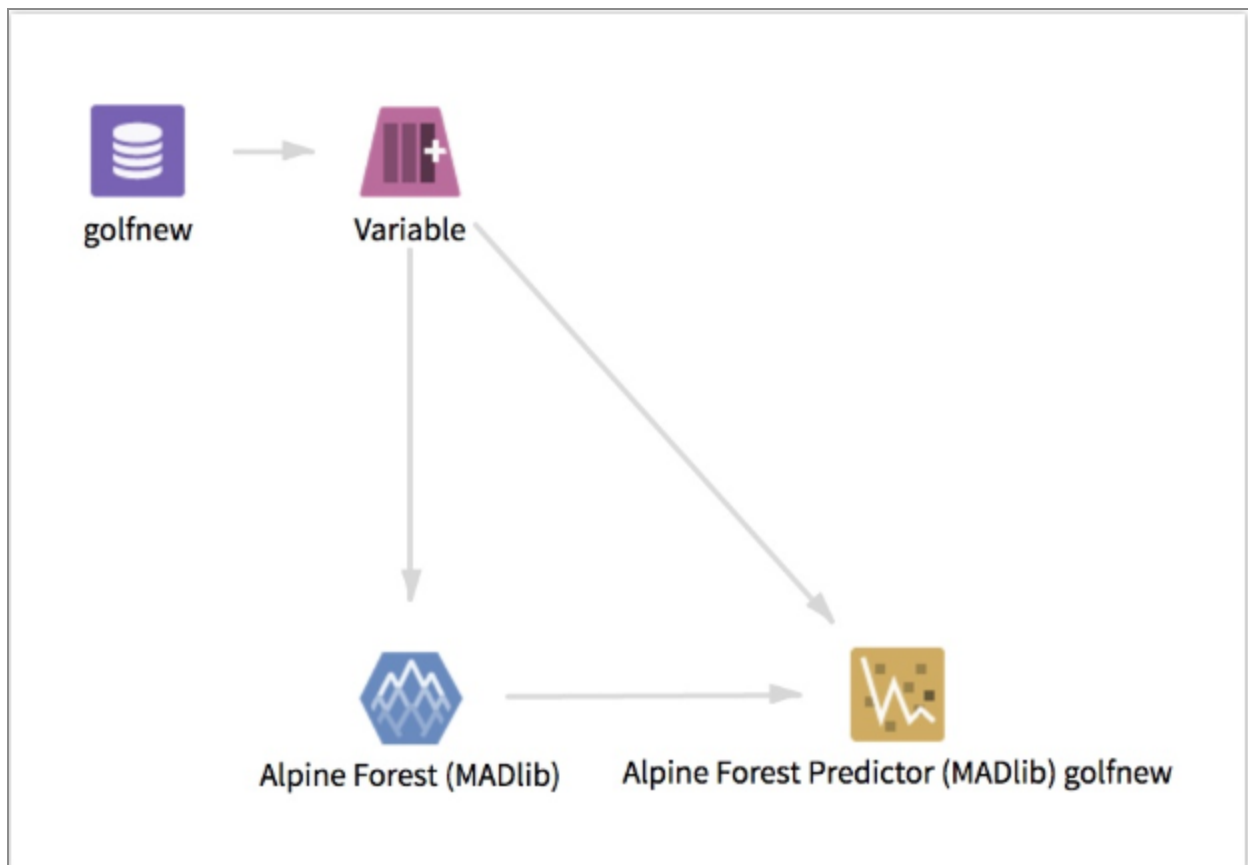
**Note:** This operator works only with MADlib 1.8 or higher.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Prediction Column</b>	The name of the prediction column.
<b>Prediction Data Type</b>	The data type of the prediction column.

Parameter	Description
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Drop If Exists</b>	Specifies whether to overwrite an existing table. <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Example



## Alpine Forest Regression

Applies an ensemble algorithm to make a numerical prediction by aggregating (majority vote or averaging) the numerical regression tree predictions of the ensemble.



### Information at a Glance

Parameter	Description
Category	Model
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce, Spark

### Input

A data set that contains the dependent and independent variables for modeling.

### Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	The quantity to model or predict. A <b>Dependent Column</b> must be specified for Alpine Forest. Select the data



Parameter	Description
	<p>column to be considered the Dependent Variable for the classification.</p> <p><b>Note:</b> For regression models, the Dependent variable must be numerical.</p>
<b>Columns</b>	<p>Allows you to select the Independent Variable data columns to include for the decision tree training.</p> <ul style="list-style-type: none"> <li>At least one column must be specified.</li> <li>Click <b>Columns</b> to open the dialog for selecting the available columns from the input data set for analysis.</li> </ul>
<b>Number of Trees</b>	<p>Specifies how many individual decision trees to train in the Alpine Forest Regression. Increasing the number of trees created generally increases the accuracy of the model. However, as long as enough trees are created, the Alpine Forest Regression model is not very sensitive to changing this parameter.</p> <p><b>Note:</b> The user interface only displays a maximum of 20 tree results, even if more are generated internally.</p> <p>Default value: <b>10</b>.</p>
<b>Use Automatic Configuration</b>	<p>Allows TIBCO Data Science - Team Studio to determine all the required Alpine Forest configuration parameters except for the <b>Number of Trees</b> parameter.</p> <p>Default value: <b>true</b>.</p>
<b>Number of Features Function</b>	<p>Automatically determines the <b>Number of Features per Node</b> parameter.</p> <p>Options:</p> <ul style="list-style-type: none"> <li>Square Root: The <b>Number of Features per Node</b> is set to the square root of the number of columns (truncated to an integer), or at least to 1.</li> <li>1/3: The <b>Number of Features per Node</b> is set to the (number of columns)/3 (truncated to an integer), or at least to 1.</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• All : The <b>Number of Features per Node</b> is set to the number of columns.</li> <li>• User Defined: The user directly sets the <b>Number of Features per Node</b> value, which is otherwise grayed out for other <b>Number of Features Function</b> choices (for Hadoop configuration).</li> </ul> <p>Default value: <b>Square Root</b>.</p>
<b>Number of Features per Node</b>	<p>Specifies <math>m</math>, the number of predictors to consider at each node during tree building process. The Alpine Forest algorithm calculates the best split for the tree based on these <math>m</math> variables that are selected randomly from the training set.</p> <p><b>Number of Features per Node</b> should be much less than the number of columns specified for the <b>Columns</b> property.</p> <div> <p><b>Note:</b> <b>Number of Features per Node</b> is the main configuration parameter to which an Alpine Forest model is most sensitive. Increasing the variable number per split makes each of the Decision Trees bigger, providing more information at each node. However, it also becomes harder to interpret for the modeler.</p> </div> <p>Default value: <b>1</b> (for Hadoop).</p>
<b>Sampling with Replacement</b>	<p>Specifies whether to use replacement when selecting training variable data row samples from the input data set. This property controls whether a data row can be reused for each of the <math>n</math> training data samples collected from the available data set rows.</p> <ul style="list-style-type: none"> <li>• Setting this value to <b>true</b> (the default) increases the training performance time because there are more possible random data set combinations.</li> <li>• Setting this value to <b>false</b> specifies that the system does not choose a data row more than once for each of the decision trees. This setting is appropriate for a small sample of <math>n</math> data rows from a large data set. In such a case, sampling without replacement is approximately the same as sampling with replacement (where the odds of randomly choosing the same data point twice is low).</li> </ul>

Parameter	Description
<b>Sampling Percentage (-1=Automatic)</b>	<p>Specifies the fraction of overall data rows available to select for the random sample data rows used for each decision tree.</p> <ul style="list-style-type: none"> <li>This value must be entered as a decimal.</li> <li>The Sampling Percentage is typically set low (10%-20%) since it is limited by how much data can fit in memory of individual reducers. For example, if a reducer has 2 GB of memory available but the entire data amounts to 10 GB, then most likely, individual reducers sample somewhat less than 20% of data (2/10). By contrast, the Spark version of Alpine Forest can perform 100% sampling on arbitrarily large data sets. The Sampling Percentage for Database is typically set to be 65-100% of the data.</li> <li>If Sampling Percentage is -1 (the default), TIBCO Data Science - Team Studio automatically determines the value and makes sure the sampling percentage is not too large to fit in memory.</li> </ul> <p><b>Caution:</b> If <b>Sampling Percentage</b> is set too large for Hadoop, the number of samples might be larger than what an individual tree trainer can fit in memory (which is determined in Hadoop by <b>Max JVM Heap Size</b>). In this case, TIBCO Data Science - Team Studio drops random samples so that eventually all training samples can fit in memory.</p>
<b>Max Depth (-1=Unlimited)</b>	<p>Sets the "depth" of the tree or the maximum number of decision nodes it can branch out to beneath the root node during the tree-growth phase. A tree stops growing any deeper if either a node becomes empty (that is, there are no more examples to split in the current node) or the depth of the tree exceeds this <b>Max Depth</b> limit.</p> <ul style="list-style-type: none"> <li>The range of possible values is between -1 and any integer greater than 0.</li> <li>A value of -1 represents "no bound" - the tree can take on any size or an unlimited number of decision nodes until the nodes become empty.</li> </ul> <p>Default value: <b>5</b>.</p>
<b>Min Size For</b>	(Pre-pruning parameter)

Parameter	Description
<b>Split</b>	<p>Specifies the minimal size (or number of members) of a node in the decision tree in order to allow a further split. If the node has fewer data members than the <b>Minimal Size for Split</b>, it must become a leaf or end node in the tree. When individual trees are being trained, this is a criteria for stopping tree training.</p> <p><b>Minimal Size for Split</b> is referenced during the pre-pruning phase.</p> <ul style="list-style-type: none"> <li>The range of possible values is any integer <math>\geq 2</math>.</li> </ul> <p>Default value: <b>2</b>.</p>
<b>Min Leaf Size</b>	<p>(Pre-pruning parameter)</p> <p>Limits the tree depth based on the size of the leaf nodes, ensuring enough data makes it to each part of the tree.</p> <p>This is useful when the model construction is taking too long or when the model shows very good ROC on training data but not nearly as good performance on hold-out or cross-validation data (due to over-fitting). For example, if the <b>Min Leaf Size</b> is <b>2</b>, each terminal leaf node must contain at least 2 training data points.</p> <p>The range of possible values is any integer value <math>\geq 1</math>.</p> <p>Default value: <b>1</b>.</p>
<b>Max JVM Heap Size (MB) (-1=Automatic)</b>	<p>The <b>Max JVM Heap Size</b> (for Hadoop only) determines the amount of virtual memory assigned to an individual tree trainer. The number of training samples for a single tree is limited by this.</p> <p>Default value: <b>1024</b>.</p> <p>A value of <b>-1</b> automatically sets the <b>Max JVM Heap Size</b> to avoid out-of-memory issues.</p>
<b>Use Spark</b>	<p>If <b>Yes</b> (the default), uses Spark to optimize calculation time.</p>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li><b>Yes</b> specifies using the default Spark optimization settings.</li> <li><b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

- **Variable Importance** - The results provide the Regression Coefficient values for each independent variable in the model.

At each split, we calculate how much this split reduces node impurity (purity gain). Then for each variable, we sum up over all splits where it is used (weighted by the number of samples used in the node), over all trees. We then find the variable that has the maximum purity gain and divide by this value across all variables.

For Alpine Forest Regression, we use Variance Reduction as the impurity function.

Results - Alpine Forest Regression - Hadoop		
<b>Variable Importance</b> <b>Individual Tree Statistics</b> (Shows Max 20 Trees) <b>Average Tree Statistics</b>	Column Name	Variable Importance
	Column8	1
	Column2	0.608792
	Column6	0.596574
	Column5	0.567612
	Column4	0.55131
	Column7	0.452899
	Column3	0.449208
	Column1	0.083688

- **Individual Tree Statistics** - Shows the results for each Decision Tree in the model, up to a maximum of 20 trees.

Results - Alpine Forest Regression - Hadoop										
<b>Variable Importance</b> <b>Individual Tree Statistics</b> (Shows Max 20 Trees) <b>Average Tree Statistics</b>	Statistics	Tree0	Tree1	Tree2	Tree3	Tree4	Tree5	Tree6	Tree7	Tree8
	Number of Training Samples	4098	4185	4105	4213	4225	4213	4166	4171	4154
	Number of Dropped Training Samples	0	0	0	0	0	0	0	0	0
	Number of Non-Leaf Nodes	1952	1964	1897	1993	1920	2056	1945	1970	1949
	Number of Leaves	2012	1995	1949	1996	1950	2060	1962	1986	1979

- **Average Tree Statistics** - Provides a snapshot summary of each tree used in the model.
  - Overall number of trees in the model
  - Average number of training samples used
  - Average number of dropped training samples
  - Average number of non-leaf nodes
  - Average number of leaves
  - Impurity Function used for the model

Results - Alpine Forest Regression - Hadoop		
Variable Importance Individual Tree Statistics (Shows Max 20 Trees) Average Tree Statistics	Statistics	Average over All Trees
	Number of Trees	10
	Number of Training Samples	4164.1
	Number of Dropped Training Samples	0
	Number of Non-Leaf Nodes	1957.4
	Number of Leaves	1982
	Impurity Function	Variance

## Data Output

Typically, an Alpine Forest Regression model is followed by a Predictor operator which provides the prediction value for each data row compared against the actual data set training value and the associated confidence level.

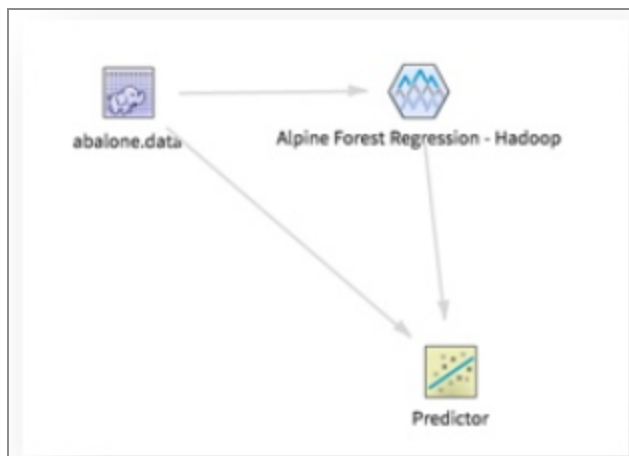
**i Note:** Currently, the Alpine Forest Regression operator does not have a specific Evaluator operator for it. Use the Predictor operator to compare predicted versus actual values and generally assess the accuracy of the Alpine Forest Regression operator .

The following illustration shows output from the Predictor operator for the Alpine Forest Regression operator.

Results - Predictor									
Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	P_Column9
I	0.57	0.425	0.14	0.7655	0.331	0.14	0.24	10	9.8
M	0.58	0.45	0.14	0.824	0.3465	0.1765	0.263	10	10.8
I	0.58	0.425	0.145	0.83	0.379	0.1605	0.2575	11	10.9
I	0.585	0.47	0.17	0.985	0.3695	0.2395	0.315	10	10.1
M	0.585	0.45	0.15	0.997	0.4055	0.283	0.251	11	10.3
F	0.595	0.455	0.14	0.914	0.3895	0.2225	0.271	9	9
F	0.6	0.5	0.17	1.13	0.4405	0.267	0.335	11	11.4
F	0.615	0.495	0.155	1.0805	0.52	0.19	0.32	9	9.7
M	0.63	0.505	0.155	1.105	0.492	0.226	0.325	11	10.4
M	0.63	0.49	0.155	1.229	0.535	0.29	0.335	11	10.6
F	0.635	0.495	0.175	1.2355	0.5205	0.3085	0.347	10	10
F	0.645	0.535	0.19	1.2395	0.468	0.2385	0.424	10	12.1
F	0.65	0.505	0.165	1.357	0.5725	0.281	0.43	11	11
M	0.655	0.525	0.18	1.402	0.624	0.2935	0.365	13	12.4

The **P\_ column** can be compared to the actual values of the dependent column (in this case Column9) in order to assess the accuracy of the model.

## Example



## ARIMA Time Series (DB)

Applies the ARIMA algorithm to an input time series data set and generates step forecasts for simulation or predictive modeling needs.



## Information at a Glance

Parameter	Description
Category	Model
Data source type	DB
Send output to other operators	Yes
Data processing tool	MADlib

**i Note:** The ARIMA Time Series (DB) operator is for database data only. For Hadoop data, use the [ARIMA Time Series \(HD\)](#) operator.

A previous version of this operator was deprecated and removed in version 6.1. To use this new operator, you must remove the old Time Series operator from your workflow and replace it with the new ARIMA Time Series operator.

- Users must specify a column by which to order the time series data.
- The time series column should be evenly spaced, or else the resulting output is inconsistent.
- Users can specify a column to group the time series data by, and the operator applies the algorithm separately to the time series filtered by group.

Example use case applications of this operator include predicting future retail sales, modeling the evolution of financial market prices, forecasting weather trends, and predicting IT server loads.



## Algorithm

The ARIMA (AutoRegressive, Integrated, Moving Average) class of time series model is a generalization of the ARMA (AutoRegressive, Moving Average) models.

To understand an ARIMA model, it is necessary to first understand the ARMA model.

An ARMA model of order  $(p,q)$  for a time series  $\{X_t\}$  can be written as

$$X_t = a_1 X_{t-1} + \dots + a_p X_{t-p} + \epsilon_t + b_1 \epsilon_{t-1} + \dots + b_q \epsilon_{t-q}$$

where  $p$  is the order of the autoregressive component,  $q$  is the order of the moving average, and the  $\epsilon_t$  are the error (white noise) terms.

To understand the ARIMA model with integrated part, it is helpful to use the lag operator.

This lag, or backstep, operator  $L$  acts on a term in a time series by taking it back one time step:

$$LX_t = X_{t-1}, L^2 X_t = X_{t-2}, \dots, L^s X_t = X_{t-s}.$$

Then the above ARMA( $p,q$ ) model can be written as

$$\left(1 - \sum_{i=1}^p a_i L^i\right) X_t = \left(1 + \sum_{i=1}^q b_i L^i\right) \epsilon_t.$$

We can then introduce an integrated part of order  $d$  using a unit root of order  $d$ :

$$(1 - L)^d.$$

Thus, the full ARIMA( $p,d,q$ ) model is given by:

$$\left(1 - \sum_{i=1}^p a_i L^i\right) (1 - L)^d X_t = \left(1 + \sum_{i=1}^q b_i L^i\right) \epsilon_t.$$

## Input

A tabular data set from the preceding operator that contains a column of time series data and a column by which to order the time series data.

## Bad or Missing Values

If a row contains a null value in at least one of the **Time Series**, **Column to Order By**, or **Grouping Column**, the row is removed from the data set. The number of null values removed can be listed in the **Summary** section of the output (depending on the chosen option for **Write Rows Removed Due to Null Data To File**).

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>MADlib Schema</b>	The schema where MADlib is installed in the database. MADlib must be installed in the same database as the input data set. If a "madlib" schema exists in the database, this parameter defaults to <code>madlib</code> .
<b>Time Stamp</b>	Choose the column that contains the timestamp data for the ARIMA model. This can be a numeric or datetime type.
<b>Time Series</b>	Choose the column to use as the time series. This can be a numeric column with the datatype double. If needed, a preceding Variable operator can be used to convert other numeric types to double.
<b>Grouping Columns</b>	Choose a list of column names used to group the input data set into discrete groups, training one ARIMA model per group. It is similar to the SQL GROUP BY clause. When this value is null, no grouping is used and a single result model is generated.
<b>Include Mean</b>	The mean value of the data series is added in the ARIMA model if this variable is True. Default value: <b>true</b> .
<b>Steps Ahead</b>	Choose the number of steps ahead the ARIMA time series runs. Default value: <b>20</b> .
<b>Autoregressive</b>	The AR parameter $\phi(B)$ . Default value: <b>1</b> .
<b>Integrated</b>	The integrated parameter. Default value: <b>1</b> .

Parameter	Description
<b>Moving Average</b>	The MA parameter $\theta(B)$ . Default value: <b>1</b> .
<b>Max Iterations</b>	The maximum number of iterations to run learning algorithm. Default value: <b>100</b> .
<b>Optimizer tau</b>	Computes the initial step size for gradient algorithm. Default value: <b>0.001</b> .
<b>Optimizer e1</b>	The algorithm-specific threshold for convergence. Default value: <b>1e-15</b> .
<b>Optimizer e2</b>	The algorithm-specific threshold for convergence. Default value: <b>1e-15</b> .
<b>Optimizer e3</b>	The algorithm-specific threshold for convergence. Default value: <b>1e-15</b> .
<b>Optimizer Hessian Delta</b>	The delta parameter to compute a numerical approximation of the Hessian matrix. Default value: <b>1e-6</b> .
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	<p>Advanced database settings for the operator output. Available only for <b>TABLE</b> output.</p> <p>See <a href="#">Storage Parameters dialog</a> for more information.</p>
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

The output consists of four tabbed sections: **Steps Ahead**, **Model**, **Summary**, and **Residual**.

- The **Steps Ahead** tab tabulates ARIMA forecasts for the given input data set and configuration.

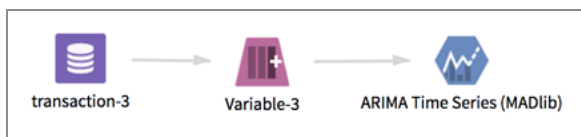
Results - ARIMA (MADlib)		
<a href="#">Steps Ahead</a> <a href="#">Model</a> <a href="#">Summary</a> <a href="#">Residual</a>	Operation created output with 2 columns and 20 rows.	
	steps_ahead	forecast_value
	1	3,471.7256
	3	3,472.5157

- The **Model** tab tabulates, for each model, the fitted parameters for the AR, MA, and intercept terms, along with other metrics describing the trained time series model.
- The **Summary** tab displays the parameters selected, a report on null data removal, and the steps ahead and model metrics data set locations in HDFS.
- The **Residual** tab shows the tabular data used to train the model.

## Data Output

The **Steps Ahead** output can be consumed by any operator that processes tabular data sets.

## Example



## ARIMA Time Series (HD)

Applies the ARIMA algorithm to an input time series data set and generates step forecasts for simulation or predictive modeling needs.



## Information at a Glance

Parameter	Description
Category	Model
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

**i Note:** The ARIMA Time Series (HD) operator is for Hadoop data only. For database data, use the [ARIMA Time Series \(DB\)](#) operator.

A previous version of this operator was deprecated and removed in version 6.1. To use this new operator, you must remove the old Time Series operator from your workflow and replace it with the new ARIMA Time Series operator.

- Users must specify a column by which to order the time series data.
- The time series column should be evenly spaced, or else the resulting output is inconsistent.
- Users can specify a column to group the time series data by, and the operator applies the algorithm separately to the time series filtered by group.

Example use case applications of this operator include predicting future retail sales, modeling the evolution of financial market prices, forecasting weather trends, and predicting IT server loads.

## Algorithm

The ARIMA (AutoRegressive, Integrated, Moving Average) class of time series model is a generalization of the ARMA (AutoRegressive, Moving Average) models.

To understand an ARIMA model, it is necessary to first understand the ARMA model.

An ARMA model of order  $(p,q)$  for a time series  $\{X_t\}$  can be written as

$$X_t = a_t X_{t-1} + \dots + a_p X_{t-p} + \epsilon_t + b_1 \epsilon_{t-1} + \dots + b_q \epsilon_{t-q}$$

where  $p$  is the order of the autoregressive component,  $q$  is the order of the moving average, and the  $\epsilon_t$  are the error (white noise) terms.

To understand the ARIMA model with integrated part, it is helpful to use the lag operator.

This lag, or backstep, operator  $L$  acts on a term in a time series by taking it back one time step:

$$LX_t = X_{t-1}, L^2 X_t = X_{t-2}, \dots, L^s X_t = X_{t-s}.$$

Then the above ARMA( $p,q$ ) model can be written as

$$\left(1 - \sum_{i=1}^p a_i L^i\right) X_t = \left(1 + \sum_{i=1}^q b_i L^i\right) \epsilon_t.$$

We can then introduce an integrated part of order  $d$  using a unit root of order  $d$ :

$$(1 - L)^d.$$

Thus, the full ARIMA( $p,d,q$ ) model is given by:

$$\left(1 - \sum_{i=1}^p a_i L^i\right) (1 - L)^d X_t = \left(1 + \sum_{i=1}^q b_i L^i\right) \epsilon_t.$$

## Input

A tabular data set from the preceding operator that contains a column of time series data and a column by which to order the time series data.

## Bad or Missing Values

If a row contains a null value in at least one of the **Time Series**, **Column to Order By**, or **Grouping Column**, the row is removed from the data set. The number of null values removed can be listed in the **Summary** section of the output (depending on the chosen option for **Write Rows Removed Due to Null Data To File**).

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Column To Order By</b>	Define the column by which to order the time series data. Ordering is required to ensure the time series is correctly processed in sequence.  Supported data types: Int, Long, DateTime.
<b>Time Series</b>	Define the column that contains the time series data.  Supported data types: Int, Long, Float, Double.
<b>Grouping Column</b>	(Optional) Define the column to use to split the time series data into groups or categories. This is useful when the input data set contains data sampled at the same time, but for multiple groups.  All data types are supported.
<b>Include Intercept</b>	Specify whether the ARIMA model should be fitted with an intercept.  Default value: <b>true</b> .
<b>Auto-regressive (p)</b>	Define the AR order, that is, the degree to which the time series data is to be lagged and regressed on itself.  Range: [0, Int.Max] although it is recommended to keep $p < 5$ .
<b>Integrated (d)</b>	Define the degree of differencing, that is, the number of times the time series data is replaced with the difference between the value of a time step and that of the previous step. This parameter is used to account for time

Parameter	Description
	<p>series data that is non-stationary in nature.</p> <p>Range:[0, Int.Max] although it is recommended to keep <math>d &lt; 5</math>.</p>
<b>Moving Average (q)</b>	<p>Define the MA order, that is, the degree by which the regression error of a time step is a linear combination of errors from previous time steps.</p> <p>Range:[0, Int.Max] although it is recommended to keep <math>q &lt; 5</math>.</p>
<b>Steps Ahead</b>	Define the number of time steps to forecast using the fitted ARIMA model.
<b>Write Rows Removed Due to Null Data To File</b>	<p>Rows with null values in at least one of the independent columns or the dependent column are removed from the analysis. This parameter allows you to specify that the data with null values are written to a file.</p> <p>The file is written to: @default_tempdir/tsds_out/@user_name/@flow_name/@operator_name_uuid/bad_data</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> (the default) - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>



Parameter	Description
	Available Avro compression options. <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	Specifies whether to delete existing data at that path. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Outputs

### Visual Output

The output consists of three tabbed sections: **Steps Ahead**, **Model**, and **Summary**.

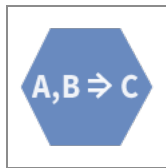
- The **Steps Ahead** tab tabulates ARIMA forecasts for the given input data set and configuration.
- The **Model** tab tabulates, for each model, the fitted parameters for the AR, MA, and intercept terms, along with other metrics describing the trained time series model.
- The **Summary** tab displays the parameters selected, a report on null data removal, and the steps ahead and model metrics data set locations in HDFS.

## Data Output

The **Steps Ahead** output can be consumed by any operator that processes tabular data sets.

## Association Rules

Association Rules modeling refers to the process of determining patterns that occur frequently within a data set, such as identifying frequent combinations or sets of items bunched together, subsequences or substructures within the data.



## Information at a Glance

Parameter	Description
Category	Model
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

For more information about Association Rules, see [Patterns in Data Sets](#)

## Input

An HDFS tabular data set with a sparse item set (or "basket") column (key/value dictionary with distinct items as keys and frequency as values).

The input is most likely to be generated by the [Collapse](#) operator, using the transaction **ID** column(s) in a **Group By** clause and the **item** column in the **Columns to Collapse** checkboxes. See the example for an illustration.

## Bad or Missing Values

If the **Item Set Column** selected column has null values in the **Item Sets Column** selected, the rows are removed from the data set. The number of null values removed can be listed in the Summary section of the output (depending on the chosen option for **Write Rows Removed Due to Null Data To File**). If the **Item Set Column** selected has bad key/values pairs in the dictionary, the operator fails at runtime with a meaningful error message.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Item Set Column</b>	<p>Select the dictionary column that represents the item set (or "basket") for each transaction (<b>ID, group...</b>).</p> <p>Data type supported: TIBCO Data Science - Team Studio Sparse</p> <p>This column is most likely to be the output column of the Collapse operator (see the Input section, above).</p>
<b>Maximum Size of Frequent Item Sets</b>	Specify the maximum number of items a frequent item set can contain in order to be used to generate association rules. The range of possible values is between 1 and 50. Default value: <b>5</b> .
<b>Minimum Support</b>	<p>Support of an association rule is the percentage of groups that contain all of the items listed in the Association Rule. <b>Minimum Support</b> specifies the lower threshold percentage of groups or transactions that contain all of the items listed for the association rule.</p> <ul style="list-style-type: none"> <li>• Therefore, an Association Rule is considered valid only when the <b>Support</b> value equals or exceeds this <b>Minimum Support</b> value.</li> <li>• The range of possible values is any decimal value between 0.001 and 1. The default value is <b>0.3</b>, or at least 30% of the transactions that contain the item set.</li> </ul>

Parameter	Description
<b>Minimum Rule Confidence</b>	<p>The <b>Confidence</b> of an Association Rule is a percentage value that shows how frequently the rule head occurs among all groups that contain the rule body. The <b>Confidence</b> value indicates how reliable this rule is. The higher the value, the more often this set of items is associated together. <b>Minimum Confidence</b> specifies the lower percentage threshold for the required frequency of the rule occurring.</p> <p>For a rule <math>X \Rightarrow Y</math> :</p> $\text{conf}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)} \quad **$ <ul style="list-style-type: none"> <li>Therefore, the <b>Confidence</b> measurement indicates how reliable a rule is. The higher the value, the more often this set of items is associated together. For example, if the <b>Confidence</b> is 85% for the list {meat, potatoes}, it means that 85% of the times that meat &amp; potatoes (premise) was purchased, so were potatoes (rule body).</li> <li>The range of possible values is any decimal value between 0 and 1. The default value is <b>0.7</b>, which means that the items must be associated together 70% of the time.</li> </ul>
<b>Minimum Rule Lift</b>	<p>The <b>Lift</b> of an Association Rule is the ratio of the observed support for the rule ( = support (XUY) ) to that expected if X and Y were independent.</p> $\text{lift}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X) \times \text{supp}(Y)} \quad **$ <p>If a rule has a lift of 1, it implies that the probability of occurrence of the antecedent X and that of the consequent Y are independent of each other. When 2 events are independent of each other, no rule should be drawn involving those two events.</p> <p>If the lift is &gt; 1, that lets us know the degree to which those two occurrences are dependent on one another, and makes those rules potentially useful for predicting the consequent in future data sets.</p> <p>The value of lift is that it considers both the confidence of the rule and the</p>

Parameter	Description
	<p>overall data set.</p> <ul style="list-style-type: none"> <li>• Default value is 1.5</li> <li>• Range of possible values is any decimal value between 0 and 10E5.</li> </ul> <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p><b>Note:</b> Valid rules - To be considered a valid association rule, a rule must meet all the following requirements:</p> <ul style="list-style-type: none"> <li>◦ Minimum support for the item set</li> <li>◦ Minimum rule confidence</li> <li>◦ Minimum rule lift</li> </ul> </div> <p>Note that <math>\text{supp}(X \cup Y)</math> means the support of the union of the items in <math>X</math> and <math>Y</math>. This is somewhat confusing, since we normally think in terms of probabilities of events and not sets of items. We can rewrite <math>\text{supp}(X \cup Y)</math> as the joint probability <math>P(E_X \cap E_Y)</math> where <math>E_X</math> and <math>E_Y</math> are the events that a transaction contains itemset <math>X</math> and <math>Y</math>, respectively.</p>
<b>Write Rows Removed Due to Null Data To File</b>	<p>Rows with null values in <b>Item Sets Column</b> are removed from the analysis. This parameter allows you to specify that the data with null values be written to a file.</p> <p>The file is written to the same directory as the rest of the output. The filename is bad_data.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data, but do not count and display in the result UI</li> <li>• <b>Do Not Write Null Rows</b> - remove null value data, but do not write to an external file.</li> <li>• <b>Write Up to 1000 Null Rows to File</b> - remove null value data and write the first 1000 rows of that data to the external file.</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Outputs

### Visual Outputs

- Association Rules **Output** (the main output transmitted to subsequent operators). An example is shown here.

Results - Association Rules 1						
rule_antecedent	rule_consequent	support	confidence	lift	conviction	
(HOT WATER BOTTLE TEA AND SYMPHY)	(HOT WATER BOTTLE KEEP CALM)	0.01319	0.4659	11.6433	1.7973	
(CHOCOLATE HOT WATER BOTTLE)	(HOT WATER BOTTLE KEEP CALM)	0.01415	0.3651	9.125	1.512	
(HOT WATER BOTTLE I AM SO POORELY)	(HOT WATER BOTTLE KEEP CALM)	0.01063	0.3793	9.4801	1.5466	
(LOVE HOT WATER BOTTLE)	(HOT WATER BOTTLE KEEP CALM)	0.01245	0.4966	12.4114	1.907	
(SCOTTIE DOG HOT WATER BOTTLE)	(HOT WATER BOTTLE KEEP CALM)	0.0104	0.2824	7.05818	1.3378	

It displays the rules selected from the user's specified criteria, with the related measures of significance of these rules:

- **Support** (see Parameter section - **Minimum Support**)
- **Confidence**: (see Parameter section - **Minimum Rule Confidence**)
- **Lift**: (see Parameter section - **Minimum Rule Lift**)
- **Conviction**: The conviction of a rule can be interpreted as the ratio of the expected frequency that  $X$  occurs without  $Y$  (that is, the frequency that the rule makes an incorrect prediction) if  $X$  and  $Y$  were independent, divided by the observed frequency of incorrect predictions. For example, if a rule ( $X \Rightarrow Y$ ) has a conviction of 1.2, it means that this rule would be incorrect 20% more often (1.2 times as often) if the association between  $X$  and  $Y$  was purely random chance. The higher conviction is, the more reliable the rule is.

$$\text{conv}(X \Rightarrow Y) = \frac{1 - \text{supp}(Y)}{1 - \text{conf}(X \Rightarrow Y)} .$$

**i Note:** Null Conviction - If the observed frequency of incorrect convictions is 0 (that is, confidence is 1), the rule conviction cannot be calculated and a null value is displayed.

- ◦ **Frequent Item Sets data set** (stored in HDFS):

Results - Association Rules-1			
Output	Item_set	frequency	support
Frequent Item Sets Dataset Summary	{WOODEN ROUNDERS GARDEN SET }	909	0.01722
	{PACK OF 12 SUKI TISSUES }	765	0.01449
	{SET OF 60 I LOVE LONDON CAKE CASES }	879	0.01665
	{LUNCH BAG BLACK SKULL,, LUNCH BAG CARS BLUE, LUNCH BAG WOODLAND}	603	0.01142
	{JUMBO BAG PINK POLKADOT, JUMBO STORAGE BAG SUKI}	861	0.01631

- Summary of the parameters selected, rows removed due to null values, rules generated, and output location.

Output Frequent Item Sets Dataset Summary	Parameters Selected	
	Item Sets Column:	itemSet
	Maximum Size of Frequent Item Sets	5
	Minimum Support:	0.010
	Minimum Rule Confidence :	0.700
	Output	
	Number of Frequent Item Sets:	1128
	Number of Association Rules:	80
	Input data size:	52785 rows
	Input size after removing rows due to null values:	52785 rows
No data removed due to null values		
The Output with association rules is stored at:		
/tmp/alpine_out/emilie/6.3_Test_Association_Rules/Association_Rules_1481861155176		
The Frequent Items Dataset is stored at:		
/tmp/alpine_out/emilie/6.3_Test_Association_Rules/Association_Rules_1481861155176_frequent_item_sets		

## Data Output

The output of this operator is the Association Rules data set that can be further filtered out based on rule metrics (for example, using a Row Filter).

**Note:** The Frequent Items data set is also stored in HDFS and can be dragged on the canvas for further analysis.

## Example

id	item
1	vanilla
1	banana
2	vanilla
2	strawberry
2	chocolate

id	itemSet
2	{"chocolate":1,"strawberry":1,"vanilla":1}
1	{"banana":1,"vanilla":1}

## Additional Notes

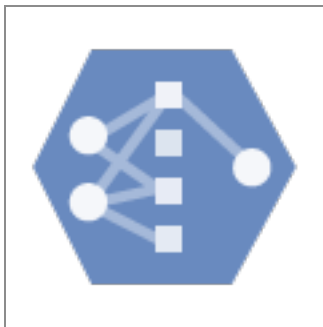
**FP-Growth Performance** - The FP-Growth algorithm can result in slow processing when the minimum support chosen is very low (close to 0). If you detect slowness, try one of the following remedies.



- Increase the value of **Minimum Support**.
- Increase Spark executor memory and/or Number of Partitions in **Advanced Spark Settings**.

## Collaborative Filter Trainer

Collaborative filtering is used commonly for recommender systems. Given input data for users, products, and ratings, the Collaborative Filtering Trainer uses an alternating least squares (ALS) method, in which users and products are described by a small set of latent factors that can be used to predict unknown or empty entries in the sparse matrix.



### Information at a Glance

Parameter	Description
Category	Model
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

### Input

An HDFS tabular data set with at least three columns. There should be one column each to represent people, products, and ratings.

## Bad or Missing Values

Null values are removed from the data set. The number of null values removed is listed in the Summary section of the output.

## Restrictions

Using a high value for **Additional Parameters to Try** impacts performance because the ALS algorithm must be run many times. See the parameters list for details.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Users Column</b>	Select the column that has information about the users in your data set.
<b>Products Column</b>	Select the column that has information about the products in your data set.
<b>Ratings Column</b>	Select the column that has information about the ratings in your data set.
<b>Number of Latent Factors (Rank)</b>	The parameter for the MLLib model. This represents the number of latent factors to train the model on. Must be between 3 and 100.  Range: 3-100 (inclusive). Default value: <b>5</b> .
<b>Number of Iterations</b>	The parameter that is passed to the MLLib ALS model. It represents the number of iterations to be used for each run of ALS.  If in the <b>Vary Parameters and Pick Model with Lowest Error</b> parameter you select to train more than one model, the value of this parameter is the first one selected and the next values are chosen randomly using this number as a seed.  Range: 1-20 (inclusive). Default value: <b>10</b> .

Parameter	Description
<b>Regularization Parameter (<math>\lambda</math>) per Model</b>	<p>The parameter that is passed to the MLLib model to use as a scalar when training ALS to prevent overfitting the data.</p> <p>Range: 0.0-1.0 (exclusive). Default value: <b>0.01</b>.</p>
<b>Vary Parameters and Pick Model with Lowest Error (slower)</b>	<ul style="list-style-type: none"> <li>• <b>no</b> (the default) - Run the MLLib ALS Algorithm one time with the values of Rank, Number of Iterators, and <math>\lambda</math>, chosen above. (We use <math>\alpha=0.01</math>).</li> <li>• <b>yes</b> - Run multiple versions of the MLLib ALS model and pick the one with the best root mean squared error. More specifically, we train the algorithm as follows. <ol style="list-style-type: none"> <li>1. Split the training data into training, test, and validation set (60% Training, 20% Test and 20% Validation).</li> <li>2. Run the MLLib ALS Algorithm on the training set: first with the values of Rank, Number of Iterators, and <math>\lambda</math> shown above.</li> <li>3. Run the algorithm several more times using different, semi-random values of those parameters.</li> <li>4. For each run, calculate the root mean squared error (RMSE) on the test set against the validation set.</li> <li>5. For the training parameters that yield the lowest RMSE, retrain the model on all of the input data and return it.</li> </ol> </li> </ul> <p>In the output, we show the parameters used to train each pass of the ALS Algorithm and the Root Mean Squared Error associated with each.</p>
<b>Number of Additional Parameters to Try</b>	<p>Number of random values to try for each of the model parameters (Rank, Number of Iterators, Lambda). We run the MLLib ALS algorithm for each combination of the original parameter values and the randomly chosen one.</p>

Parameter	Description
	<p><b>Note:</b> The higher this value is, the more expensive the computation. This can take a very long time for higher values. For example, if you entered 5, for each of the three parameters (Rank, Number of Iterators, Lambda), we look at six values (the value you input + five randomly chosen ones). We run the ALS algorithm in this case <math>3^6</math>, or 243, times. For each of those runs, we also compute the RMSE. This requires a shuffle of the data on the Spark side, further impacting performance. Choose a value for this parameter wisely.</p> <p>Range: 0-20 (inclusive). Default value: <b>0</b>.</p>
<b>Output Directory</b>	The location to store the output files.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Outputs

### Visual Output

A table that shows the parameters chosen and the Root Mean Squared Error. If there are multiple models trained during this process (if you selected **Vary Parameters and Pick Model with Lowest Error**), the other models are shown here.

Model Number	Rank	Number of Iterators	Lambda	RMSE
0	5	10	0.01	0.78988088

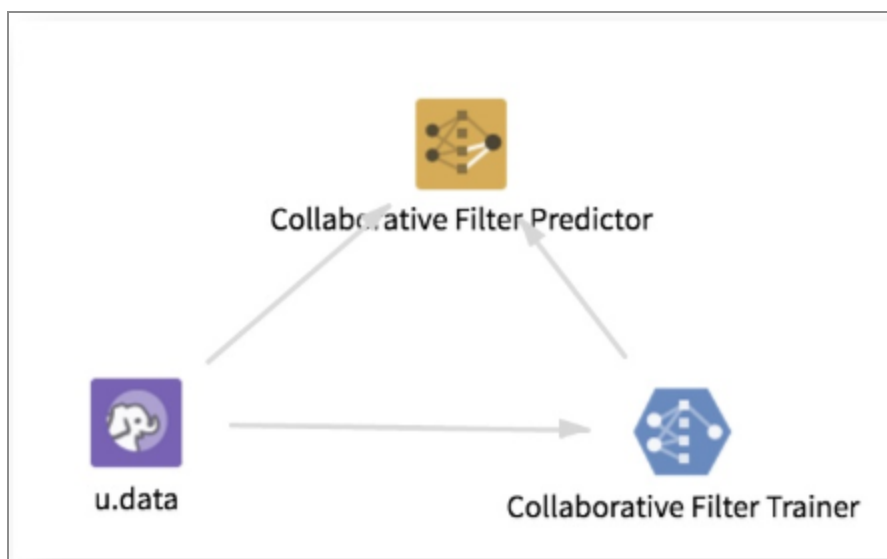
A Summary that describes the output location of the results and the number of rows processed, as well as the number of distinct users and products.

1 Models Trained.	
Operator Type	Collaborative Filter Trainer
Operator Name	Collaborative Filter Trainer
Input data size:	100000 rows
Input size after removing rows with null values :	100000 rows
No data removed due to null values	
<b>Input Parameters</b>	
Users Column	User
Products Column	Product
<b>Training Data statistics</b>	
Distinct Users	943
Distinct Products	1682
<b>Best Model</b>	
RMSE	0.7898808822358485
Number of Latent Factors (Rank)	5
Number of Iterations	10
Regularization Parameter ( $\lambda$ )	0.01
<b>Output</b>	
The latent factors for the Userids (User) are stored at:	
/tmp/alpine_out/allison/CustomOperator/userFactors	
The latent factors for the Productids (Product) are stored at:	
/tmp/alpine_out/allison/CustomOperator/productFactors	

## Data Output

The latent factors for the user IDs and for the product IDs are stored on HDFS and sent to the next operator. Use this operator along with a Collaborative Filter Recommender or Predictor to create recommendations.

## Example



## Decision Tree

Applies a classification modeling algorithm to a set of input data. The Decision Tree operator has three configuration phases: tree growth, pre-pruning, and pruning.



### Information at a Glance

Parameter	Description
Category	Model
Data source type	HD
Send output to other operators	No
Data processing tool	MapReduce

This topic applies to configuring a decision tree for Hadoop. To see information for database decision tree options, see [Decision Tree - MADlib](#), [Decision Tree Classification - CART](#), or [Decision Tree Regression - CART](#). For more information about working with decision trees, see [Classification Modeling with Decision Tree](#).

### Algorithm

The Decision Tree operator implements a supervised classification algorithm. After the decision steps of the tree are learned, the Decision Tree algorithm is quick to evaluate the predicted classifications for new data.

The Decision Tree Operator implements a classification tree only, meaning its node labels represent a range of discrete values, as opposed to the CART Operator, which is both a classification and regression tree, with its labels representing a range of continuous (numeric) values.

**i Note:** The Decision Tree Operator can split a node into more than just two sub-groups (that is, it is not restricted to being a binary tree like the CART Operator).

The Decision Tree Operator supports the C4.5 (Quinlan, 1993) deterministic method for constructing the decision tree structure, using information gain as the criteria. Therefore, the following is true.

- At each node of the tree, C4.5 chooses one attribute of the data that most effectively splits its set of samples into subsets with one attribute value or another.
- The attribute with the highest normalized information gain is chosen to make the decision and is placed higher in the tree compared to other attributes.
- Information Gain measures the gain in purity from parent to children sub-nodes.
- The C4.5 algorithm then recurses on the smaller sub-lists.
- For pruning, the C4.5 algorithm employs a pessimistic pruning method that estimates error rates when making decisions regarding sub-tree pruning.

## Input

A data set that contains the dependent and independent variables for modeling.

## Configuration

For the Decision Tree Operator, the following relationship exists among the properties.

- The tree-growth phase refers to the **Minimal Gain** and **Maximal Depth** configuration properties.
- The pre-pruning phase refers to the **Number of Pre-Pruning Alternatives**, **Minimal Leaf Size**, and **Minimal Size for Split** configuration properties (in addition to the **Minimal Gain** and **Maximal Depth** parameters).
- The pruning phase refers to the **Confidence** configuration property.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the

Parameter	Description
	operator.
<b>Dependent Column</b>	<p>The quantity to model or predict. A <b>Dependent Column</b> must be specified for the decision tree. Select the data column to be considered the dependent variable for the regression.</p> <p><b>Dependent Column</b> must be a categorical type (including integer), such as <i>eye color = blue, green, or brown</i>.</p> <p><b>Note:</b> Null values are ignored during model training. If <b>Dependent Column</b> has three or more non-null values, a leaf node maps to each of the possible values.</p>
<b>Columns</b>	<p>Allows the user to select the independent variable data columns to include for the decision tree training. At least one column must be specified.</p> <p><b>Note:</b> All data types are allowed for the column names.</p> <p>Each independent column is limited to 1,000 distinct values.</p> <p>The number of leaf nodes for each independent variable depends on the data (that is, there can be more than two leaf nodes in decision trees).</p> <p>Click <b>Column Names</b> to open the dialog for selecting the available columns from the input data set for analysis.</p>
<b>Max Depth (-1=Unlimited)</b>	<p>Sets the "depth" of the tree or the maximum number of decision nodes it can branch out to beneath the root node. A tree stops growing any deeper if either a node becomes empty (that is, there are no more examples to split in the current node) or the depth of the tree exceeds this <b>Maximal Depth</b> limit.</p> <ul style="list-style-type: none"> <li>• <b>Maximal Depth</b> is used during the tree-growth phase.</li> <li>• The range of possible values is between -1 and any integer greater than 0.</li> <li>• A value of -1 represents "no bound" - the tree can take on any size or number of decision nodes until the nodes become empty.</li> </ul>



Parameter	Description
	Default value: 5.
<b>Confidence</b>	<p>Specifies the confidence % boundary to use for the pessimistic error algorithm of pruning.</p> <p><b>Confidence</b> controls the pruning phase of the decision tree algorithm.</p> <ul style="list-style-type: none"> <li>• The pruning phase uses confidence intervals to estimate the "worst case" error rate of the node.</li> <li>• The confidence level is the certainty factor or upper limit of the chance of an error being found in a leaf node.</li> <li>• If the node has an error rate greater than this <b>Confidence</b> limit, it is pruned. It could be thought of as the probability of there being an incorrect classification in the leaf node.</li> <li>• Setting a higher <b>Confidence</b> value allows the model to use nodes with higher individual error rates (less pruning).</li> <li>• Setting a lower <b>Confidence</b> value is indicating less tolerance for error and, therefore, more pruning.</li> <li>• <b>Confidence</b> applies only when pruning is set to <b>true</b> (that is, the <b>No Pruning</b> property is true). However it still must always be set to .50 or less or the system does not run - it would not make sense for there to be a greater than 50% chance of there being an error in a leaf node.</li> </ul> <p>Default value: <b>0.25</b>, representing a 25% probability of there being an error in the leaf node classification set.</p>
<b>Minimal Gain</b>	<p>Specifies the threshold gain in purity from parent to child node that must be achieved between various decision node options in order to justify producing a split. Once a split results in less than the minimal information gain setting, that node is made a leaf node.</p> <ul style="list-style-type: none"> <li>• <b>Minimal Gain</b> is used for the tree-growth phase.</li> <li>• The range of possible values is any decimal value <math>\geq 0</math>.</li> <li>• The way this is phrased in information theory is as the threshold of the amount of entropy that must be removed by a node split based</li> </ul>

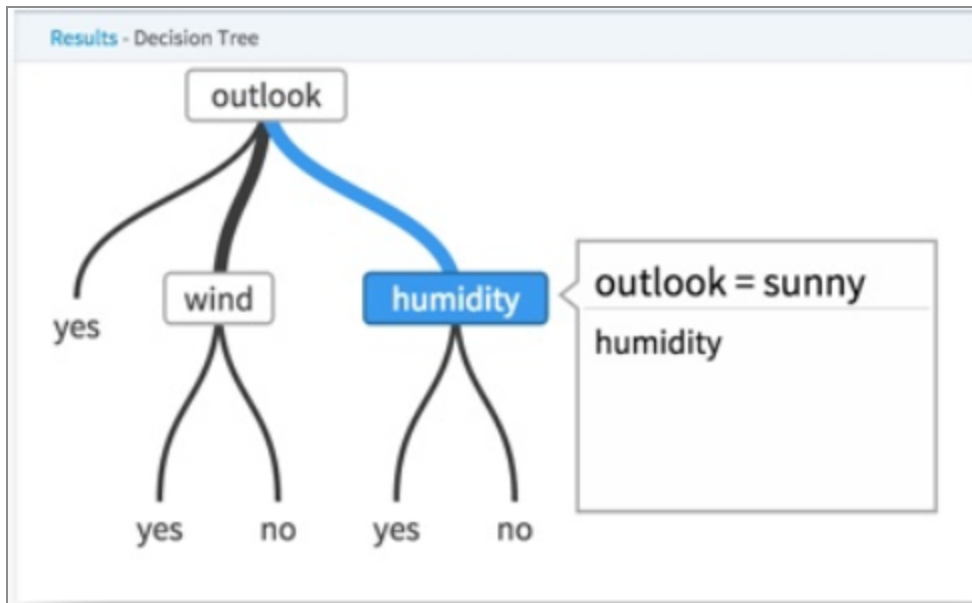
Parameter	Description
	<p>on a variable value.</p> <p>Default value: <b>0.1</b>. This represents a limit of 10% information gain needing to be achieved in order to justify splitting the node.</p>
<b>Number of Pre-Pruning Alternatives</b>	<p>Specifies the maximum number of alternative nodes allowed when pre-pruning the tree. This property is only relevant when pre-pruning is set to true (that is, the <b>No Pre-Pruning</b> property is not checked).</p> <p>The independent variable columns are sorted by their score (that is, information gain) before splitting occurs. During the pre-pruning phase, if the column with the bigger score does not meet the minimal leaf size condition, the column is ignored for splitting, and the next column is checked until the column meeting the condition is found or the <b>Number of Pre-Pruning Alternatives</b> is met.</p> <p>The range of possible values is any integer <math>\geq 0</math>.</p> <p>Default value: <b>3</b>.</p>
<b>Min Size for Split</b>	<p>Specifies the minimal size (or number of members) of a node in the decision tree in order to allow a further split. If the node has fewer data members than the <b>Minimal Size for Split</b>, it must become a leaf or end node in the tree.</p> <ul style="list-style-type: none"> <li>• The range of possible values is any integer <math>\geq 1</math>.</li> <li>• <b>Minimal Size for Split</b> is referenced during the pre-pruning phase.</li> </ul> <p>Default value: <b>4</b>.</p>
<b>No Pruning</b>	<p>If <b>true</b>, no pruning is performed. If <b>false</b> (the default), pruning is performed.</p>
<b>No Pre-pruning</b>	<p>If <b>true</b>, no pre-pruning is performed. If <b>false</b> (the default), pre-pruning is performed.</p> <p>For more information, see <a href="#">Pruning or Pre-Pruning</a>.</p>
<b>Min Leaf Size</b>	<p>Specifies the least number of data instances that can exist within a terminal leaf node of a decision tree.</p>

Parameter	Description
	<p>The range of possible values is any integer value <math>\geq 1</math>.</p> <ul style="list-style-type: none"> <li>• This setting limits the tree depth based on the size of the leaf nodes, ensuring enough data makes it to each part of the tree.</li> <li>• This is useful, for example, when the model construction is taking too long or when the model shows very good ROC on training data but not nearly as good performance on hold-out or cross-validation data (due to over-fitting).</li> <li>• For example, if the <b>Minimal Leaf Size</b> is 2, each terminal leaf node must contain at least 2 training data points.</li> </ul> <p>Default value: <b>2</b>.</p>
<b>Category Limit</b>	Limits the number of the categories into which a single categorical input column can split the data.
<b>Numerical Granularity</b>	Limits the number of the categories into which a single numerical input column can split the data.

## Output

### Visual Output

Highlight a decision tree node to see the decision criteria used to arrive at the node.



Decision trees need additional operators added to them in order to effectively analyze their effectiveness. Every decision tree modeling operator must be followed by a Predictor operator, which provides the prediction value for each data row compared against the actual data set training value and the associated confidence level.

The output from the Predictor operator looks like the following image.

DOC Use Case Decision Tree 1 Alpine Workflow Result							
Result Log	golfnew (3).csv	Decision Tree	Predictor				
outlook	temperature	humidity	wind	play	P_play	C_play	C_play_details
sunny	85	85	false	no	no	1	{{yes:0.0,no:1.0}}
sunny	80	90	true	no	no	1	{{yes:0.0,no:1.0}}
overcast	83	78	false	yes	yes	1	{{yes:1.0,no:0.0}}
overcast	64	65	true	yes	yes	1	{{yes:1.0,no:0.0}}
sunny	72	95	false	no	no	1	{{yes:0.0,no:1.0}}
sunny	69	70	false	yes	yes	1	{{yes:1.0,no:0.0}}
overcast	72	90	true	yes	yes	1	{{yes:1.0,no:0.0}}
rain	70	96	false	yes	yes	1	{{yes:1.0,no:0.0}}
rain	68	80	false	yes	yes	1	{{yes:1.0,no:0.0}}
rain	65	70	true	no	no	1	{{yes:0.0,no:1.0}}
rain	75	80	false	yes	yes	1	{{yes:1.0,no:0.0}}
rain	71	80	true	no	no	1	{{yes:0.0,no:1.0}}
sunny	75	70	true	yes	yes	1	{{yes:1.0,no:0.0}}
overcast	81	75	false	yes	yes	1	{{yes:1.0,no:0.0}}

The **P\_play** prediction column shows a value of yes or no and uses a threshold assumption of > than 50% confidence that the prediction happens.

The **C\_play** column indicates the confidence that the Dependent value is 1.

- Usually, **C\_play** is a decimal value. In this example, the data set is small and created as an example.

The **C\_play** details column indicates the confidence for each possible value of the dependent variable.

### Using With Other Scoring Operators

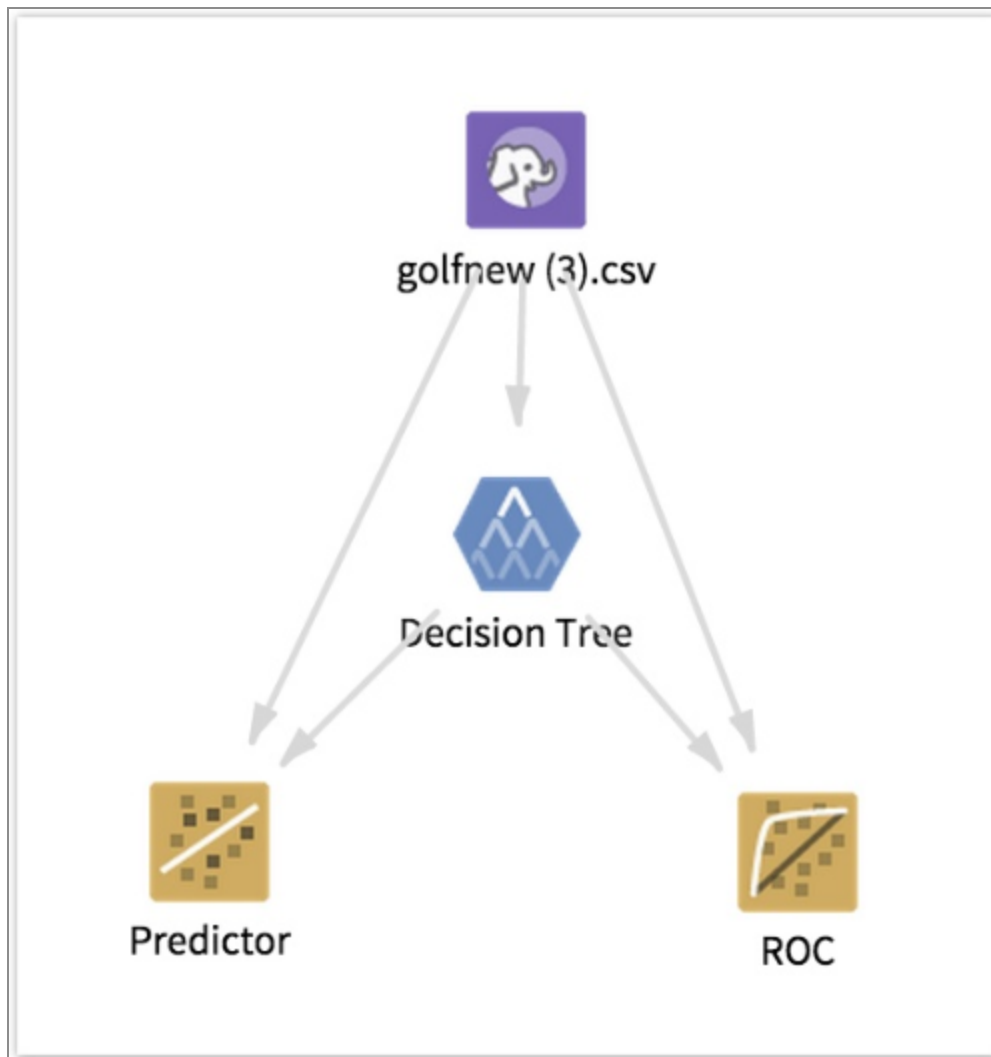
Adding additional scoring operators, such as a ROC graph, is also helpful in immediately assessing how predictive the Decision Tree model is.

### Data Output

None. This is a terminal operator.

### Example

The following image shows a typical Analytic Flow configuration for Decision Tree operators.



## Decision Tree - MADlib

TIBCO Data Science - Team Studio supports the MADlib Decision Tree model implementation.



## Information at a Glance

Parameter	Description
Category	Model
MADlib version	< 1.8
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

For more information about working with decision trees, see [Classification Modeling with Decision Tree](#).

## Algorithm

The Decision Tree (MADlib) Operator supports the C4.5 deterministic method for constructing the decision tree structure, allowing users to choose information gain, Gini coefficient, or gain ratio as the split criteria. The MADlib implementation also supports decision tree pruning and missing value handling.

Note that the MADlib Decision Tree is considered an 'Early Stage Development' algorithm.

More information including general principles can be found in the [official MADlib documentation](#).

## Input

A data set that contains the dependent and independent variables for modeling.

## Configuration

Parameter	Description
Notes	Notes or helpful information about this operator's parameter settings. When

Parameter	Description
	you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>MADlib Schema Name</b>	The schema where MADlib is installed in the database. MADlib must be installed in the same database as the input data set. If a madlib schema exists in the database, this parameter defaults to <b>madlib</b> .
<b>Split Criterion</b>	The criterion used to determine the split of the data at each node of the tree. The Split Criterion can be the <b>Information Gain</b> , <b>Gini Coefficient</b> , or <b>Information Gain Ratio</b> .
<b>Model Output Schema Name</b>	The name of the schema where the output is stored.
<b>Model Output Table Name</b>	<p>The name of the table that is created to store the Regression model. The model output table stores:</p> <p>id   tree_location   feature   probability   ebp_coeff   maxclass   scv   live   sample_size   parent_id   lmc_nid   lmc_fval   is_continuous   split_value   tid   dp_ids</p> <p>See the <a href="#">official MADlib decision tree documentation</a> for more info.</p>
<b>Drop If Exists</b>	<ul style="list-style-type: none"> <li>• If <b>Yes</b> (the default), drop the existing table of the same name and create a new one.</li> <li>• If <b>No</b>, stop the flow and alert the user that an error has occurred.</li> </ul>
<b>Validation Table Name</b>	<p>The table name for a validation data set to score the learned decision tree model against. A ratio of correctly classified items in the validation set is given.</p> <p>Default value: null (or no validation table).</p>
<b>Continuous Features</b>	<p>The user can select the continuous <b>Independent Variable</b> data columns for the decision tree training.</p> <p>At least one <b>Continuous Features</b> column or one <b>Categorical Features</b> column must be specified.</p>



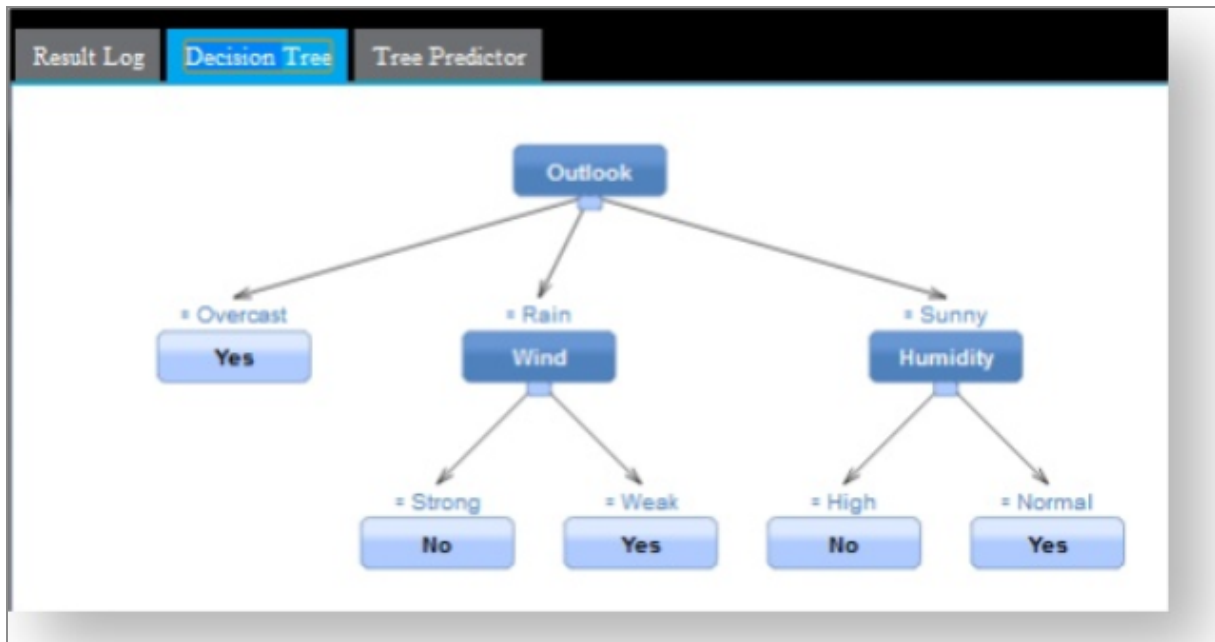
Parameter	Description
	Click <b>Column Names</b> to open the dialog for selecting the available columns from the input data set for analysis.
<b>Categorical Features</b>	<p>The user can select the categorical <b>Independent Variable</b> data columns to include for the decision tree training.</p> <p>At least one <b>Continuous Features</b> column or one <b>Categorical Features</b> column must be specified.</p>
<b>Class Column</b>	Required. The data column to be the Dependent Variable. This is the quantity to model or predict.
<b>Confidence Level</b>	<p>Specifies the confidence % boundary to use for the pessimistic error algorithm of pruning.</p> <p><b>Confidence Level</b> controls the pruning phase of the Decision Tree algorithm.</p> <ul style="list-style-type: none"> <li>• The pruning phase uses confidence intervals to estimate the "worst case" error rate of the node.</li> <li>• The confidence level is the certainty factor or upper limit of the chance of an error being found in a leaf node.</li> <li>• If the node has an error rate greater than this Confidence limit, it is pruned. Consider this as the probability of there being an incorrect classification in the leaf node.</li> <li>• Setting a higher <b>Confidence Level</b> value allows the model to use nodes with higher individual error rates (less pruning).</li> <li>• Setting a lower <b>Confidence Level</b> value indicates less tolerance for error, therefore more pruning.</li> </ul> <p>Default value: 25, representing a 25% probability of there being an error in the leaf node classification set.</p>
<b>Handle Missing Values</b>	<p>Specifies how to handle missing values in the data set.</p> <ul style="list-style-type: none"> <li>• <b>ignore</b> - Missing values are ignored.</li> <li>• <b>explicit</b> - Missing values are explicitly replaced with the average value for the feature.</li> </ul>

Parameter	Description
	Default value: <b>ignore</b>
<b>Maximum Tree Depth</b>	<p>Sets the "depth" of the tree or the maximum number of decision nodes it can branch out to beneath the root node. A tree stops growing any deeper if either a node becomes empty (that is, there are no more examples to split in the current node) or the depth of the tree exceeds this <b>Maximal Tree Depth</b> limit.</p> <p><b>Maximal Tree Depth</b> is used during the tree-growth phase.</p> <p>Values must be greater than 0.</p> <p>Default value: <i>10</i></p>
<b>Node Prune Threshold</b>	<p>The minimum percentage of the number of records required in a child node. This threshold applies only to the non-root nodes.</p> <p>The value must be in <math>[0,1]</math> .</p> <ul style="list-style-type: none"> <li>• If the value is <i>1</i>, the trained tree only has one node (the root node)</li> <li>• If the value is <i>0</i>, no nodes are pruned by this parameter.</li> </ul> <p><b>Note:</b> You can use pruning to avoid overfitting the decision tree.</p>
<b>Node Split Threshold</b>	<p>Minimum percentage of the number of records required in a node for a further split to be possible.</p> <p>The value must be in <math>[0,1]</math> .</p> <ul style="list-style-type: none"> <li>• If the value is <i>1</i>, the trained tree only has two levels, since only the root node can grow.</li> <li>• If the value is <i>0</i>, then the tree can grow extensively.</li> </ul>
<b>Verbosity</b>	A Boolean value that indicates whether to log all output of the training results. Default: <b>false</b> .

## Output

### Visual Output

The Decision Tree (MADlib) Operator has an intuitive output - the classification tree structure with the leaf nodes that indicate the count of data set rows (members) they contain.



Double-click a decision tree node if its sub-nodes are not displayed in the UI.

## Additional Notes

### Output Details

Connect this operator to the following succeeding operators.

- Predictor operators
- Scoring operators (such as ROC)

Decision trees need succeeding operators to effectively analyze their effectiveness. A Predictor operator provides the prediction value for each data row, compared against the actual data set training value and the associated confidence level.

Adding additional scoring operators, such as a ROC graph, is also helpful in immediately assessing how predictive the Decision Tree model is. For the ROC graph, any AUC value over .80 is typically considered a "good" model. A value of 0.5 just means the model is no better than a "dumb" model that can guess the right answer half the time.

The output from the Predictor operator appears as follows:

<div>Result Log</div> <div>Decision Tree</div> <div>Tree Predictor</div> <div>ROC</div>								
Day	Outlook	Temperature	Humidity	Wind	PlayTennis	P(PlayTennis)	C(Yes)	C(No)
D1	Sunny	Hot	High	Weak	No	No	0	1
D2	Sunny	Hot	High	Strong	No	No	0	1
D3	Overcast	Hot	High	Weak	Yes	Yes	1	0
D4	Rain	Mild	High	Weak	Yes	Yes	1	0
D5	Rain	Cool	Normal	Weak	Yes	Yes	1	0
D6	Rain	Cool	Normal	Strong	No	No	0	1
D7	Overcast	Cool	Normal	Strong	Yes	Yes	1	0
D8	Sunny	Mild	High	Weak	No	No	0	1
D9	Sunny	Cool	Normal	Weak	Yes	Yes	1	0
D10	Rain	Mild	Normal	Weak	Yes	Yes	1	0
D11	Sunny	Mild	Normal	Strong	Yes	Yes	1	0
D12	Overcast	Mild	High	Strong	Yes	Yes	1	0
D13	Overcast	Hot	Normal	Weak	Yes	Yes	1	0
D14	Rain	Mild	High	Strong	No	No	0	1

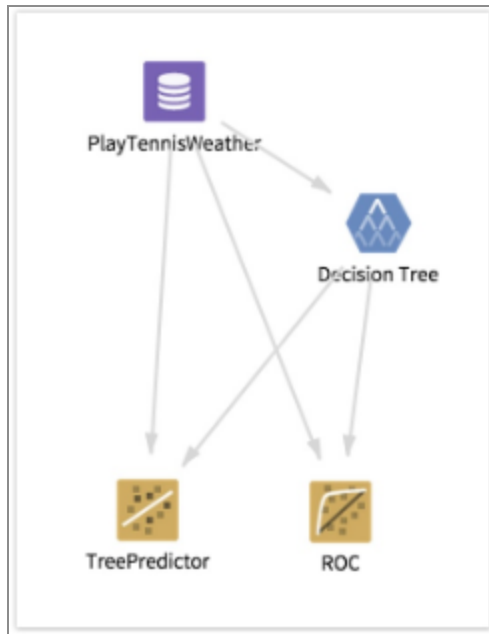
- The prediction value (Yes or No) uses a threshold assumption of > than 50% confidence that the prediction happens.
- The **C(Yes)** column indicates that the confidence that the Dependent value is 1.

**i Note:** Usually this is a decimal value. In this case, the data set is small and created as an example.

- The **C(No)** column indicates the confidence that the Dependent value is 0.

## Example

The following example illustrates a typical analytic flow configuration for Decision Tree operators.



## Decision Tree Classification - CART

Uses the MADlib built-in function `tree_train()` to generate a decision tree that predicts the value of a categorical column based on several independent columns.




### Information at a Glance

Parameter	Description
Category	Model
MADlib version	1.8+
Data source type	DB
Send output to other operators	Yes

Parameter	Description
Data processing tool	n/a

The generated tree is a binary tree, with each node representing either a branching condition or a predicted value. The output of the operator can be sent to a predictor or confusion matrix. MADlib 1.8 or higher must be installed on the database.

 **Important:** This operator does not work with MADlib 1.7.1 or lower, due to a change in the way MADlib handles column names. If you have an older version of MADlib, consider using [Decision Tree - MADlib](#) instead.

For more information about working with decision trees, see [Classification Modeling with Decision Tree](#).

## Input

The input table must have a single, categorical (string or integer) column to predict, and one or more independent columns to serve as input.

### Bad or Missing Values

In the source table, any rows that contain null values for the predicted or independent columns are ignored.

## Restrictions

- This operator works only on databases with MADlib 1.8+ installed.
- Source data tables must have a numeric ID column that uniquely identifies each row in the source table.
- The prediction column must be numeric, and all predictions are double-precision values.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>MADlib Schema</b>	Name of schema where MADlib is installed. By default, this is <code>madlib</code> .
<b>Model Output Schema</b>	The name of the schema to use for MADlib-generated output tables.
<b>Model Output Table</b>	The name of the MADlib-generated output table. This table is generated by the tree trainer. An additional table with the same name and the suffix <code>_summary</code> also is generated.
<b>Drop If Exists</b>	<ul style="list-style-type: none"> <li>• If <b>Yes</b> (the default), drop the existing table of the same name and create a new one.</li> <li>• If <b>No</b>, stop the flow and alert the user that an error has occurred.</li> </ul>
<b>ID column</b>	All source tables must have a numeric ID column to uniquely identify each row.
<b>Dependent Variable</b>	The name of the numeric column to predict. This must be a floating-point column.
<b>Feature List</b>	Click <b>Select Columns</b> to specify one or more columns to use as independent variables to predict the dependent variable. See <a href="#">Select Columns dialog</a> for more information.
<b>Split Criterion</b>	The algorithm to use for calculating branch nodes during tree generation. For categorical tables, this must be <b>gini</b> , <b>entropy</b> , or <b>misclassification</b> . The default is <b>gini</b> .
<b>Maximum Tree Depth</b>	The generated tree does not exceed this depth. If not specified, the default is <code>10</code> .

Parameter	Description
<b>Minimum Observations Before Splitting</b>	If not specified, the default is 20.
<b>Minimum Observations in Terminal Nodes</b>	If not specified, the default is the minimum observations before splitting, divided by 3.
<b>Number of Bins for Split Boundaries</b>	If not specified, the default is 100.

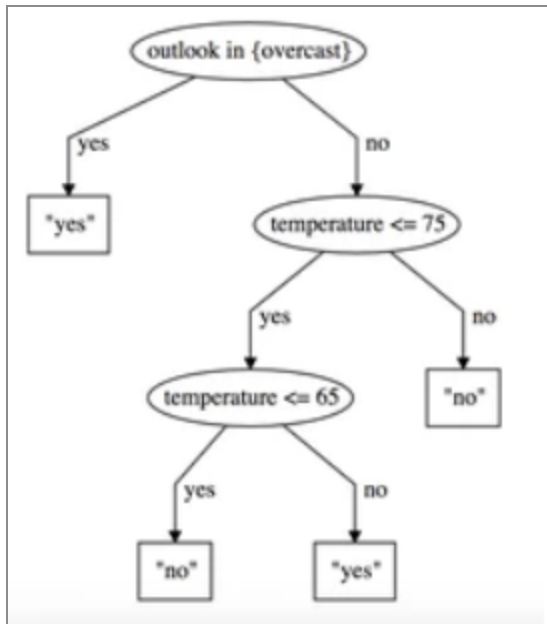
## Outputs

### Visual Output

This operator produces the following tabs.

- **Decision Tree Text** - Contains a text representation of the generated decision tree. Each branch node contains a number of rows and a prediction. Branch nodes also contain a branching condition.
- **Decision Tree Graph** - Contains a tree graph. Branches reflect split conditions and associated predictions.





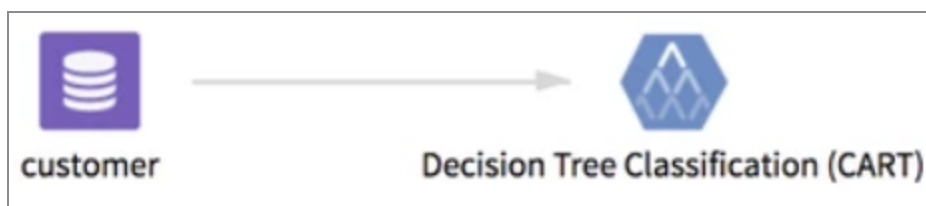
## Additional Notes

### Output to Succeeding Operators

Connect this operator to the following succeeding operators.

- Prediction
- Confusion Matrix

## Example



## Decision Tree Regression - CART

Generates a decision tree that predicts the value of a numeric column based on several independent columns.



## Information at a Glance

Parameter	Description
Category	Model
MADlib version	1.8+
Data source type	DB
Send output to other operators	Yes
Data processing tool	MADlib

This operator uses the MADlib built-in function, `tree_train()`. The generated tree is a binary tree, with each node representing either a branching condition or a predicted value. The output of the operator can be sent to a predictor or confusion matrix. MADlib 1.8 or higher must be installed on the database.

**Important:** This operator does not work with MADlib 1.7.1 or lower, due to a change in the way MADlib handles column names. If you have an older version of MADlib, consider using [Decision Tree - MADlib](#) instead.

For more information about working with decision trees, see [Classification Modeling with Decision Tree](#).

## Input

The input table must have a single, numeric (floating point) column to predict, and one or more independent columns to serve as input.

## Bad or Missing Values

Any rows in the source table that contains NULL values for the predicted or independent columns are ignored.

## Restrictions

This operator works only on databases with MADlib 1.8+ installed. Source data tables must have a numeric ID column that uniquely identifies each row in the source table. The prediction column must be numeric, and all predictions are double-precision values.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>MADlib Schema</b>	The name of schema where MADlib is installed. By default, this is madlib.
<b>ID column</b>	All source tables must have a numeric ID column to uniquely identify each row.
<b>Dependent Variable</b>	Name of numeric column to predict. This must be a floating-point column.
<b>Feature List</b>	Selection of one or more columns to use as independent variables to predict the dependent variable.
<b>Model Output Schema</b>	The name of the schema to use for MADlib-generated output tables.
<b>Model Output Table</b>	The name of the MADlib-generated output table. This table is generated by the tree trainer. An additional table with the same name and the suffix <code>_summary</code> is also generated.
<b>Drop Model Output Tables if</b>	If yes, generated output tables from a previous run are dropped first. If no, and if the tables already exist when the tree trainer is run, an error

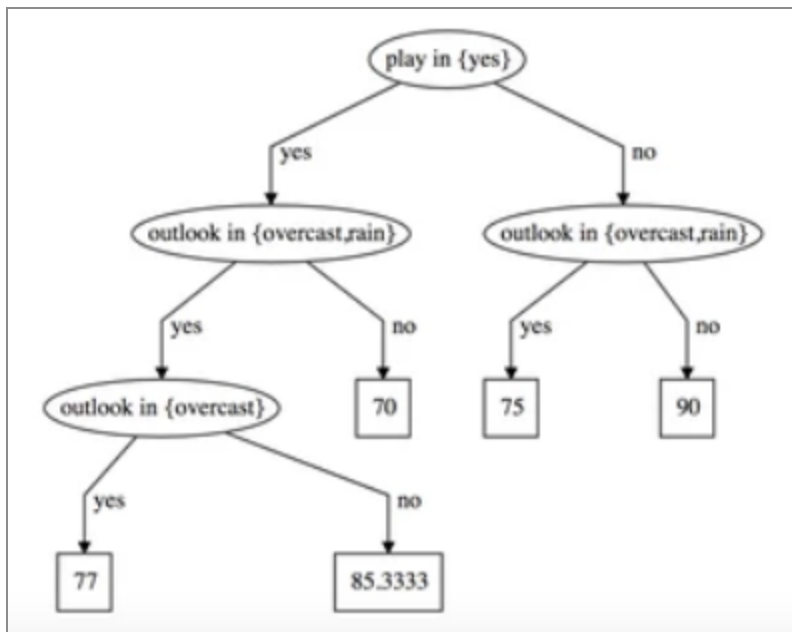
Parameter	Description
<b>They Exist?</b>	occurs.
<b>Split Criterion</b>	The algorithm for calculating branch nodes during tree generation. For regression tables, this algorithm is always <b>mse</b> .
<b>Maximum Tree Depth</b>	The generated tree does not exceed this depth. If not specified, the default is <i>10</i> .
<b>Minimum Observations Before Splitting</b>	If not specified, the default is <i>20</i> .
<b>Minimum Observations in Terminal Nodes</b>	If not specified, the default is the minimum observations before splitting, divided by 3.
<b>Number of Bins for Split Boundaries</b>	If not specified, the default is <i>100</i> .

## Outputs

### Visual Output

This operator produces the following tabs.

- **Decision Tree Text** - Contains a text representation of the generated decision tree. Each branch node contains a number of rows and a prediction. Branch nodes also contain a branching condition.
- **Decision Tree Graph** - Contains a tree graph. Branches reflect split conditions and associated predictions.



## Additional Notes

### Output to Succeeding Operators

Connect this operator to the following succeeding operators.

- Prediction
- Confusion Matrix

## Example



## Elastic Net Linear - MADlib

TIBCO Data Science - Team Studio supports the MADlib open-source implementation of the Elastic Net Linear Regression algorithm. This operator implements MADlib's open-source elastic net regularization algorithm for linear regression problems.



### Information at a Glance

Parameter	Description
Category	Model
Data source type	DB
Send output to other operators	Yes
Data processing tool	MADlib

### Algorithm

Elastic net regularization seeks to find a weight vector that, for any given training example set, minimizes a metric function that combines the L1 and L2 penalties of the lasso and ridge regression methods.

More information including general principles can be found in the [official MADlib documentation](#).

### Input

A data set that contains the dependent and independent variables for modeling.

## Configuration

The following parameters must be set for minimal configuration.

- **MADlib Schema Name**
- **Model Output Table Name**
- **Dependent Variable**
- **Independent Variables**

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>MADlib Schema Name</b>	The schema where MADlib is installed in the database. MADlib must be installed in the same database as the input data set. If a "madlib" schema exists in the database, this parameter defaults to madlib.
<b>Model Output Schema Name</b>	The name of the schema where the output is stored.
<b>Model Output Table Name</b>	<p>The name of the table that is created to store the Regression model. The model output table stores the following.</p> <p>family   features   features_selected   coef_nonzero   coef_all   intercept   log_likelihood   standardize   iteration_run</p> <p>See the <a href="#">official MADlib elastic net regularization documentation</a> for more information.</p>
<b>Drop If Exists</b>	<ul style="list-style-type: none"> <li>• If <b>Yes</b> (the default), drop the existing table of the same name and create a new one.</li> <li>• If <b>No</b>, stop the flow and alert the user that an error has occurred.</li> </ul>
<b>Dependent Variable</b>	The quantity to model or predict. The list of the available data columns for the Elastic Net Linear operator are displayed. Select the data column to be considered the dependent variable for the regression. The

Parameter	Description
	dependent variable should be a numerical data type.
<b>Independent Variables</b>	<p>Allows the user to select the independent variable data columns to include for the regression analysis or model training. At least one column or one interaction variable must be specified.</p> <p>Click <b>Select Columns</b> to open the <a href="#">Select Columns dialog</a> and select the available columns from the input data set for analysis.</p>
<b>Control Parameter</b>	The elastic net control parameter (alpha) must be a value between 0 and 1, inclusive.
<b>Regularization Parameter</b>	Must be a positive value.
<b>Standardize</b>	<p>Specifies whether to normalize the data.</p> <ul style="list-style-type: none"> <li>• If <b>true</b> (the default), normalizes the data. This option often yields better results and faster convergence.</li> <li>• If <b>false</b>, does not normalize the data.</li> </ul>
<b>Optimizer</b>	<p>Can be Fast Iterative Shrinkage-Thresholding Algorithm (<b>FISTA</b>) or Incremental Gradient Descent (<b>IGD</b>). The required parameters for the optimizer configuration are dependent on the optimizer selected.</p> <p>See the <a href="#">official MADlib elastic net regularization documentation</a> for more information.</p>
<b>FISTA Maximum Stepsize</b>	<p>The initial backtracking step size. At each iteration, the algorithm first tries <math>stepsize = max\_stepsize</math>, and if it does not work, it then tries a smaller step size, <math>stepsize = stepsize/eta</math>, where <math>eta</math> must be larger than 1. At first, this seems to perform repeated iterations for even one step, but using a larger step size actually greatly increases the computation speed and minimizes the total number of iterations. A careful choice of <math>max\_stepsize</math> can decrease the computation time by more than 10 times.</p> <p>Default value: <b>4.0</b>.</p>
<b>FISTA Eta</b>	If $stepsize$ does not work, $stepsize / eta$ is tried. Must be greater than 1.



Parameter	Description
	Default value: <b>2.0</b> .
<b>Warmup</b>	<p>A value of <b>true</b> specifies a series of lambda values, which is strictly descent and ends at the lambda value that the user wants to calculate, is used. The larger lambda gives a very sparse solution, and the sparse solution again is used as the initial guess for the next lambda's solution, which speeds up the computation for the next lambda. For larger data sets, this can sometimes accelerate the whole computation and may be faster than computation on only one lambda value.</p> <p>A value of <b>false</b> (the default) specifies that this warmup procedure is not performed.</p>
<b>Warmup Lambdas</b>	The lambda value series to use when <b>Warmup</b> is <b>true</b> . The default is <b>NULL</b> , which means that lambda values are automatically generated.
<b>Number of Warmup Lambdas</b>	<p>The number of lambdas to use in <b>Warmup</b>. If <code>warmup_lambdas</code> is not <b>NULL</b>, this value is overridden by the number of provided lambda values.</p> <p>Default value: <b>15</b>.</p>
<b>Warmup Tolerance</b>	<p>The value of tolerance used during warmup.</p> <p>Default value: <b>1e-6</b>.</p>
<b>FISTA Use Active Method</b>	<p>A value of <b>true</b> specifies an active-set method is used to speed up the computation. Considerable speedup is obtained by organizing the iterations around the active set of features - those with nonzero coefficients. After a complete cycle through all the variables, we iterate on only the active set until convergence. If another complete cycle does not change the active set, we are done; otherwise the process is repeated.</p> <p>A value of <b>false</b> (the default) specifies that the active-set method is not used.</p>
<b>FISTA Active Tolerance</b>	<p>The value of tolerance used during active set calculation.</p> <p>Default value: <b>1e-6</b>.</p>

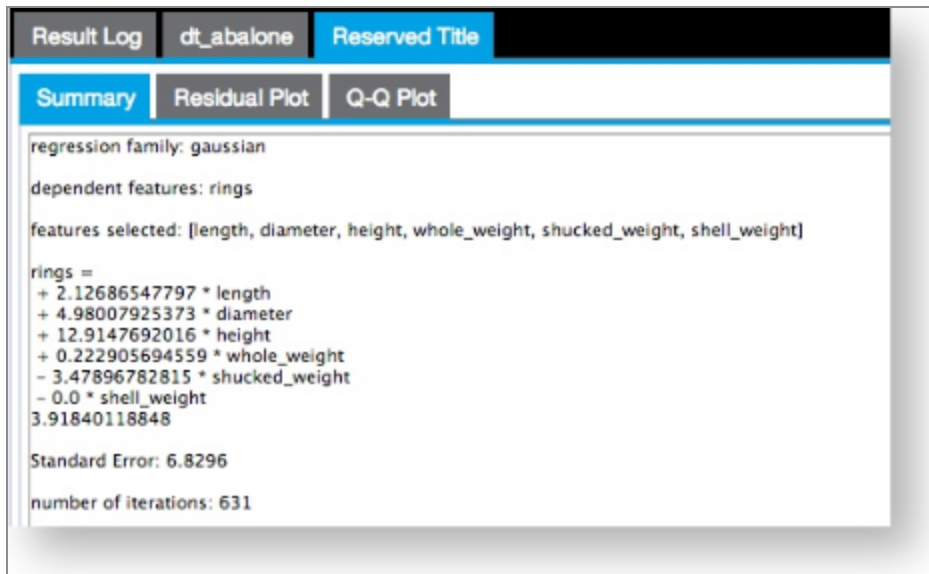
Parameter	Description
<b>FISTA Random Step Size</b>	<p>Whether to add some randomness to the step size. Sometimes, this can speed up the calculation.</p> <p>Default value: <b>1e-6</b>.</p>
<b>IGD Step Size</b>	<p>Initial backtracking step size.</p> <p>Default value: <b>0.01</b>.</p>
<b>IGD Zero Coefficient Threshold</b>	<p>When a coefficient is very small, set this coefficient to <b>0</b>.</p> <p>Due to the stochastic nature of SGD, only very small values can be obtained for the fitting coefficients. Therefore, threshold is needed at the end of the computation to screen out tiny values and hard-set them to zeros. This is accomplished as follows.</p> <ol style="list-style-type: none"> <li>1. Multiply each coefficient with the standard deviation of the corresponding feature.</li> <li>2. Compute the average of absolute values of re-scaled coefficients.</li> <li>3. Divide each re-scaled coefficient with the average, and if the resulting absolute value is smaller than threshold, set the original coefficient to zero.</li> </ol> <p>Default value: <b>1e-10</b>.</p>
<b>IGD Parallelize</b>	<p>A value of <b>true</b> specifies that the computation should be run on multiple segments.</p> <p>SGD is a sequential algorithm in nature. When running in a distributed manner, each segment of the data runs its own SGD mode, and then the models are averaged to get a model for each iteration. This averaging might slow down the convergence speed, although the ability to process large data sets on multiple machines is also acquired. This algorithm, therefore, provides the parallel option to allow you to choose whether to do parallel computation.</p> <p>Default value: <b>true</b>.</p>
<b>Maximum</b>	When the difference between coefficients of two consecutive iterations is

Parameter	Description
<b>Iterations</b>	<p>smaller than the <b>Convergence Tolerance</b> or the iteration number is larger than <b>Maximum Iterations</b>, the computation stops.</p> <p>Default value: <b>10000</b>.</p>
<b>Convergence Tolerance</b>	<p>When the difference between coefficients of two consecutive iterations is smaller than the <b>Convergence Tolerance</b> or the iteration number is larger than <b>Maximum Iterations</b>, the computation stops.</p> <p>Default value: <b>1e-6</b>.</p>
<b>Draw Residual Plot</b>	<p>Specifies the option to draw the residual plot and Q-Q (Quantile-Quantile) plot used for model validation.</p> <p>The residual plot displays a graph that shows the residuals of a linear regression model on the vertical axis and the independent variable on the horizontal axis.</p> <p>The Q-Q plot graphically compares the distribution of the residuals of a given variable to the normal distribution (represented by a straight line).</p> <p>Default value: <b>true</b>, meaning that the regression operator has two additional outputs showing the residual plot and Q-Q plot.</p>

## Output

### Visual Output

Results are displayed across the **Summary**, **Residual Plot**, and **Q-Q Plot** tabs.



See the [Linear Regression - MADlib](#) and [Official MADlib elastic net regularization documentation](#) for more information.

## Data Output

None.

## Elastic Net Logistic - MADlib

TIBCO Data Science - Team Studio supports the MADlib implementation of the Elastic Net Logistic Regression algorithm.



## Information at a Glance

Parameter	Description
Category	Model

Parameter	Description
Data source type	DB
Send output to other operators	Yes
Data processing tool	MADlib

## Algorithm

This module implements elastic net regularization for logistic regression problems. Elastic net regularization seeks to find a weight vector that, for any given training example set, minimizes a metric function that combines the L1 and L2 penalties of the lasso and ridge regression methods.

For more information, including general principles, see the [official MADlib documentation](#).

## Input

A database data set that contains the dependent and independent variables for modeling. The data set must have at least one Boolean column and at least one numeric type column.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>MADLib Schema Name</b>	The schema where MADlib is installed in the database. MADlib must be installed in the same database as the input data set. If a "madlib" schema exists in the database, this parameter defaults to madlib.
<b>Model Output Schema Name</b>	The name of the model output schema.

Parameter	Description
<b>Model Output Table Name</b>	<p>The name of the table that is created to store the regression model.</p> <p>The model output table stores the following. family   features   features_selected   coef_nonzero   coef_all   intercept   log_likelihood   standardize   iteration_run</p> <p>See the <a href="#">official MADlib elastic net regularization documentation</a> for more information.</p>
<b>Drop If Exists</b>	<ul style="list-style-type: none"> <li>• If <b>Yes</b> (the default), drop the existing table of the same name and create a new one.</li> <li>• If <b>No</b>, stop the flow and alert the user that an error has occurred.</li> </ul>
<b>Dependent Variable</b>	<p>The quantity to model or predict. This must be a Boolean column. The list of the available data columns for the Elastic Net Logistic operator is displayed. Select the data column to be considered the dependent variable for the regression.</p>
<b>Independent Variables</b>	<p>Select the data columns to include for the regression analysis or model training. At least one column or one interaction must be specified.</p> <p>Click <b>Select Columns</b> to open the dialog for selecting the available columns from the input data set for analysis.</p> <p>See: <a href="#">Select Columns dialog</a>.</p>
<b>Control Parameter</b>	<p>The elastic net <b>Control Parameter</b> (alpha) must be a value in [0,1].</p>
<b>Regularization Parameter</b>	<p>The <b>Regularization Parameter</b> (lambda) must be a positive value.</p>
<b>Standardize</b>	<p>A Boolean flag that specifies the option to normalize the data. The default value is <b>true</b>, which often yields better results and faster convergence.</p>
<b>Optimizer</b>	<p>Can be fista (Fast Iterative Shrinkage-Thresholding Algorithm) or igd (Incremental Gradient Descent). The required proceeding optimizer configuration parameters are dependent on the optimizer selected.</p>

Parameter	Description
	See the <a href="#">official MADlib elastic net regularization documentation</a> for more information.
<b>FISTA Maximum Stepsize</b>	<p>The initial backtracking step size. At each iteration, the algorithm first tries <code>stepsize = max_stepsize</code>, and if it does not work, it tries a smaller step size, <code>stepsize = stepsize/eta</code>, where <code>eta</code> must be larger than 1. At first glance, this seems to perform repeated iterations for even one step, but using a larger step size actually greatly increases the computation speed and minimizes the total number of iterations. A careful choice of <code>max_stepsize</code> can decrease the computation time by more than 10 times.</p> <p>Default value: <b>4.0</b>.</p>
<b>FISTA Eta</b>	<p>If <code>stepsize</code> does not work, <code>stepsize /eta</code> is tried. This value must be greater than 1.</p> <p>Default value: <b>2.0</b>.</p>
<b>Warmup</b>	<ul style="list-style-type: none"> <li>If <b>true</b>, a series of lambda values, which is strictly descent and ends at the lambda value that the user wants to calculate, is used. The larger lambda gives a very sparse solution, and the sparse solution again is used as the initial guess for the next lambda's solution, which speeds up the computation for the next lambda. For larger data sets, this can sometimes accelerate the whole computation and might be faster than computation on only one lambda value.</li> <li>If <b>false</b> (the default), this warmup procedure is not performed.</li> </ul>
<b>Warmup Lambdas</b>	The lambda value series to use when warmup is <b>true</b> . The default is <b>NULL</b> , which means that lambda values are automatically generated.
<b>Number of Warmup Lambdas</b>	<p>The number of lambdas used in warmup. If <code>warmup_lambdas</code> is not <b>NULL</b>, this value is overridden by the number of provided lambda values.</p> <p>Default value: <b>15</b>.</p>
<b>Warmup Tolerance</b>	The value of tolerance used during warmup. Default value: <b>1e-6</b> .

Parameter	Description
<b>FISTA Use Active Method</b>	<ul style="list-style-type: none"> <li>If <b>true</b>, an active-set method is used to speed up the computation. Considerable speedup is obtained by organizing the iterations around the active set of features - those with nonzero coefficients. After a complete cycle through all the variables, we iterate on only the active set until convergence. If another complete cycle does not change the active set, we are done, otherwise the process is repeated.</li> <li>If <b>false</b> (the default), the active-set method is not used.</li> </ul>
<b>FISTA Active Tolerance</b>	The value of tolerance used during active set calculation. Default value: <b>1e-6</b> .
<b>FISTA Random Step Size</b>	Specify whether to add some randomness to the step size. Sometimes, this can speed up the calculation. Default value: <b>1e-6</b> .
<b>IGD Step Size</b>	The initial backtracking step size. Default value: <b>0.01</b> .
<b>IGD Zero Coefficient Threshold</b>	<p>When a coefficient is very small, set this coefficient to <b>0</b>. Due to the stochastic nature of SGD, we can only obtain very small values for the fitting coefficients. Therefore, threshold is needed at the end of the computation to screen out tiny values and hard-set them to zeros. This is accomplished as follows.</p> <ol style="list-style-type: none"> <li>1. Multiply each coefficient with the standard deviation of the corresponding feature.</li> <li>2. Compute the average of absolute values of re-scaled coefficients.</li> <li>3. Divide each rescaled coefficient with the average, and if the resulting absolute value is smaller than threshold, set the original coefficient to zero.</li> </ol> <p>Default value: <b>1e-10</b>.</p>
<b>IGD Paralellize</b>	Specify whether to run the computation on multiple segments. SGD is a sequential algorithm in nature. When running in a distributed manner, each segment of the data runs its own SGD model and then the models are averaged to get a model for each iteration. This averaging might slow



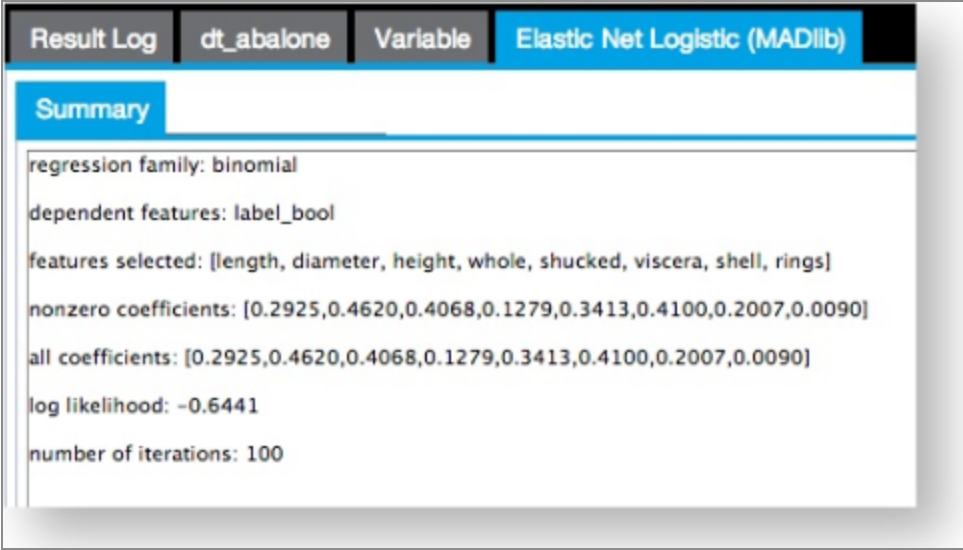
Parameter	Description
	<p>down the convergence speed, although we also acquire the ability to process large data sets on multiple machines. This algorithm, therefore, provides the parallel option to allow you to choose whether to do parallel computation.</p> <p>Default value: <b>true</b>.</p>
<b>Maximum Iterations</b>	<p>When the difference between coefficients of two consecutive iterations is smaller than the convergence tolerance or the iteration number is larger than <b>Maximum Iterations</b>, the computation stops.</p> <p>Default value: <b>10000</b>.</p>
<b>Convergence Tolerance</b>	<p>When the difference between coefficients of two consecutive iterations is smaller than the <b>Convergence Tolerance</b> or the iteration number is larger than Maximum Iterations, the computation stops.</p> <p>Default value: <b>1e-6</b>.</p>

## Output

### Visual Output

Output is displayed on the **Summary** tab, which displays the features selected, non-zero coefficients, all coefficients, log likelihood, and number of iterations run until termination. Additional assessment is often done using ROC, LIFT, and Logistic Regression Prediction Operator results.

See the [Official MADlib documentation for elastic net regularization](#) for more information.



Data Output

None.

Generalized Linear Regression Models

Fits a regression model to predict a dependent variable that follows some distribution from the exponential family of distributions.



Information at a Glance

Parameter	Description
Category	Model
Data source type	HD

Parameter	Description
Send output to other operators	Yes
Data processing tool	Spark

For example, if you have the National Transportation Safety Board's data set of the number of auto accidents by states in a year, you could use the [Poisson distribution](#) to fit a model that can predict future accident counts based on the predictor variables available in the data set. TIBCO Data Science - Team Studio leverages the [Mllib implementation](#) of generalized linear regression, so you should have Spark version 2.0 or later.

You can connect this operator to the [Predictor \(DB\)](#) to obtain predictions on new data.

## Input

A tabular input on Hadoop. The input should contain at least one numeric column that represents the dependent variable, and any number of columns that represent the independent variables. The operator [one-hot encodes](#) all of the string columns selected as independent variables.

### Bad or Missing Values

The training example is dropped from the data set if any of the predictors or the dependent variable is missing.

## Restrictions

To run binomial regression on a string-dependent column, you must first string index the column to produce a numeric-dependent column.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.

Parameter	Description
<b>Distribution Family</b>	<p>Distribution of the dependent variable.</p> <ul style="list-style-type: none"> <li>• <b>gaussian</b> (the default)</li> <li>• <b>binomial</b></li> <li>• <b>poisson</b></li> <li>• <b>gamma</b></li> </ul>
<b>Link Function</b>	<p>The link function that defines the relationship between the expected value of the dependent variable and the linear predictor.</p> <ul style="list-style-type: none"> <li>• <b>cloglog</b></li> <li>• <b>identity</b></li> <li>• <b>inverse</b></li> <li>• <b>log</b> (the default)</li> <li>• <b>logit</b></li> <li>• <b>probit</b></li> <li>• <b>sqrt</b></li> </ul>
<b>Dependent Column</b>	A numeric column to use as the output.
<b>Independent Columns</b>	<p>One or more columns to serve as input.</p> <p>Spark currently supports up to 4096 features.</p>
<b>Max. Iterations</b>	<p>Number of iterations the <a href="#">IRLS</a> solver should perform.</p> <p>Default value: <b>100</b>.</p>
<b>Convergence Tolerance</b>	The integer value in this field is used as the (negative) exponent of a base-10 constant (for example, 4 evaluates to 10E-4) to check for convergence of the IRLS procedure.
<b>Regularization Parameter</b>	<p>A regularization parameter to perform a constrained optimization to overcome overfitting.</p> <p>The default value of <b>0.0</b> indicates unconstrained fit.</p>

Parameter	Description
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

The visual output includes **Summary**, **Goodness of Fit**, and **Parameter Estimates** tables.

#### Parameter Estimates

The following image shows the **Parameter Estimates** table and the associated fit statistics, with  $t$  representing the student's  $t$  statistic and  $p$  representing probability.

Results - Generalized Linear Regression Models					
Summary	Parameter	Estimate	Standard Error	t	p
Goodness of Fit	Intercept	4.18861	0.0025205	0.80474	0.44171
Parameter Estimates	humidity	0.0020283	0.047867	-1.075153	0.31028
	wind	-0.051464	0.059223	-1.49337	0.16954
	outlook_rain	-0.088441	0.057403	-0.045269	0.96488
	outlook_sunny	-0.0025985	0.20531	20.40183	7.61791e-9

#### Goodness of Fit

The following image shows the **Goodness of Fit** table.

Results - Generalized Linear Regression Models		
Summary	Statistic	Value
Goodness of Fit	Dispersion	39.35635
Parameter Estimates	Null Deviance	561.42857
	Residual df Null Deviance	13
	Deviance	354.20717
	Residual df Deviance	9
	AIC	96.96182

## Gradient Boosting Classification

A predictive method by which a series of shallow decision trees incrementally reduce prediction errors of previous trees. This method can be used for both classification and regression.



### Information at a Glance

Parameter	Description
Category	Model
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

See [Gradient Boosting](#) for more information.

### Input

A tabular data set.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	<p>Specify for the gradient boosting classification model. Select the data column to consider the dependent variable for the classification.</p> <p><b>Note:</b> Be sure that the values chosen for the <b>Independent</b> and the <b>Dependent</b> variable are different. If the user chooses a field as a <b>Dependent</b> variable that is also selected in the <b>Independent</b> variable list, an error occurs.</p> <ul style="list-style-type: none"> <li>• This is the quantity to model or predict.</li> <li>• For the Gradient Boosting Classification operator, the dependent column must be a binary categorical or continuous numerical value that has only 0 or 1 as its values. Multi-class classification is currently not supported.</li> </ul>
<b>Independent Columns</b>	<p><b>Independent Variable</b> data columns to include for the gradient boosting tree training. At least one column must be specified. Click <b>Select Values</b> to open the dialog for selecting the available columns from the input data set for analysis.</p> <p>Check/Clear the checkbox in front of the column names to select/de-select the columns.</p>
<b>Loss Function</b>	<p>Choose the loss function to use to calculate the gradient boosting trees. Choosing a different loss function leads to different interpretations for trained models and might have an effect on the final prediction accuracy as well. For mathematical details of different loss functions, see various online resources; for example, <a href="http://www.saedsayad.com/docs/gbm2.pdf">http://www.saedsayad.com/docs/gbm2.pdf</a> has good technical definitions of different loss functions.</p> <p>AdaBoost: Uses the exponential loss function.</p>

Parameter	Description
	<p>Logistic: Uses the log loss function.</p> <p>TruncatedHinge: Uses a truncated hinge loss function. It could be a bit more accurate in problems where there are a lot of outlier training samples. Intuitively speaking, this means that there is a cap on penalties imposed on grossly misclassified examples, which tend to happen with outliers. Currently, the cap is fixed but in the future, it might be exposed as a configurable parameter.</p>
<b>Number of Trees</b>	<p>Number of trees to use to train the gradient boosting classification model. Boosting uses results from previous trees to find training samples that need more attention (has larger losses). The more trees one has, the more accurate the training predictions. The validation accuracy might go down if there are too many trees. The number of trees also depends highly on the shrinkage parameter. Typically, the smaller shrinkage value means one needs more trees and vice versa.</p> <p><b>Important:</b> Performance of this operator is linear based on the number of trees selected. Therefore, if the user selects 200 trees, the runtime is twice that of 100 trees.</p> <p>Default value: <b>100</b></p>
<b>Maximum Tree Depth</b>	<p>Sets the "depth" of the tree or the maximum number of nodes it can branch out to beneath the root node. A tree stops growing any deeper if either a node becomes empty (that is, there are no more examples to split in the current node) or the depth of the tree exceeds this <b>Maximum Tree Depth</b> limit. The smaller the depth is, the shallower and weaker individual trees are. Smaller trees also often necessitate a larger number of trees.</p> <ul style="list-style-type: none"> <li>• The range of possible values is between -1 and any integer greater than 0.</li> <li>• A value of -1 represents "no bound" - the tree can take on any size or number of decision nodes until the nodes become empty.</li> </ul> <p>Default value: <b>4</b>.</p>
<b>Minimum Node</b>	Specifies the minimal size (or number of members) of a node in the tree



Parameter	Description
<b>Split Size</b>	<p>to allow a further split. If the node has fewer data members than the <b>Minimum Node Split Size</b>, it must become a leaf or end node in the tree. Similar to the <b>Maximum Tree Depth</b> parameter mentioned above, a larger node split size means that trees are smaller.</p> <ul style="list-style-type: none"> <li>The range of possible values is any integer <math>\geq 1</math>.</li> </ul> <p>Default value: <b>10</b>.</p>
<b>Bagging Rate</b>	<p>The approximate fraction of training data that is sampled (without replacement) when training each tree. For example, if this value is 0.5, the first tree is trained on a random 50% of the training data set, and the second trained on a different random 50% of the training data set, and so on. A proper value might improve model performance by mitigating overfitting.</p> <p>Default value: <b>0.5</b>.</p>
<b>Shrinkage</b>	<p>Weight given to individual trees. The smaller this number is, the more trees one might need. The larger the number is, the fewer trees one might need.</p> <p>Default value: <b>0.01</b>.</p>
<b>Fraction of Data for Training</b>	<p>Use for training the gradient boosting trees. The rest of the data set is used to measure the validation performance while the training is happening. This allows the training algorithm to estimate the optimal number of trees for validation data set accuracy.</p> <p>Default value: <b>0.8</b>.</p>
<b>Return the Optimal Number of Trees</b>	<p>When enabled, returns the optimal number of trees for the gradient boosting classification model. This is the optimal number of trees as measured against the validation data set (if the training fraction is less than 1).</p> <p>Default value: <b>yes</b>.</p>
<b>Finetune Terminal Nodes</b>	<p>Fine-tuning decision tree nodes might improve accuracy. The mathematical details of this is described in Jerome Friedman's original</p>

Parameter	Description
	<p>gradient boosting paper as 'TreeBoost'.</p> <p>.</p> <p>Default value: <b>yes</b>.</p>
<b>Maximum Number of Bins (2-65536)</b>	<p>The maximum number of bins to use during the classification. A larger number might improve accuracy in some cases, particularly if the number of unique values in categorical features exceeds the default value. For example, if the number of unique values in categorical columns exceeds this number, feature hashing is automatically performed on that column (see <a href="#">Feature Hashing</a>).</p> <p>The range of available values is <b>2-65536</b>, inclusive.</p> <p>Default value: <b>256</b>.</p>
<b>Maximum Number of Samples for Bin Finding</b>	<p>The number of samples used to determine the numeric feature discretization. A larger number might improve accuracy in some cases.</p> <p>Default value: <b>5000</b>.</p>
<b>Discretization Type</b>	<p>The method to use to group variable values into bins. If <b>Equal Width</b>, the values are divided into intervals of equal widths. If <b>Equal Frequency</b>, the values are sorted in ascending order and divided into a number of intervals that contain an equal number of sorted values.</p> <p>Default value: <b>Equal Width</b>.</p>
<b>Verbose Training</b>	<p>If <b>yes</b>, the algorithm prints many more messages to the console and the log. This can be useful when troubleshooting.</p> <p>Default value: <b>no</b>.</p>
<b>Spark Checkpoint Directory</b>	<p>The HDFS path where various intermediate Spark calculations are stored. Typically, there is no need to change this.</p> <p>Default value: @default_tmpdir/tsds_ runtime/@user_name/@flow_name.</p> <p>See <a href="#">Workflow Variables</a> for more information about the default value</p>

Parameter	Description
	variable.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

Results display the Variable Importance value, which shows the impact each variable has on the model.

## Gradient Boosting Regression

A predictive method by which a series of shallow decision trees incrementally reduce prediction errors of previous trees. This method can be used for both regression and classification.



## Information at a Glance

Parameter	Description
Category	Model
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

For more information, see [Gradient Boosting](#).

## Input

A tabular data set.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	<p>Specify the dependent column for the Gradient Boosting Regression model. Select the data column to be considered the dependent variable for the regression.</p> <div><p><b>Note:</b> Be sure that the values chosen for the independent and dependent variables are different. If the user chooses a field as a dependent variable that is also selected in the Independent variable list, an error occurs.</p><ul style="list-style-type: none"><li>• This is the quantity to model or predict.</li><li>• For the Gradient Boosting Classification operator, the dependent</li></ul></div>

Parameter	Description
	column must be a binary categorical or continuous numerical value that has only 0 or 1 as its values. Multi-class classification is currently not supported.
<b>Independent Columns</b>	<p>Select the independent variable data columns to include for the gradient boosting tree training.</p> <ul style="list-style-type: none"> <li>At least one column must be specified. Click <b>Select Values</b> to open the dialog for selecting the available columns from the input data set for analysis.</li> <li>Select or clear the box in front of the column names to select or deselect the columns.</li> </ul>
<b>Loss Function</b>	<p>The loss function used for calculating the Gradient Boosting trees. Choosing a different loss function leads to different interpretations for trained models and might have an effect on the final prediction accuracy as well. For mathematical details of different loss functions, see various online resources; for example, <a href="http://www.saedsayad.com/docs/gbm2.pdf">http://www.saedsayad.com/docs/gbm2.pdf</a> has good technical definitions of the Gaussian, Laplacian, and Poisson loss functions.</p>
<b>Number of Trees</b> *required	<p>The number of trees to use to train the Gradient Boosting Classification model. Boosting uses results from previous trees to find training samples that need more attention (have larger losses). The more trees one has, the more accurate the training predictions would be, but the validation accuracy might go down if there are too many trees. The number of trees also depends highly on the shrinkage parameter. Typically, the smaller shrinkage value means one needs more trees and vice versa.</p> <p><b>Note:</b> Performance of this operator is linear based on the number of trees selected. Therefore, if the user selects 200 trees, the runtime is twice that of 100 trees.</p> <p>Default value: <b>100</b>.</p>
<b>Maximum Tree Depth</b>	<p>Sets the depth of the tree or the maximum number of nodes it can branch out to beneath the root node. A tree stops growing any deeper if either a</p>

Parameter	Description
*required	<p>node becomes empty (that is, there are no more examples to split in the current node) or the depth of the tree exceeds this <b>Maximum Tree Depth</b> limit. The smaller the depth is, the shallower and 'weaker' individual trees are. Smaller trees also often necessitate a larger number of trees.</p> <ul style="list-style-type: none"> <li>The range of possible values is between -1 and any integer greater than 0.</li> <li>A value of -1 represents "no bound" - the tree can take on any size or number of decision nodes until the nodes become empty.</li> </ul> <p>Default value: <b>4</b>.</p>
<b>Minimum Node Split Size</b> *required	<p>Specifies the minimal size (or number of members) of a node in the tree to allow a further split. If the node has fewer data members than the <b>Minimum Node Split Size</b>, it must become a leaf or end node in the tree. Similar to the maximum depth parameter mentioned above, a larger node split size means that trees are smaller.</p> <p>The range of possible values is any integer <math>\geq 1</math>.</p> <p>Default value: <b>10</b>.</p>
<b>Bagging Rate</b> *required	<p>The approximate fraction of training data that is sampled (without replacement) when training each tree. For example, if this value is 0.5, the first tree is trained on a random 50% of the training data set, and the second is trained on a different random 50% of the training data set, and so on. A proper value might improve model performance by mitigating over fitting.</p> <p>Default value: <b>0.5</b>.</p>
<b>Shrinkage</b> *required	<p>The weight given to individual trees. The smaller this number is, the more trees one might need. The larger the number is, the fewer trees one might need.</p> <p>Default value: <b>0.01</b>.</p>
<b>Fraction of Data for Training</b>	<p>The fraction of the data used for training the Gradient Boosting trees. The rest of the data set is used to measure the validation performance while</p>

Parameter	Description
*required	<p>the training is happening. This allows the training algorithm to estimate the optimal number of trees for validation data set accuracy.</p> <p>Default value: <b>0.8</b>.</p>
<b>Return the Optimal Number of Trees</b>	<p>When enabled, returns the optimal number of trees for the Gradient Boosting Classification model. This is the optimal number of trees as measured against the validation data set (if the training fraction is less than 1).</p> <p>Default value: <b>yes</b>.</p>
<b>Finetune Terminal Nodes</b>	<p>Fine-tuning decision tree nodes might improve accuracy. The mathematical details of this are described in Jerome Friedman's original gradient boosting paper as TreeBoost.</p> <p>Default value: <b>yes</b>.</p>
<b>Maximum Number of Bins (2-65536)</b>	<p>The maximum number of bins to use during the classification. A larger number might improve accuracy in some cases, particularly if the number of unique values in categorical features exceed the default value. For example, if the number of unique values in categorical columns exceeds this number, <a href="#">feature hashing</a> is automatically performed on that column.</p> <p>The range of available values is 2-65536, inclusive.</p> <p>Default value: <b>256</b>.</p>
<b>Maximum Number of Samples for Bin Finding</b>	<p>The number of samples used to determine the numeric feature discretization. A larger number might improve accuracy in some cases.</p> <p>Default value: <b>5000</b>.</p>
*required	
<b>Discretization Type</b>	<p>The method to use to group variable values into bins. If <b>Equal Width</b> (the default), the values are divided into intervals of equal widths. If <b>Equal Frequency</b>, the values are sorted in ascending order and divided into a number of intervals that contain an equal number of sorted values.</p>

Parameter	Description
<b>Repartition Data</b>	<p>A Spark operation that might improve training performance (speed) in certain cases. Typically, there is no need to change this parameter.</p> <p>Default value: <b>no</b>.</p>
<b>Verbose Training</b>	<p>If set to <b>yes</b>, the algorithm prints many more messages to the console and log, which can be useful when troubleshooting.</p> <p>Default value: <b>no</b>.</p>
<b>Spark Checkpoint Directory</b> *required	<p>An HDFS path where various intermediate Spark calculations are stored. Typically, there is no need to change this parameter.</p> <p>Default value: <b>@default_tempdir/tsds_runtime/@user_name/@flow_name</b>.</p>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## K-Means (DB)

K-Means configuration is a data set that contains the various attribute values of the data members to use as clustering or partitioning criteria.



### Information at a Glance

Parameter	Description
Category	Model



Parameter	Description
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a



**Note:** The K-Means (DB) operator is for database data only. For Hadoop data, use the [K-Means \(HD\)](#) operator.

For more information about using K-Means, see [Cluster Analysis Using K-Means](#).

## Algorithm

The K-Means Operator algorithm works to make the objects within a cluster "similar" to one another and "dissimilar" to members of the other clusters in terms of their attributes. In order to achieve this, K-Means employs a centroid-based partitioning technique that uses the centroid of a cluster to represent that cluster. Conceptually, the centroid of a cluster is its center point.

The TIBCO Data Science - Team Studio K-Means Operator defines the centroid as the mean of the attribute values for members within the cluster. The difference between a cluster member and the centroid is used to determine the quality of the cluster model. Note: the distance can be measured in different ways, with the default being the Euclidean distance between two points. Specifically, the algorithm calculates a within-cluster variation value, which is the sum of the squared distance between all members of the cluster and the centroid of the cluster.

The overall K-Means algorithm's objective is to make the  $k$  clusters for a data set to be as compact (smallest within-cluster variation) and separate (minimal overlap) from each other as possible. In other words, the algorithm aims for high intra-cluster similarity and low inter-cluster similarity, across the various attribute dimensions included in the model.

The K-Means algorithm works as follows.

1.  $k$  random points from the data set are chosen as the initial centroids of the  $k$  clusters.

2.  $k$  clusters are created by associating each observation to the nearest centroid.
3. The new centroids are calculated for the clusters; determine whether centroid values change coordinates.
4. Steps 2 and 3 are repeated until convergence (when the centroid values do not change) or a specified termination criterion is met.

## Input

Non-numeric variables in the input data set must first be transformed and scaled or normalized before the clustering can take place. Numeric variables can be normalized as well. Depending on the chosen transformations, normalizations, and distance calculations, certain variables might dominate clustering results or might be completely ignored.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	<p>Specifies the various attributes (that is, data columns) to consider during the cluster analysis. At least one column must be specified.</p> <p>Click <b>Columns</b> to open the dialog for selecting the available columns from the input data set for analysis.</p> <p>See <a href="#">Select Columns dialog</a>.</p>
<b>Number of Clusters</b>	<p><math>k</math> specifies the number of clusters to create during the cluster analysis process. If the number of clusters, <math>k</math>, for the k-means algorithm is not chosen to match the natural structure of the data, the results are not ideal. The proper way to alleviate this is to experiment with different values for <math>k</math>. In principle, the best <math>k</math> value exhibits the smallest intra-cluster distances and largest inter-cluster distances.</p> <p>Therefore, the modeler might experiment with various values for <math>k</math> based on the cluster results. An increase in clusters possibly is needed, for example, if the analysis does not converge or if the groups are too</p>

Parameter	Description
	<p>overlapping or spread out.</p> <p>Default value: <b>3</b>.</p>
<b>Number of Random Starts</b>	<p>Specifies the maximum number of runs performed for the k-means method, with random initiations. Increasing this value also increases processing time for the algorithm because it is, effectively, increasing the number of times the k-means algorithm is run overall. Therefore, while increasing the <b>Number of Random Starts</b> tends to provide better cluster results, modelers should balance length of processing time for the analysis against overall quality of the model.</p> <p>Default value: <b>1</b>.</p>
<b>Maximum Optimization Steps</b>	<p>Specifies the maximum number of iterations performed for one run of the k-means algorithm.</p> <div> <p><b>Note:</b> A warning text tab is displayed in the results if the K-Means analysis has not converged within the <b>Maximum Optimization Steps</b>. The modeler must try increasing this value if that happens.</p> </div> <p>Default value: <b>10</b>.</p>
<b>Tolerance</b>	<p>This value is similar to the epsilon value on Logistic Regression and Linear Regression. The lower the value is set (closer to 0), the stricter we are about when we say the analysis has converged. A smaller number results in more iterations of the algorithm, but it is still capped by the iteration limit.</p> <p>Default value: <b>1.0E-4</b>.</p>
<b>Working Schema</b>	<p>The defined schema is used as a temporary location to write a view and then delete it when the operator finishes running. You need write access to this schema.</p> <p>Default value: <b>@default_schema</b>.</p>

## Output

### Visual Output

For data sets with 1000 or fewer columns, the K-Means Operator results are displayed across multiple sections.

**Note:** To learn more about the visualization available in this operator, see [Explore Visual Results](#).

- **Center Points** - The Center Point results section displays a results table with the various mean distance from centroid measurements for each of the Variables, per cluster:

Results - K-Means								
Center Points	Cluster	times90dayslate	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslate_2years	srsdliqncy
<a href="#">Cluster Profiles</a>	0	0	0.71	0.73	7	8,070.3	0	0
<a href="#">Training Runs</a>	1	0	0.3	0.76	5	2,312.31	0	0
<a href="#">Labelled Samples</a>	2	0	0.29	0.36	4	3,156.83	0	0
<a href="#">Scatter Plots</a>								

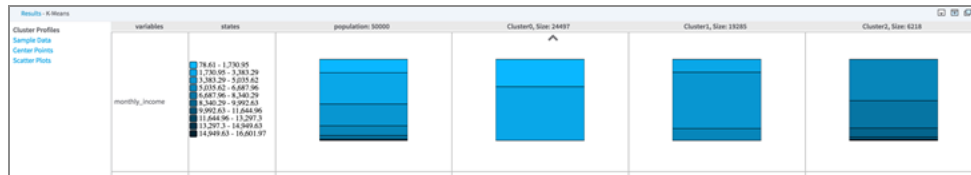
This can help compare a specific variable value from one cluster against another.

**Note:** The K-Means Center Point data can be used as a succinct and precise definition of the characteristics of the cluster.

- **Cluster Profiles** - The Cluster Profiles results section provides an overall sense of how the data is partitioned into clusters.

It displays a table with a row for each attribute (that is, variable) used in the cluster analysis. For each variable, the following results are provided:

- **States:** specify how the variable's numeric value was split or stratified into the specified Number of Splits groups. Note: this acts as a "legend" for understanding how each cluster's members are distributed across the possible value ranges.
- **Population:** specifies the number of overall data rows analyzed and shows the overall breakdown of the data set across the states for each particular variable.
- **Cluster 0...K:** for each of the K clusters created, a column shows the breakdown of the cluster members across the various states for each particular variable. Note: the size of each cluster (that is, number of members) is displayed at the top of the **Cluster** columns.



**Note:** When analyzing the K-Means Cluster Profile results tab, the modeler should look for the following.

- Each variable having different distribution patterns per cluster. This shows that, for that variable, the clusters have distinct characteristics and therefore, implies a successful K-Means analysis.
  - Distortions in the cluster size relative to one another might highlight interesting conclusions, such as showing a dominant group for targeting specific marketing.
  - Clusters that have distinct characteristics might help target specific conditions for research.
- **Training Runs** - The **Run** section shows what run of the algorithm is displayed. If you only choose one run, only one row is visible here.

Run	Average Distance	Converged	Chosen
1	711.66	false	
2	636.032	false	yes

**Average Distance** refers to the average Euclidean distance for each sample to the centroid in its cluster.

A run has 'converged' if two iterations produce practically the same model (the difference is less than the convergence threshold, or tolerance). If it doesn't converge, the algorithm continues up to the allotted amount of iterations.

We choose the model with the smallest **Average Distance** to show you results.

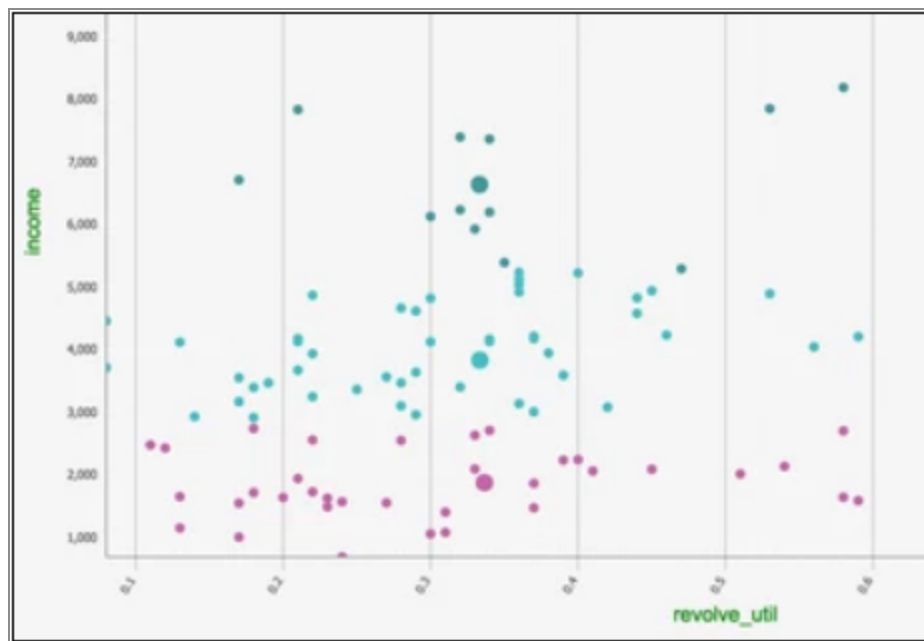
- **Scatter Plots** - The **Cluster** results tab displays a Cluster graph, which is a

visualization of each cluster's member values based on two of the variable dimensions used for the K-Means analysis.

Although there is typically some overlap between members of the clusters, the goal is to minimize the overlap of clusters, as illustrated in the example above. For a perfect cluster analysis model, there would be zero overlap between the clusters for each variable analyzed.

The output can only be displayed two dimensions at a time. Therefore, the modeler needs to review all the possible clustering diagrams in order to get an overall assessment of which attribute dimensions have the greatest influence on the clustering. Note: the **Cluster Profiles** results tab provides a quick sense of which variables have the most unique distribution profile across clusters, so those variables would be good ones to further analyze in this **Cluster** graph section.

Cluster graphing can be toggled on/off per cluster. Therefore, the graph can be viewed showing one cluster at a time, which helps understand just the spread of members per cluster and visually see their distance from their center, as in the following example, which shows three visually distinct clusters with a clear separation of the three center points for each cluster.



A lot of "cluster overlap" for two variables might indicate that they are not as significant in the cluster analysis, or that there is not much variation of the overall

population for those particular variables.

Another cause of "cluster overlap" might be that the variable values were not appropriately normalized before the analysis was run.

### Data Output

A model output that must be connected to a Predictor operator to produce a data set output. It can also be exported in PFA and AM format along with PMML.

## K-Means (HD)

K-Means configuration is a data set that contains the various attribute values of the data members to use as clustering or partitioning criteria.



### Information at a Glance

Parameter	Description
Category	Model
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce, Spark



**Note:** The K-Means (HD) operator is for Hadoop data only. For database data, use the [K-Means \(DB\)](#) operator.

For more information about using K-Means, see [Cluster Analysis Using K-Means](#).

## Algorithm

The K-Means Operator algorithm works to make the objects within a cluster "similar" to one another and "dissimilar" to members of the other clusters in terms of their attributes. In order to achieve this, K-Means employs a centroid-based partitioning technique that uses the centroid of a cluster to represent that cluster. Conceptually, the centroid of a cluster is its center point.

The TIBCO Data Science - Team Studio K-Means Operator defines the centroid as the mean of the attribute values for members within the cluster. The difference between a cluster member and the centroid is used to determine the quality of the cluster model. Note: the distance can be measured in different ways, with the default being the Euclidean distance between two points. Specifically, the algorithm calculates a within-cluster variation value, which is the sum of the squared distance between all members of the cluster and the centroid of the cluster.

The overall K-Means algorithm's objective is to make the  $k$  clusters for a data set to be as compact (smallest within-cluster variation) and separate (minimal overlap) from each other as possible. In other words, the algorithm aims for high intracluster similarity and low intercluster similarity, across the various attribute dimensions included in the model.

The K-Means algorithm works as follows.

1.  $k$  random points from the data set are chosen as the initial centroids of the  $k$  clusters.
2.  $k$  clusters are created by associating each observation to the nearest centroid.
3. The new centroids are calculated for the clusters; determine whether centroid values change coordinates.
4. Steps 2 and 3 are repeated until convergence (when the centroid values do not change) or a specified termination criterion is met.

## Input

Non-numeric variables in the input data set must first be transformed and scaled or normalized before the clustering can take place. Numeric variables can be normalized as well. Depending on the chosen transformations, normalizations, and distance calculations, certain variables might dominate clustering results or might be completely ignored.



## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	Specifies the various attributes (that is, data columns) to consider during the cluster analysis. At least one column must be specified.  Click <b>Columns</b> to open the dialog for selecting the available columns from the input data set for analysis.  See <a href="#">Select Columns dialog</a> .
<b>Number of Clusters</b>	The number of clusters to determine from the data for the k-means analysis.  Default value: <b>3</b> .
<b>Maximum Optimization Steps</b>	Specifies the maximal number of iterations performed for one run of the k-means algorithm.  A warning message displays in the results if the k-means analysis has not converged within the <b>Maximum Optimization Steps</b> . If this occurs, try increasing this value.  Default value: <b>10</b> .
<b>Tolerance</b>	This value is similar to the epsilon value on Logistic Regression and Linear Regression. The lower the value is set (closer to 0), the stricter we are about when we say the analysis has converged. A smaller number results in more iterations of the algorithm, but it is still capped by the iteration limit.  Default value: <b>1.0E-4</b> .
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.
<b>Advanced Spark Settings</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> </ul>

Parameter	Description
<b>Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

For data sets with 1000 or fewer columns, the K-Means Operator results are displayed across multiple sections.

**Note:** To learn more about the visualization available in this operator, see [Explore Visual Results](#).

- **Center Points** - Displays a results table with the various mean distance from centroid measurements for each of the Variables, per cluster:

Results - K-Means								
<a href="#">Center Points</a> <a href="#">Cluster Profiles</a> <a href="#">Training Runs</a> <a href="#">Labelled Samples</a> <a href="#">Scatter Plots</a>	Cluster	times90dayslate	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslate_2years	srsdlqncy
	0	0	0.71	0.73	7	8,070.3	0	0
	1	0	0.3	0.76	5	2,312.31	0	0
	2	0	0.29	0.36	4	3,156.83	0	0

This can help compare a specific variable value from one cluster against another.

**Note:** The K-Means Center Point data can be used as a succinct and precise definition of the characteristics of the cluster.

- **Cluster Profiles** - Provides an overall sense of how the data is partitioned into clusters.

It displays a table with a row for each attribute (that is, variable) used in the cluster analysis. For each variable, the following results are provided:

- **States:** specify how the variable's numeric value was split or stratified into the specified Number of Splits groups. Note: this acts as a "legend" for understanding how each cluster's members are distributed across the

possible value ranges.

- **Population:** specifies the number of overall data rows analyzed and shows the overall breakdown of the data set across the states for each particular variable.
- **Cluster 0...K:** for each of the K clusters created, a column shows the breakdown of the cluster members across the various states for each particular variable. Note: the size of each cluster (that is, number of members) is displayed at the top of the **Cluster** columns.



**Note:** When analyzing the K-Means Cluster Profile results tab, the modeler should look for:

- Each variable having different distribution patterns per cluster. This shows that, for that variable, the clusters have distinct characteristics and therefore, implies a successful K-Means analysis.
  - Distortions in the cluster size relative to one another might highlight interesting conclusions, such as showing a dominant group for targeting specific marketing.
  - Clusters that have distinct characteristics might help target specific conditions for research.
- **Training Runs** - The **Run** section shows what run of the algorithm is displayed. If you only choose one run, only one row is visible here.

Run	Average Distance	Converged	Chosen
1	711.66	false	
2	636.032	false	yes

**Average Distance** refers to the average Euclidean distance for each sample to the centroid in its cluster.

A run has 'converged' if two iterations produce practically the same model (the difference is less than the convergence threshold, or tolerance). If it doesn't converge, the algorithm continues up to the allotted amount of iterations.

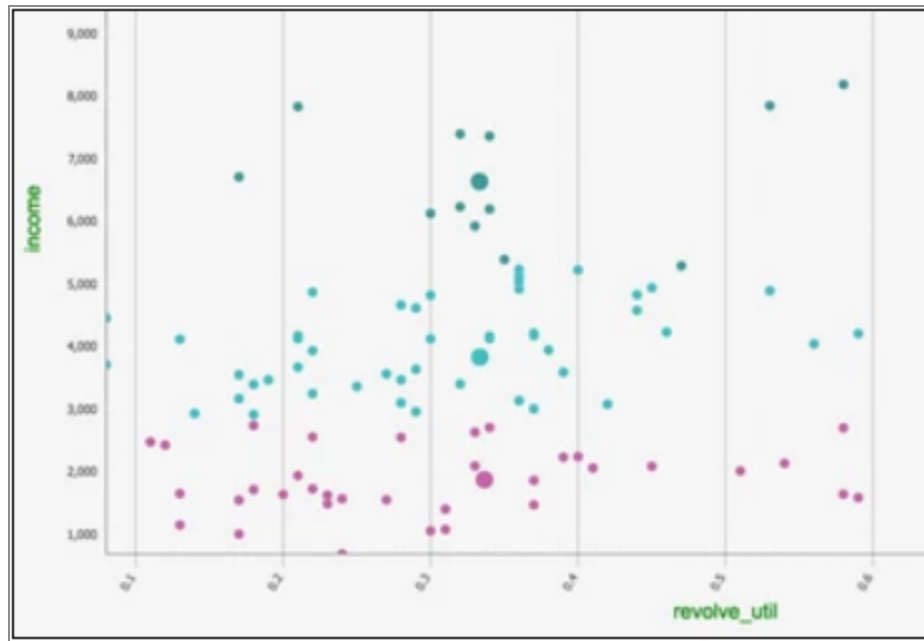
We choose the model with the smallest **Average Distance** to show you results.

- **Scatter Plots** - The **Cluster** results tab displays a Cluster graph, which is a visualization of each cluster's member values based on two of the variable dimensions used for the K-Means analysis.

Although there is typically some overlap between members of the clusters, the goal is to minimize the overlap of clusters, as illustrated in the example above. For a perfect cluster analysis model, there would be zero overlap between the clusters for each variable analyzed.

The output can only be displayed two dimensions at a time. Therefore, the modeler needs to review all the possible clustering diagrams in order to get an overall assessment of which attribute dimensions have the greatest influence on the clustering. Note: the **Cluster Profiles** results tab provides a quick sense of which variables have the most unique distribution profile across clusters, so those variables would be good ones to further analyze in this **Cluster** graph section.

Cluster graphing can be toggled on/off per cluster. Therefore, the graph can be viewed showing one cluster at a time, which helps understand just the spread of members per cluster and visually see their distance from their center, as in the following example, which shows three visually distinct clusters with a clear separation of the three center points for each cluster.



A lot of "cluster overlap" for two variables might indicate that they are not as significant in the cluster analysis, or that there is not much variation of the overall population for those particular variables.

Another cause of "cluster overlap" might be that the variable values were not appropriately normalized before the analysis was run.

### Data Output

A model output that must be connected to a Predictor operator to produce a data set output. It can also be exported in PFA and AM format along with PMML.

## K-Means Clustering - MADlib

TIBCO Data Science - Team Studio supports the MADlib K-Means Clustering model implementation.



## Information at a Glance

Parameter	Description
Category	Model
Data source type	DB
Send output to other operators	No
Data processing tool	n/a

## Algorithm

Clustering refers to the problem of partitioning a set of objects according to some problem-dependent measure of similarity. In the k-means variant, one is given  $n$  points, and the goal is to position  $k$  centroids so that the sum of distances between each point and its closest centroid is minimized. Each centroid represents a cluster that consists of all points to which this centroid is closest.

In the most common case, the distance measure used is the square of the Euclidean distance.

This problem is computationally difficult (NP-hard), yet the local-search heuristic proposed by Lloyd [1] performs reasonably well in practice. In fact, it is so ubiquitous today that it is often referred to as the standard algorithm or even just the k-means algorithm. It works as follows:

1. Seed the  $k$  centroids.
2. Repeat until convergence:
  - a. Assign each point to its closest centroid.
  - b. Move each centroid to a position that minimizes the sum of distances in this cluster.
3. Convergence is achieved when no points change their assignments during step 2a.

Since the objective function decreases in every step, this algorithm is guaranteed to converge to a local optimum.

More information can be found in the [Official MADlib documentation](#) page.

## Input

A data set that contains an array column with the various attribute values to be used as clustering criteria.

- Non-numeric variables in the input data set must first be transformed and scaled or normalized before the clustering can take place.
- Numeric variables can be normalized as well.
- Depending on the chosen transformations, normalizations, and distance calculations, certain variables may dominate clustering results or may be completely ignored.

## Restrictions

Output can be sent to a K-Means Predictor operator only.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>MADlib Schema Name</b>	<p>The schema where MADlib is installed in the database. MADlib must be installed in the same database as the input data set.</p> <p>If a madlib schema exists in the database, this parameter defaults to madlib.</p>
<b>Model Output Schema Name</b>	The name of the schema where the output is stored.
<b>Model Output Table Name</b>	<p>The name of the table that is created to store the model. The model output table stores:</p> <p>centroids   objective_fn   frac_reassigned   num_iterations</p> <p>For more information, see the <a href="#">official MADlib k-means documentation</a>.</p>
<b>Drop If Exists</b>	<ul style="list-style-type: none"> <li>• If <b>Yes</b> (the default), drop the existing table of the same name and</li> </ul>

Parameter	Description
	<p>create a new one.</p> <ul style="list-style-type: none"> <li>• If <b>No</b>, stop the flow and alert the user that an error has occurred.</li> </ul>
<b>Centroid Seeding</b>	<p>Centroid seeding can be done in three ways:</p> <ul style="list-style-type: none"> <li>• <b>random</b> (the default) - Selects the initial centroids randomly from the input data set.</li> <li>• <b>distributed</b> - Selects the initial centroids through the k-Means Plus Plus \[1\] algorithm, which strives to pick centroids far apart from one another.</li> <li>• <b>user-defined</b> - You specify a centroid source input table (<b>Centroid Source</b>) and its <b>Centroid Column</b> that has the initial centroid positions.</li> </ul>
<b>Centroid Source</b>	If <b>Centroid Seeding</b> is set to user-defined, set the input table for the centroid source.
<b>Centroid Column</b>	If <b>Centroid Seeding</b> is set to user-defined, set the column in the input table specified by <b>Centroid Source</b> for the centroid column.
<b>Points Column</b>	<p>The <b>Points</b> column in the input data set contains the array of attributes for each point.</p> <p>This parameter must be an array type.</p>
<b>K</b>	<p>Specifies the number of clusters to create during the cluster analysis process.</p> <ul style="list-style-type: none"> <li>• The default value is <b>2</b>.</li> </ul> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p><b>Note:</b> If the number of clusters, <i>K</i>, for the K-means algorithm is not chosen to match the natural structure of the data, the results are not ideal. The proper way to alleviate this is to experiment with different values for <i>K</i>. In principle, the best <i>K</i> value exhibits the smallest intra-cluster distances and largest inter-cluster distances.</p> </div> <ul style="list-style-type: none"> <li>• Therefore, the modeler might want to experiment with various values</li> </ul>



Parameter	Description
	for $K$ based on the cluster results. Perhaps an increase in clusters is needed, for example, if the analysis does not converge or if the groups are too overlapping or spread out.
<b>Distance Function</b>	<p>Calculates the difference between the cluster member's values from the cluster's centroid (mean) value. The distance can be calculated in the following different ways:</p> <p><b>Cosine</b> - Measures the cosine of the angle between two vectors. For <math>n</math> dimensional vectors <math>\mathbf{u} = (u_1, u_2, \dots, u_n)</math> and <math>\mathbf{v} = (v_1, v_2, \dots, v_n)</math>, it is calculated using the dot product formula as:</p> $\frac{\mathbf{u} \cdot \mathbf{v}}{\ \mathbf{u}\  \ \mathbf{v}\ } = \frac{\sum u_i v_i}{\sqrt{\sum u_i^2} \sqrt{\sum v_i^2}}.$ <p><b>Euclidean</b> - The square root of the sum of the squares of distances along each attribute axes. For <math>n</math> dimensional vectors <math>\mathbf{u} = (u_1, u_2, \dots, u_n)</math> and <math>\mathbf{v} = (v_1, v_2, \dots, v_n)</math>, it is calculated as:</p> $\sqrt{\sum_{i=1}^n (u_i - v_i)^2}.$ <p><b>Manhattan</b> - The Manhattan, or taxicab, metric measures the distance between two points when traveling parallel to the axes. For <math>n</math> dimensional vectors <math>\mathbf{u} = (u_1, u_2, \dots, u_n)</math> and <math>\mathbf{v} = (v_1, v_2, \dots, v_n)</math>, it is given by</p> $\sum_{i=1}^n  u_i - v_i .$ <p><b>Squared Euclidean</b> (the default) - The default method for calculating the straight line distance between two points. It is the sum of the squares of distances along each attribute axes. For <math>n</math> dimensional vectors <math>\mathbf{u} = (u_1, u_2, \dots, u_n)</math> and <math>\mathbf{v} = (v_1, v_2, \dots, v_n)</math>, it is calculated as</p>

Parameter	Description
	$\sum_{i=1}^n (u_i - v_i)^2.$ <p><b>Tanimoto</b> - Measures dissimilarity between sample sets. It is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union. As with Dice similarity, 0-1 vectors are used to represent sets. Then the Jaccard Similarity of sets A and B represented by set-vectors <math>(a_1, a_2, \dots, a_n)</math> and <math>(b_1, b_2, \dots, b_n)</math> is given by:</p> $\frac{ A \cap B }{ A \cup B } = \frac{\sum_{i=1}^n a_i b_i}{\sum_{i=1}^n a_i + \sum_{i=1}^n b_i - \sum_{i=1}^n a_i b_i}.$ <p><b>User defined</b> - Specified by the user. See <b>User-Defined Distance</b>.</p>
<b>User-Defined Distance</b>	<p>If, for <b>Distance Function</b>, you specify <b>User-defined</b>, provide the function.</p> <p>The modeler might try to start with the default <b>Squared Euclidean</b> method, then experiment with the various other calculation methods to determine whether the cluster results seem more intuitive or provide more business insight.</p>
<b>Aggregation of Centroid</b>	<p>Defines how cluster centroid positions should be calculated using the points assigned to the same cluster. The options are <b>average</b> (the default) or <b>normalized average</b>.</p>
<b>Maximum Iterations</b>	<p>The number of iterations required before computation stops. When the number of iterations is greater than the <b>Maximum Iterations</b>, or when the fraction of reassigned points is below the <b>Minimum Fraction Reassigned</b>, computation stops.</p>
<b>Mini Frac Reassigned</b>	<p>The minimum fraction of reassigned points required before computation stops. When the number of iterations is greater than the <b>Maximum Iterations</b>, or when the fraction of reassigned points is below the value of <b>Mini Frac Reassigned</b>, computation stops.</p>

## Output

### Visual Output

The K-Means Clustering (MADlib) operator outputs **Center Point** and **Cluster** tabs.

**Note:** To get the assignment of points to each cluster, run the K-Means Predictor (MADlib) operator.

The **Center Point** results tab displays the cluster centroid positions, objective function value, fraction of reassigned points in the last iteration, number of total iterations, and the simple silhouette coefficient - a popular method to assess the quality of the clustering (the closer to 1, the better the clustering).

- The **Cluster** results tab displays a graph of the final centroids determined from the K-Means analysis.
- The output can be displayed only two dimensions at a time.

Result Log   testkmeans_float   K-means (MADlib)					
Center Point   Cluster					
CenterPoint0	CenterPoint1	Objective Function	Frac Reassigned	Number Of Iteration	Silhouette Coefficient
{1.0, 2.0, 3.0}	{2.0, 2.0, 2.0}	6.0	0.0	2	0.7

### Data Output

You can send output only to a [K-Means Predictor - MADlib](#) operator.

## Additional Notes

### References

[1] Lloyd, Stuart: Least squares quantization in PCM. Technical Note, Bell Laboratories. Published much later in: IEEE Transactions on Information Theory 28(2), pp. 128-137. 1982.

[1] David Arthur, Sergei Vassilvitskii: k-means++: the advantages of careful seeding, Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'07), pp. 1027-1035, <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

## Linear Regression (HD)

Use the Linear Regression operator to fit a trend line to an observed data set, in which one of the data values - the dependent variable - is linearly dependent on the value of the other causal data values or variables - the independent variables.



### Information at a Glance

Parameter	Description
Category	Model
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce, Spark



**Note:** The Linear Regression (HD) operator is for Hadoop data only. For database data, use the [Linear Regression \(DB\)](#) operator.

For more information about using linear regression, see [Fitting a Trend Line for Linearly Dependent Data Values](#).

## Algorithm

The TIBCO Data Science - Team Studio Linear Regression operator applies a Multivariate Linear Regression (MLR) algorithm to the input data set. For MLR, a Regularization Penalty Parameter that can be applied in order to prevent the chances of over-fitting the model.

This Linear Regression operator implements either Ordinary or Elastic Net Linear Regression.

The Ordinary Regression algorithm uses the Ordinary Least Squares (OLS) method of regression analysis, meaning that the model is fit such that the sum-of-squares of differences of observed and predicted values is minimized.

The Elastic Net Regression algorithm supports the Ordinary Least Squares (OLS) method of linear regression, along with implementing the Elastic Net Objective Function to support either Lasso(L1) or Ridge (L2) penalty cost functions.

This Linear Regression operator implements Ordinary Linear Regression with an option for the Elastic Net Regularization feature to avoid over-fitting a model with too many variables.

## Input

A data set that contains the dependent and independent variables for modeling.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	<p>The dependent column specified for the regression. This is the quantity to model or predict. The list of the available data columns for the Regression operator are displayed. Select the data column to consider the dependent variable for the regression.</p> <p>The <b>Dependent Column</b> should be a numerical data type.</p>
<b>Maximum Number of</b>	The total number of iterations that are processed before the algorithm stops, if the coefficients do not converge or show relevance.

Parameter	Description
<b>Iterations</b>	<ul style="list-style-type: none"> <li>• <b>Maximum Number of Iterations</b> must be a value <math>\geq 1</math>.</li> </ul> <p>Default value: <b>20</b>.</p>
<b>Tolerance</b>	<p>Maximum allowed error value for the calculation method. When the error is smaller than this value, the linear regression model training stops.</p> <p>Default value: <b>0.000001</b>.</p>
<b>Columns</b>	<p>Click <b>Select Columns</b> to select the available columns from the input data set for analysis.</p> <p>For a linear regression, select the independent variable data columns for the regression analysis or model training.</p> <p>You must select at least one column or one interaction variable.</p>
<b>Interaction Parameters</b>	<p>Enables selecting available independent variables, where those data parameters might have a combined effect on the dependent variable. See <a href="#">Interaction Parameters dialog</a> for detailed information.</p>
<b>Number of Cross Validation</b>	<p>Gives an option for either 5 or 10 cross-validation steps for the linear regression.</p> <p>This parameter applies only if <b>Type of Linear Regression</b> is set to <b>Elastic Net Penalty</b>.</p> <p>Cross validation is a technique for testing the model during the training phase by using a small amount of the data as "test" data. Cross validation helps avoid over-fitting a model and provides insight on how the model generalizes to an independent data set. The <b>Number of Cross Validation</b> steps specifies how many times to section off the data for testing.</p> <p>The higher the number of steps, the more accurate the calculated model error is (although the model processing time is greater).</p> <p>Default value: <b>5</b>.</p>
<b>Type of Linear Regression</b>	<p>Determines whether to perform an <b>Ordinary</b> linear regression or a linear regression with <b>Elastic Net Penalty</b> applied.</p>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Ordinary</b> implements the standard Ordinary Least Squares (OLS) algorithm.</li> <li>• <b>Elastic Net Penalty</b> (the default) implements the Ordinary Least Squares (OLS) algorithm along with the Elastic Net coefficient constraints applied to reduce over-fitting of the data. It automatically selects a Penalizing Parameter (lambda) for the first run.</li> </ul> <p>The <b>Elastic Net Penalty</b> option is helpful when the number of independent variables in the model is very large compared to the number of data observations.</p>
<b>Use Intercept?</b>	<p>Provides the option to calculate the Intercept value, <math>\beta_0</math>, for the linear regression.</p> <p>This parameter applies only if <b>Type of Linear Regression</b> is set to <b>Elastic Net Penalty</b>.</p> <p>In general, this should always be used unless the data has already been normalized.</p> <p>Default value: <b>Yes</b>.</p>
<b>Penalizing Parameter (<math>\lambda</math>)</b>	<p>Represents an optimization parameter for linear regression. It implements regularization of the trade-off between model bias (significance of loss function) and the regularization portion of the minimization function (variance of the regression correlation coefficients).</p> <p>The value can be any number 0 or greater, with the default value of 0 (for no penalty).</p> <p>The higher the Lambda, the lower chance of over-fitting with too many redundant variables. Over-fitting is the situation where the model does a good job "learning" or converges to a low error for the training data but does not do as well for new, non-training data. In general, use regularization to avoid overfitting, so multiple models with different lambda should be trained, and the model with the smallest testing error should be chosen. The lambda value should be greater than 0.</p>

Parameter	Description
	<p>For linear regression equations, you can use the cross-validation process to pick the best Lambda value.</p> <p>If you choose a cross validation number, <b>Penalizing Parameter</b> is disabled. The number of cross validations in the results suggest the initial value of lambda.</p> <p>For more information, see <a href="#">Fitting a Trend Line for Linearly Dependent Data Values</a>.</p>
<b>Elastic Parameter (<math>\alpha</math>)</b>	<p>A constant value between 0 and 1 that controls the degree of the mix between L1 (Lasso) and L2 (Ridge) regularization. Specifically, it is the <math>\alpha</math> parameter in the Elastic Net Regularization loss function given by:</p> $\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \left[ \frac{(1-\alpha)}{2} \beta_j^2 + \alpha  \beta_j  \right]$ <p>The <b>Elastic Parameter</b> combines the effects of both the Ridge and Lasso penalty constraints. Both types of penalties shrink the values of the correlation coefficients.</p> <p>This parameter applies only if <b>Type of Linear Regression</b> is set to <b>Elastic Net Penalty</b>.</p> <ul style="list-style-type: none"> <li>When the Elastic Parameter <math>\alpha = 1</math>, it becomes pure L1 Regularization (Lasso). <ul style="list-style-type: none"> <li>A Lasso constraint tends to remove redundant variables and therefore results in a sparse coefficient model, which is useful when there is high dimensionality in the data.</li> </ul> </li> <li>When the Elastic Parameter <math>\alpha = 0</math>, it becomes pure L2 Regularization (Ridge). <ul style="list-style-type: none"> <li>A Ridge constraint tends to make the variables have similar correlation coefficients and is useful when the data has few variables, or low dimensionality.</li> </ul> </li> <li>When the Elastic Parameter <math>\alpha</math> is in between 0~1, it implements a mix of both L1 (Lasso) and L2 (Ridge) constraints on the coefficients.</li> </ul>



Parameter	Description
	<ul style="list-style-type: none"> <li>The default value is <b>1</b> (L1 or Lasso Regularization). A value of <b>0.5</b> implements a compromise between the Lasso and Ridge constraints.</li> </ul>
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li><b>Yes</b> specifies using the default Spark optimization settings.</li> <li><b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

#### Ordinary Linear Regression Output

Because data scientists expect model prediction errors to be unstructured and normally distributed, the Residual Plot and Q-Q Plot together are important linear regression diagnostic tools, in conjunction with R2, Coefficient and P-value summary statistics.

The remaining visual output consists of **Summary**, **Data**, **Residual Plot**, and **Q-Q Plot**.

#### Elastic Net Penalty Output

An additional output Cross Validation Plot tab is displayed when an Elastic Net Penalty linear regression is implemented.

- **Summary**
- **Data**
- **Residual Plot** (optional)
- **Q-Q Plot** (optional)
- **Cross Validation Plot**

## Summary

The **Summary** output displays the details of the derived linear regression model's Equation and Correlation Coefficient values along with the R2 and Standard Error statistical values.

Summary	turnout =
Data	- 0.0022 * msa
Residual Plot	+ 1.0E-4 * popdensity
Q-Q Plot	+ 0.0 * pop
	+ 0.0626 * popchange
	+ 0.5235 * age6574
	+ 1.6894 * age75
	+ 0.0 * crime
	+ 0.1694 * college
	+ 5.0E-4 * income
	+ 0.0518 * farm
	+ 0.1754 * democrat
	+ 0.0953 * republican
	+ 0.4265 * Perot
	+ 0.2858 * white
	+ 0.2649 * black

The derived linear regression model is shown as a mathematical equation linking the Dependent Variable (Y) to the independent variables (X1, X2, etc.). It includes the scaling or Coefficient values ( $\beta_1$ ,  $\beta_2$ , etc.) associated with each independent variable in the model.

**Note:** The resulting linear equation is expressed in the form of  $Y = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \dots$

The following overall model statistical fit numbers are displayed.

- R2: it is called the multiple correlation coefficient of the model, or the Coefficient of Multiple Determination. It represents the fraction of the total Dependent Variable (Y) variance explained by the regression analysis, with 0

meaning 0% explanation of Y variance and 1 meaning 100% accurate fit or prediction capability.

**i Note:** in general, an R2 value greater than .8 is considered a good model. However, this value is relative and in some situations just getting an improved R2 from .5 to .6, for example, would be beneficial.

- S: represents the standard error per model (often also denoted by SE). It is a measure of the average amount that the regression model equation over-predicts or under-predicts.

The rule of thumb data scientists use is that 60% of the model predictions are within +/- 1 SE and 90% are within +/- 2 SEs.

For example, if a linear regression model predicts the quality of the wine on a scale between 1 and 10 and the SE is .6 per model prediction, a prediction of Quality=8 means the true value is 90% likely to be within 2\*.6 of the predicted 8 value (that is, the real Quality value is likely between 6.8 and 9.2).

In summary, the higher the R2 and the lower the SE, the more accurate the linear regression model predictions are likely to be.

## Data

The **Data** results are a table that contains the model coefficients and statistical fit numbers for each independent variable in the model.

Summary	Attribute	Coefficient	SE	T-statistics	P-value
<b>Data</b>	farm	0.3934	0.2087	1.8849	0.0621
Residual Plot	democrat	-1.0151	0.7436	-1.3651	0.175
Q-Q Plot	republican	-1.1496	0.7465	-1.54	0.1264
	Perot	-0.9154	0.7611	-1.2028	0.2317
	white	0.3315	0.0898	3.6898	0.0004

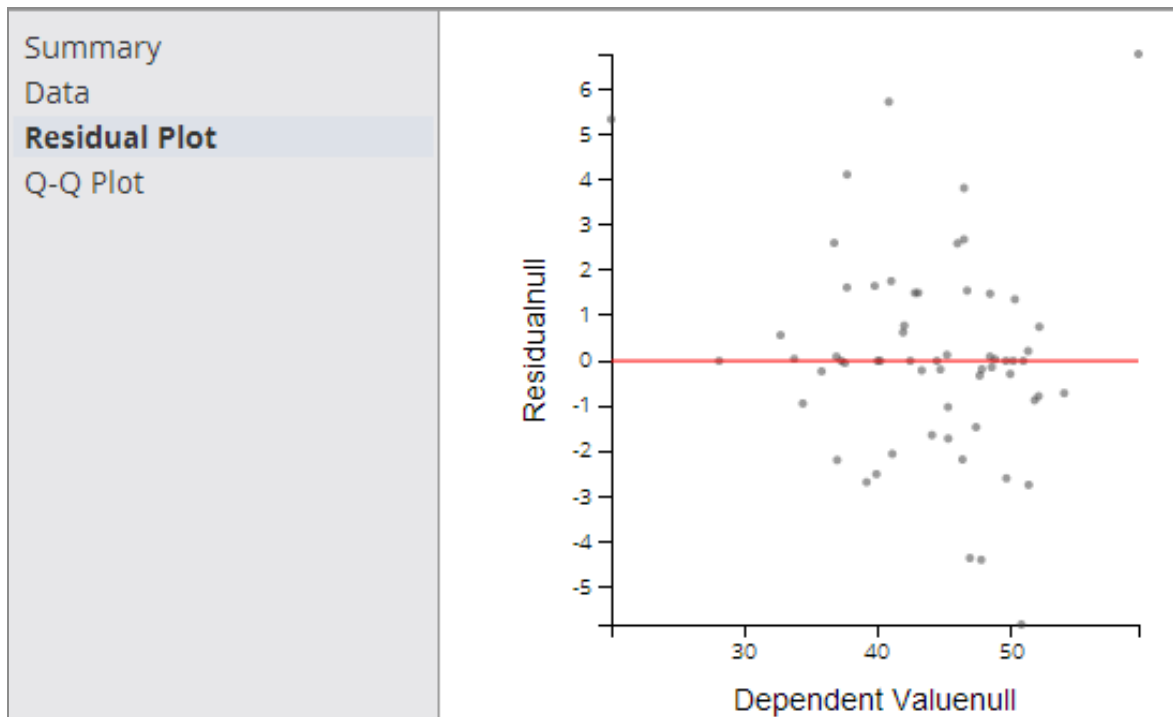
Column	Description
<b>Coefficient</b>	The model coefficient, $\beta$ , indicates the strength of the effect of the

Column	Description
	<p>associated independent variable on the dependent variable.</p> <p><b>Note:</b> when implementing Elastic Net Regularization, only the Coefficient results are displayed.</p> <p>In the case where L1 Regularization is applied (<math>\alpha &gt; 0</math>), if the resulting coefficient value is 0 it typically means that this variable is much less relevant to the model (assuming that normalization of the variables was performed beforehand).</p>
SE	<p>Standard Error, or SE, represents the standard deviation of the estimated coefficient values from the actual coefficient values for the set of variables in the regression.</p> <p>It is best practice to commonly expect + or - 2 standard errors, meaning the actual coefficient value should be within 2 SEs of the estimate. Therefore, a modeler looks for the SE values to be much smaller than the associated forecasted coefficient values.</p> <p><b>Note:</b> SE is not displayed if Elastic Net Regularization is implemented.</p>
T-statistic	<p>The T-statistic is computed by dividing the estimated value of the <math>\beta</math> Coefficient by its Standard Error, as follows: <math>T = \beta / SE</math>. It provides a scale for how big of an error the estimated coefficient has.</p> <ul style="list-style-type: none"> <li>• A small T-statistic alerts the modeler to the fact that the error is almost as big as the coefficient measurement and is therefore suspicious.</li> <li>• The larger the absolute value of T, the less likely that the unknown actual value of the coefficient could be zero.</li> </ul> <p><b>Note:</b> T-statistic is not displayed if Elastic Net Regularization is implemented.</p>

Column	Description
<b>P-value</b>	<p>The P-value represents the probability of still observing the dependent variable's value if the coefficient value for the independent variable is zero (that is, if p-value is high, the associated variable is not considered relevant as a correlated, independent variable in the model).</p> <ul style="list-style-type: none"> <li>• A low P-value is evidence that the estimated coefficient is not due to measurement error or coincidence, and therefore, is more likely a significant result. Thus, a low P-value gives the modeler confidence in the significance of the variable in the model.</li> <li>• Standard practice is to not trust coefficients with P-values greater than 0.05 (5%). Note:</li> </ul> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p><b>Note:</b> A P-value of less than 0.05 is often conceptualized as there being over 95% certainty that the coefficient is relevant.</p> <p>The P-value is not displayed if Elastic Net Regularization is implemented.</p> </div> <p>The smaller the P-value, the more meaningful the coefficient or the more certainty over the significance of the independent variable in the linear regression model. In summary, when assessing the Data tab results of the Linear Regression operator, a modeler mostly cares about the coefficient values, which indicate the strength of the effect of the independent variables on the dependent variable, and the associated P-values, which indicate how much not to trust the estimated correlation measurement.</p>

## Residual Plot

The Residual Plot displays a graph that shows the residuals (differences between the observed values of the dependent variable and the predicted values) of a linear regression model on the vertical axis and the independent variable on the horizontal axis, as shown in the following example.



A modeler should always look at the Residual Plot as it can quickly detect any systematic errors with the model that are not necessarily uncovered by the summary model statistics. It is expected that the Residuals of the dependent variable vary randomly above and below the horizontal axis for any value of the independent variable.

If the points in a residual plot are randomly dispersed around the horizontal axis, a linear regression model is appropriate for the data; otherwise, a non-linear model is more appropriate.

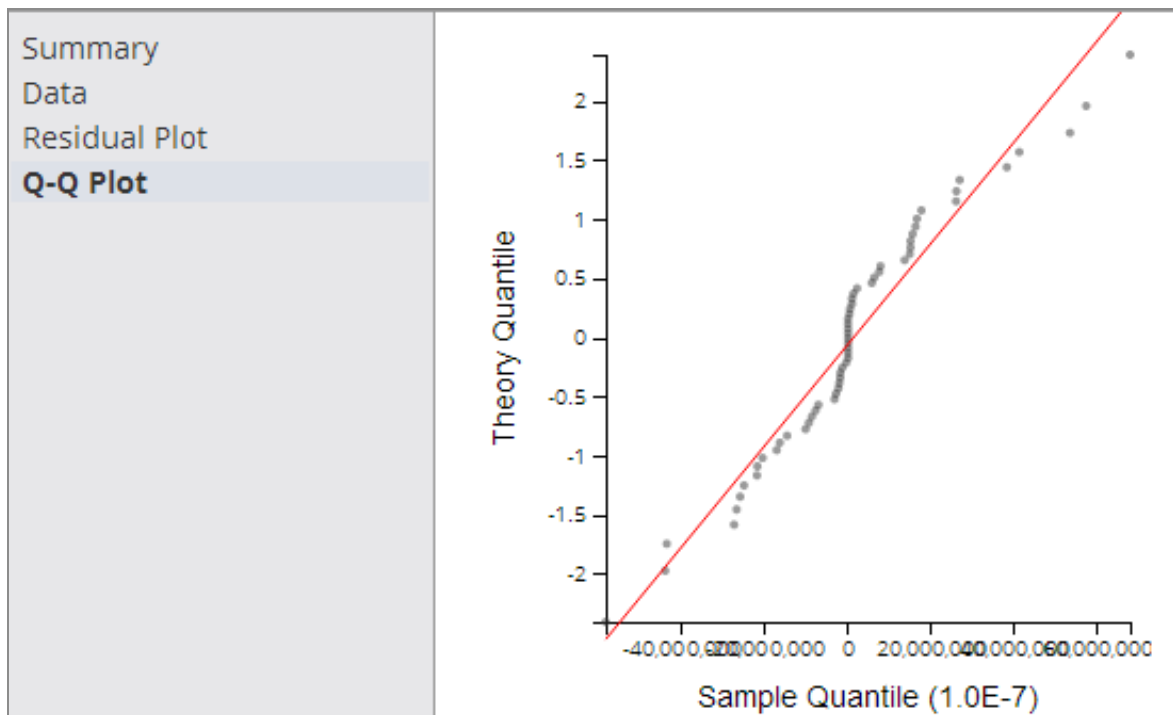
A "bad" Residual Plot has some sort of structural bend or anomaly that cannot be explained away.

For example, when analyzing medical data results, the linear regression model might show a good fit for male data but have a systematic error for female data. Glancing at a Residual Plot could quickly catch this structural weakness with the model.

In summary, the Residual Plot is an important diagnostic tool for analyzing linear regression results, allowing the modeler to keep a hand in the data while still analyzing overall model fit.

## Q-Q Plot

The Q-Q (Quantile-Quantile) Plot graphically compares the distribution of the residuals of a given variable to the normal distribution (represented by a straight line), as shown in the following example.

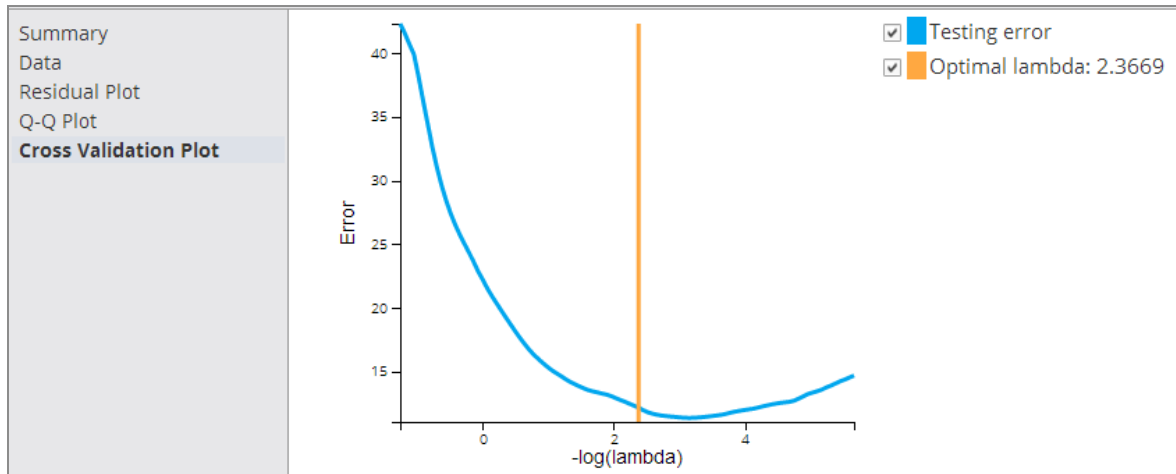


The closer the dots are to the line, the more normal of a distribution the data has. This provides a better sense of whether a linear regression model is a good fit for the data. Any sort of variance from the line for a certain quantile, or section, of data should be investigated and understood.

The Q-Q Plot is an interesting analysis tool, although not always easy to read or interpret.

## Cross Validation Plot (Elastic Net Regularization only)

Displays an automatically determined Optimal Lambda value to use for the Linear Regression Regularization penalty loss function.



This graph is only shown when Elastic Net Linear Regression is implemented.

Cross-validation is primarily a way of measuring the predictive performance of a statistical model.

- The best lambda is chosen automatically by the cross validation process. In the example above, the optimal lambda value is 2.3669.
- Lambda controls the degree of regularization with 0 meaning no regularization and infinity meaning ignoring all input variables because all correlation coefficients are turned to zero.

The higher the lambda,  $\lambda$ , the more constraints are imposed on the loss function, as shown in the following formula.

$$\max_{\{\beta_{0k}, \beta_k \in \mathbb{R}^p\}_1^K} \left[ \sum_{i=1}^N \log \Pr(g_i | x_i) - \lambda \sum_{k=1}^K \sum_{j=1}^p (\alpha |\beta_{kj}| + (1 - \alpha) \beta_{kj}^2) \right]$$

**i Note:** To learn more about the visualization available in this operator, go to [Explore Visual Results](#).

## References



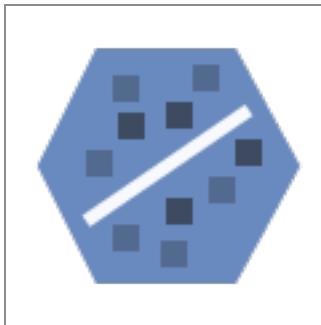
1. Definition taken from <http://www.dtreg.com/linreg.htm>
2. In actuality, the P-value is derived from the distribution curve of the T-statistic - it is the area under the curve outside of  $+ or - 2*SEs$  from the estimated Coefficient value.

### Data Output

A file with structure similar to the visual output structure is available.

## Linear Regression (DB)

Use the Linear Regression operator to fit a trend line to an observed data set, in which one of the data values - the dependent variable - is linearly dependent on the value of the other causal data values or variables - the independent variables.



### Information at a Glance

Parameter	Description
Category	Model
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a



**Note:** The Linear Regression (DB) operator is for database data only. For Hadoop data, use the [Linear Regression \(HD\)](#) operator.

For more information about using linear regression, see [Fitting a Trend Line for Linearly Dependent Data Values](#).

## Algorithm

The TIBCO Data Science - Team Studio Linear Regression operator applies a Multivariate Linear Regression (MLR) algorithm to the input data set. For MLR, a Regularization Penalty Parameter that can be applied in order to prevent the chances of over-fitting the model.

The Linear Regression operator implements Ordinary Regression and allows for the Stepwise Feature to avoid over-fitting a model with too many variables. The Ordinary Regression algorithm uses the Ordinary Least Squares (OLS) method of regression analysis, meaning that the model is fit such that the sum-of-squares of differences of observed and predicted values is minimized.

## Input

A data set that contains the dependent and independent variables for modeling.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	<p>The dependent column specified for the regression. This is the quantity to model or predict. The list of the available data columns for the Regression operator is displayed. Select the data column to consider the dependent variable for the regression.</p> <p>The <b>Dependent Column</b> should be a numerical data type.</p>
<b>Columns</b>	<p>Click <b>Select Columns</b> to select the available columns from the input data set for analysis.</p> <p>For a linear regression, select the independent variable data columns for the regression analysis or model training.</p>

Parameter	Description
	You must select at least one column or one interaction variable.
<b>Interaction Parameters</b>	Enables selecting available independent variables, where those data parameters might have a combined effect on the dependent variable. See <a href="#">Interaction Parameters dialog</a> for detailed information.
<b>Stepwise Feature Selection</b>	<ul style="list-style-type: none"> <li>• <b>true</b> implements stepwise regression methodology. Selecting <b>true</b> means that one of the possible <b>Stepwise Type</b> regression methods is used and the <b>Criterion Type</b> and <b>Check Value</b> must be specified.</li> </ul> <p>The <b>Stepwise Feature</b> allows the system to find a subset of variables that works just as well as the larger, original variable set. A smaller model is generally considered by data scientists to be safer from the danger of over fitting a model with too many variables.</p> <ul style="list-style-type: none"> <li>• <b>false</b> (the default) specifies that all the independent variables specified in <b>Column Names</b> and <b>Interaction Columns</b> are considered at once when running the regression analysis and included in the model.</li> </ul>
<b>Stepwise Type</b>	<p>Specifies the different ways to determine which of the independent variables are the most predictive to include in the model.</p> <p>This option is enabled only if <b>Stepwise Feature Selection</b> is selected.</p> <p>For all <b>Stepwise Type</b> methods, the minimum significance value is defined by the operator's <b>Check Value</b> parameter specified, and the approach for determining the significance is defined by <b>Criterion Type</b>.</p> <ul style="list-style-type: none"> <li>• <b>FORWARD</b> - (The default.) For a Forward Regression analysis process, the feature selection begins with no variables in the model and adds one variable at a time. Each potential independent variable's contribution to the model is calculated individually. The most significant variable - as defined by the approach selected for <b>Criterion Type</b> - is first added to the model. This process is repeated until none of the remaining unused variables meets a minimum significance level. After a variable is included it remains in the model.</li> </ul>

Parameter	Description
	<p><b>Note:</b> Use this method if there is a large set of variables and you suspect that only a few of them are needed.</p> <ul style="list-style-type: none"> <li>• <b>BACKWARD</b> - For a Backward Regression analysis process, the feature selection begins with all variables included in the model. The significance of the variables is calculated and the variable with the least significance - as defined by the approach selected for <b>Criterion Type</b> - is removed from the model. The process is repeated until the least significant variable meets a minimum significance level. Use this method if you are starting with a small set of variables and only a few need to be eliminated.</li> <li>• <b>STEPWISE</b> For a stepwise regression analysis process, the same FORWARD method steps are taken, except that after a variable is added to the model, the included variables are re-evaluated for significance. If an included variable no longer meets the significance criteria, it is removed from the model. Feature selection of variables to include terminates when none of the remaining variables meet the selection criteria or the last variable to be included has also just been removed. This is the most powerful and typically used <b>Stepwise Type</b>.</li> </ul>
<b>Criterion Type</b>	<p>Specifies the approach for evaluating a variable's significance in the regression model.</p> <p>Enabled only if <b>Stepwise Feature Selection</b> is selected.</p> <ul style="list-style-type: none"> <li>• <b>AIC</b> - The popular Akaike Information Criterion (AIC) is a specific measure of the relative goodness of fit of a statistical model. Selecting this criterion type applies a function of the number of features or variables included and the maximized likelihood function for the model.</li> <li>• <b>SBC</b> - The Schwarz Bayesian Information Criterion (SBC) is similar to the AIC significance function but includes a larger penalty term for the number of selected features - that is, included variables.</li> </ul> <p>Choose SBC to prevent over-fitting of a model by not taking on too many variables.</p>

Parameter	Description
<b>Check Value</b>	<p>Specifies the minimal significance level value to use as feature selection criterion in Forward, Backward, or Stepwise Regression Analysis.</p> <p>Enabled only if <b>Stepwise Feature Selection</b> is selected.</p> <p>Default value: <b>0.05</b>. If you are running without a stepwise approach, consider setting <b>Check Value</b> to 10% of the resulting AIC value.</p>
<b>Group By</b>	<p>Specifies a column for categorizing or subdividing the model into multiple models based on different groupings of the data. A typical example is using gender to create two different models based on the data for males versus females. A modeler might do this to determine if there is a significant difference in the correlation between the dependent variable and the independent variable based on whether the data is for a male or a female.</p> <p>The <b>Group By</b> column cannot be selected as a <b>Dependent Column</b> or as an independent variable (in <b>Columns</b> in the model).</p>
<b>Draw Residual Plot</b>	<p>Provides the option to output Q-Q Plot and Residual Plot graphs for the linear regression results.</p> <p>See Output for details on the resulting output graphs when <b>Draw Residual Plot</b> is set to <b>true</b>.</p> <p>Default value: <b>false</b>.</p>

## Output

### Visual Output

#### Ordinary Linear Regression Output

Because data scientists expect model prediction errors to be unstructured and normally distributed, the Residual Plot and Q-Q Plot together are important linear regression diagnostic tools, in conjunction with R<sup>2</sup>, Coefficient and P-value summary statistics.

The remaining visual output consists of **Summary**, **Data**, **Residual Plot**, and **Q-Q Plot**.

## Summary

The **Summary** output displays the details of the derived linear regression model's Equation and Correlation Coefficient values along with the R2 and Standard Error statistical values.

Summary	turnout =
Data	- 0.0022 * msa
Residual Plot	+ 1.0E-4 * popdensity
Q-Q Plot	+ 0.0 * pop
	+ 0.0626 * popchange
	+ 0.5235 * age6574
	+ 1.6894 * age75
	+ 0.0 * crime
	+ 0.1694 * college
	+ 5.0E-4 * income
	+ 0.0518 * farm
	+ 0.1754 * democrat
	+ 0.0953 * republican
	+ 0.4265 * Perot
	+ 0.2858 * white
	+ 0.2649 * black

The derived linear regression model is shown as a mathematical equation linking the Dependent Variable (Y) to the independent variables (X1, X2, etc.). It includes the scaling or Coefficient values ( $\beta_1$ ,  $\beta_2$ , etc.) associated with each independent variable in the model.

**Note:** The resulting linear equation is expressed in the form of  $Y = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \dots$

The following overall model statistical fit numbers are displayed.

- R2: it is called the multiple correlation coefficient of the model, or the Coefficient of Multiple Determination. It represents the fraction of the total Dependent Variable (Y) variance explained by the regression analysis, with 0

meaning 0% explanation of Y variance and 1 meaning 100% accurate fit or prediction capability.

**i Note:** in general, an R2 value greater than .8 is considered a good model. However, this value is relative and in some situations just getting an improved R2 from .5 to .6, for example, would be beneficial.

- S: represents the standard error per model (often also denoted by SE). It is a measure of the average amount that the regression model equation over-predicts or under-predicts.

The rule of thumb data scientists use is that 60% of the model predictions are within +/- 1 SE and 90% are within +/- 2 SEs.

For example, if a linear regression model predicts the quality of the wine on a scale between 1 and 10 and the SE is .6 per model prediction, a prediction of Quality=8 means the true value is 90% likely to be within 2\*.6 of the predicted 8 value (that is, the real Quality value is likely between 6.8 and 9.2).

In summary, the higher the R2 and the lower the SE, the more accurate the linear regression model predictions are likely to be.

## Data

The **Data** results are a table that contains the model coefficients and statistical fit numbers for each independent variable in the model.

Summary	Attribute	Coefficient	SE	T-statistics	P-value
<b>Data</b>	farm	0.3934	0.2087	1.8849	0.0621
Residual Plot	democrat	-1.0151	0.7436	-1.3651	0.175
Q-Q Plot	republican	-1.1496	0.7465	-1.54	0.1264
	Perot	-0.9154	0.7611	-1.2028	0.2317
	white	0.3315	0.0898	3.6898	0.0004

Column	Description
<b>Coefficient</b>	The model coefficient, $\beta$ , indicates the strength of the effect of the

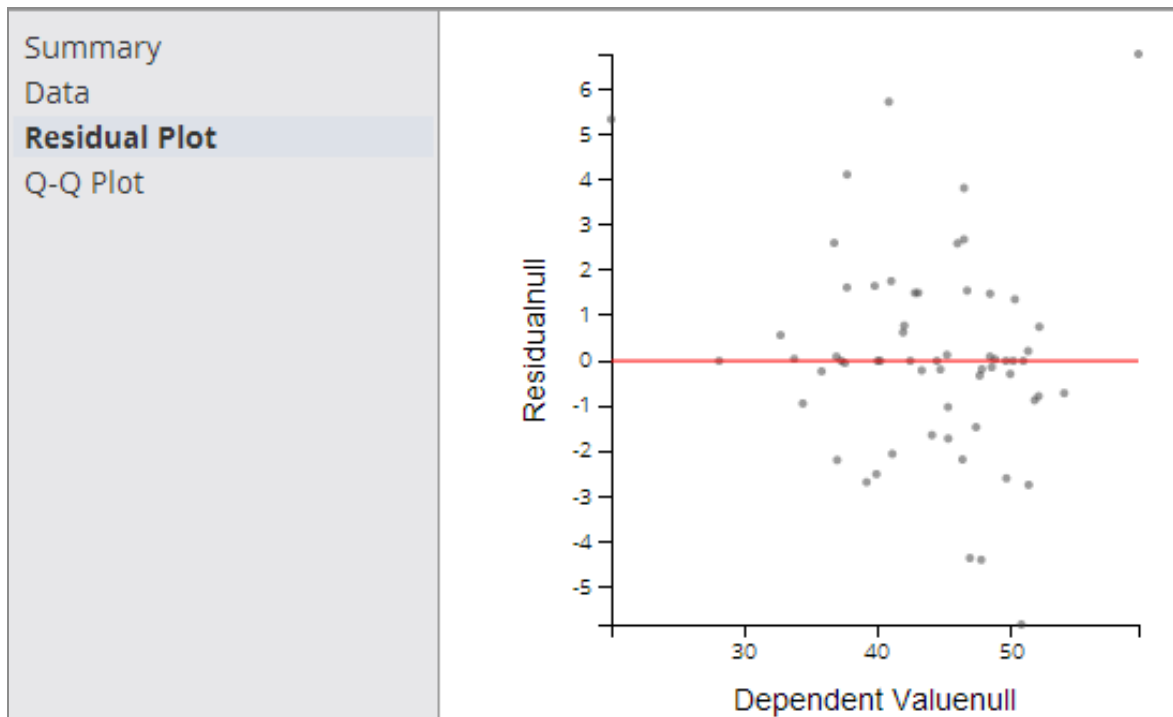
Column	Description
	<p>associated independent variable on the dependent variable.</p> <p>In the case where L1 Regularization is applied (<math>\alpha &gt; 0</math>), if the resulting coefficient value is 0 it typically means that this variable is much less relevant to the model (assuming that normalization of the variables was performed beforehand).</p>
<b>SE</b>	<p>Standard Error, or SE, represents the standard deviation of the estimated coefficient values from the actual coefficient values for the set of variables in the regression.</p> <p>It is best practice to commonly expect + or - 2 standard errors, meaning the actual coefficient value should be within 2 SEs of the estimate. Therefore, a modeler looks for the SE values to be much smaller than the associated forecasted coefficient values.</p>
<b>T-statistic</b>	<p>The T-statistic is computed by dividing the estimated value of the <math>\beta</math> Coefficient by its Standard Error, as follows: <math>T = \beta / SE</math>. It provides a scale for how big of an error the estimated coefficient has.</p> <ul style="list-style-type: none"> <li>• A small T-statistic alerts the modeler to the fact that the error is almost as big as the coefficient measurement and is therefore suspicious.</li> <li>• The larger the absolute value of T, the less likely that the unknown actual value of the coefficient could be zero.</li> </ul>
<b>P-value</b>	<p>The P-value represents the probability of still observing the dependent variable's value if the coefficient value for the independent variable is zero (that is, if p-value is high, the associated variable is not considered relevant as a correlated, independent variable in the model).</p> <ul style="list-style-type: none"> <li>• A low P-value is evidence that the estimated coefficient is not due to measurement error or coincidence, and therefore, is more likely a significant result. Thus, a low P-value gives the modeler confidence in the significance of the variable in the model.</li> </ul>



Column	Description
	<ul style="list-style-type: none"><li>Standard practice is to not trust coefficients with P-values greater than 0.05 (5%). Note:</li></ul> <div><b>Note:</b> A P-value of less than 0.05 is often conceptualized as there being over 95% certainty that the coefficient is relevant.</div> <p>The smaller the P-value, the more meaningful the coefficient or the more certainty over the significance of the independent variable in the linear regression model. In summary, when assessing the Data tab results of the Linear Regression operator, a modeler mostly cares about the coefficient values, which indicate the strength of the effect of the independent variables on the dependent variable, and the associated P-values, which indicate how much not to trust the estimated correlation measurement.</p>

## Residual Plot

The Residual Plot displays a graph that shows the residuals (differences between the observed values of the dependent variable and the predicted values) of a linear regression model on the vertical axis and the independent variable on the horizontal axis, as shown in the following example.



A modeler should always look at the Residual Plot as it can quickly detect any systematic errors with the model that are not necessarily uncovered by the summary model statistics. It is expected that the Residuals of the dependent variable vary randomly above and below the horizontal axis for any value of the independent variable.

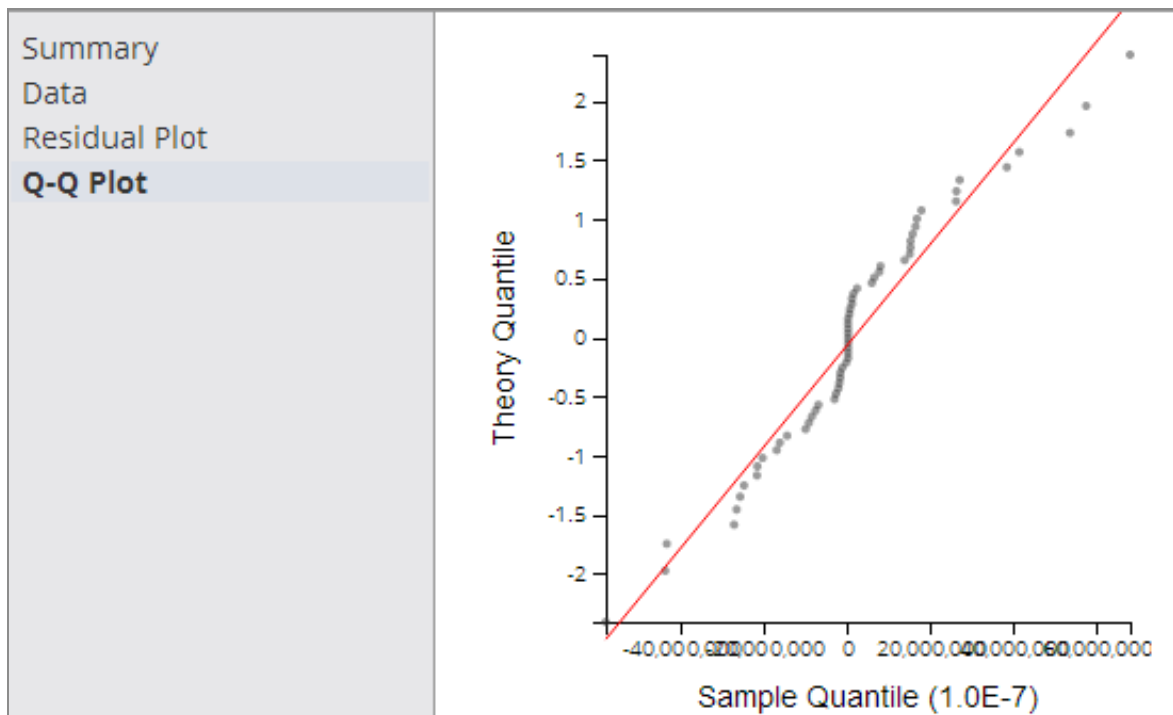
If the points in a residual plot are randomly dispersed around the horizontal axis, a linear regression model is appropriate for the data; otherwise, a non-linear model is more appropriate.

A "bad" Residual Plot has some sort of structural bend or anomaly that cannot be explained away. For example, when analyzing medical data results, the linear regression model might show a good fit for male data but have a systematic error for female data. Glancing at a Residual Plot could quickly catch this structural weakness with the model.

In summary, the Residual Plot is an important diagnostic tool for analyzing linear regression results, allowing the modeler to keep a hand in the data while still analyzing overall model fit.

## Q-Q Plot

The Q-Q (Quantile-Quantile) Plot graphically compares the distribution of the residuals of a given variable to the normal distribution (represented by a straight line), as shown in the following example.



The closer the dots are to the line, the more normal of a distribution the data has. This provides a better sense of whether a linear regression model is a good fit for the data. Any sort of variance from the line for a certain quantile, or section, of data should be investigated and understood.

The Q-Q Plot is an interesting analysis tool, although not always easy to read or interpret.

## Data Output

A file with structure similar to the visual output structure is available.

## Linear Regression - MADlib

TIBCO Data Science - Team Studio supports the MADlib open source implementation of the Linear Regression algorithm.



## Information at a Glance

Parameter	Description
Category	Model
Data source type	DB
Send output to other operators	No
Data processing tool	MADlib

## Algorithm

The MADlib Linear Regression operator applies an Ordinary Least-Squares (OLS) linear regression algorithm to the input dataset. It is processed using the least squares method of regression analysis, meaning that the model is fit such that the sum-of-squares of differences of observed and predicted values is minimized.

More information including general principles can be found in the [official MADlib documentation](#).

## Input

A data set that contains the dependent and independent variables for modeling.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>MADlib Schema Name</b>	Schema where MADlib is installed in the database. MADlib must be installed in the same database as the input dataset. If a "madlib" schema exists in the database, this parameter defaults to madlib.
<b>Model Output Schema Name</b>	The name of the schema where the output is stored.
<b>Model Output Table Name</b>	<p>The name of the table that is created to store the Regression model. Specifically, the model output table stores:</p> <pre>[ group_col_1   group_col_2   ...  ] coef   r2   std_err   t_stats   p_values   condition_no [  bp_stats   bp_p_value]</pre> <p>See the <a href="#">official MADlib linear regression documentation</a> for more information.</p>
<b>Drop If Exists</b>	<ul style="list-style-type: none"> <li>• If <b>Yes</b> (the default), drop the existing table of the same name and create a new one.</li> <li>• If <b>No</b>, stop the flow and alert the user that an error has occurred.</li> </ul>
<b>Dependent Variable</b>	<p>Required. The quantity to model or predict.</p> <ul style="list-style-type: none"> <li>• The list of the available columns is displayed. Select the data column to be considered the Dependent Variable for the regression.</li> <li>• The Dependent Variable should be a numerical data type.</li> </ul>
<b>Independent Variables</b>	<p>Click <b>Select Columns</b> to select the available columns from the input data set for analysis.</p> <p>Select the independent variable data columns for the regression</p>

Parameter	Description
	analysis or model training. You must select at least one column.
<b>Grouping Columns</b>	You can set at least one column to group the input data and build separate regression models for each group.  Click <b>Select Columns</b> to open the dialog for selecting the available columns from the input dataset for grouping.
<b>Heteroskedacity Stat</b>	Set to <b>true</b> (the default) to output two additional columns to the model table. <ul style="list-style-type: none"> <li>• Breusch-Pagan test statistics (bp_stats)</li> <li>• the corresponding p-value (bp_p_value)</li> </ul>
<b>Draw Residual Plot</b>	Set to <b>true</b> (the default) to output Q-Q Plot and Residual Plot graphs for the linear regression results. <ul style="list-style-type: none"> <li>• The Q-Q Plot graphically compares the distribution of the residuals of a given variable to the normal distribution (represented by a straight line).</li> <li>• The Residual Plot displays a graph that shows the residuals of a linear regression model on the vertical axis and the independent variable on the horizontal axis.</li> </ul>

## Output

When assessing the **Data** tab results of the Linear Regression Operator, a modeler focuses mostly the Coefficient values, which indicate the strength of the effect of the independent variables on the dependent variable, and the associated P-values, which indicate how much not to trust the estimated correlation measurement.

### Visual Output

The MADlib Linear Regression Operator results output is displayed across the **Summary** and **Data** sections.

## Summary

The derived linear regression model is a mathematical equation linking the Dependent Variable ( $Y$ ) to the Independent Variables ( $X1$ ,  $X2$ , etc.). It includes the scaling or Coefficient values ( $\beta1$ ,  $\beta2$ , etc.) associated with each independent variable in the model. Note: The resulting linear equation is expressed in the form of  $Y = \beta0 + \beta1 * X1 + \beta2 * X2 + \dots$

The following overall model statistical fit numbers:

- **R2:** R2 is called the multiple correlation coefficient of the model, or the Coefficient of Multiple Determination. It represents the fraction of the total Dependent Variable ( $Y$ ) variance explained by the regression analysis, with 0 meaning 0% explanation of  $Y$  variance and 1 meaning 100% accurate fit or prediction capability.

**i Note:** In general, an R2 value greater than .8 is considered a good model. However, this value is relative and in some situations just getting an improved R2 from .5 to .6, for example, would be beneficial.

- **S:** represents the standard error per model (often also denoted by SE). It is a measure of the average amount that the regression model equation over- or under-predicts.
  - The rule of thumb data scientists use is that 60% of the model predictions are within  $\pm 1$  SE and 90% are within  $\pm 2$  SEs.

For example, if a linear regression model predicts the quality of the wine on a scale between 1 and 10 and the SE is .6 per model prediction, then a prediction of Quality=8 means the true value is 90% likely to be within  $2 * .6$  of the predicted 8 value (that is, the real Quality value is likely between 6.8 and 9.2).

**i Note:** The higher the R2 and the lower the SE, the more accurate the linear regression model predictions are likely to be.

Results - Linear Regression (MADlib)	
Summary	temperature =
Data	58.921 * 1
	+ 0.182 * humidity
	R2: 0.075
	t stats: [3.9301155211303573, 0.9839956271590881]
	p-values: [0.0019982583549098543, 0.3445419896743074]
	condition number: 689.3414326412942

## Data

Displays the model coefficients and statistical fit numbers for each Independent variable in the model.

Results - Linear Regression (MADlib)

Summary

Data

Attribute	Coefficient	SE	T-statistics	P-value
intercept	58.921	14.992	3.93	0.002
humidity	0.182	0.185	0.984	0.345

Column	Description
<b>Coefficient</b>	<p>The model coefficient, <math>\beta</math>, indicates the strength of the effect of the associated independent variable on the dependent variable.</p> <p>Standard Error, or SE, represents the standard deviation of the estimated Coefficient values from the actual Coefficient values for the set of Variables in the regression.</p> <ul style="list-style-type: none"> <li>It is best practice to commonly expect + or - 2 Standard Errors, meaning the actual Coefficient value should be within 2 SEs of the estimate.</li> <li>Therefore, a modeler looks for the SE values to be much smaller than the associated forecasted Coefficient values.</li> </ul>
<b>T-statistic</b>	<p>The T-statistic is computed by dividing the estimated value of the <math>\beta</math> Coefficient by its Standard Error, as follows: <math>T = \beta / SE</math>. It provides a scale for how big of an error the estimated coefficient has.</p> <ul style="list-style-type: none"> <li>A small T-statistic alerts the modeler to the fact that the error is almost as big as the Coefficient measurement and is therefore</li> </ul>



Column	Description
	<p>suspicious.</p> <ul style="list-style-type: none"> <li>The larger the absolute value of T, the less likely that the unknown actual value of the Coefficient could be zero.<sup>1</sup></li> </ul>
<b>P-value</b>	<p>P-value represents the probability of still observing the dependent variable's value if the Coefficient value for the independent variable is zero (that is, if p-value is high then the associated variable would not be considered relevant as a correlated, independent variable in the model).</p> <ul style="list-style-type: none"> <li>A low P-value is evidence that the estimated Coefficient is not due to measurement error or coincidence, and therefore, is more likely a significant result. Thus, a low P-value gives the modeler confidence in the significance of the variable in the model.</li> <li>Standard practice is to not trust Coefficients with P-values greater than 0.05 (5%). Note: a P-value of less than 0.05 is often conceptualized as there being over 95% certainty that the Coefficient is relevant.<sup>2</sup></li> </ul> <div> <p><b>Note:</b> The smaller the P-value, the more meaningful the coefficient or the more certainty over the significance of the independent variable in the Linear Regression model.</p> </div>

### Data Output

None. This is a terminal operator.

## Logistic Regression (DB)

The Logistic Regression operator fits an s-curve logistic or logit function to a data set to calculate the probability of the occurrence of a specific categorical event based on the values of a set of independent variables.



## Information at a Glance

Parameter	Description
Category	Model
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

**i Note:** The Logistic Regression (DB) operator is for database data only. For Hadoop data, use the [Logistic Regression \(HD\)](#) operator.

For more detailed information on logistic regression, use cases, and this operator, see [Probability Calculation Using Logistic Regression](#).

## Algorithm

The database implementation for logistic regression implements a binomial logistic regression algorithm (and a StepWise Feature Selection capability to avoid over-fitting a model with too many variables). Binomial or binary logistic regression refers to the instance in which the criterion can take on only two possible outcomes (for example, "dead" vs. "alive", "success" vs. "failure", or "yes" vs. "no").

For binomial logistic regression, the Logistic Regression operator computes a probability model for the likelihood of the **Value to Predict** based on the values of causal independent variables.

The binomial logistic regression Algorithm uses the Iteratively Re-weighted Least Squares (IRLS) method of fitting a binomial logit function to a data set.

- The TIBCO Data Science - Team Studio Logistics Regression Operator applies a binomial regression by assuming the dependent variable is either the **Value to Predict** or Not (**Value to Predict**).
- For binomial logistic regression, the dependent variable must have only two distinct possible discrete values, such as "yes/no" or "0/1".
- For binomial logistic regression, the operator requires numeric independent variable values. However, if categorical independent variables (such as eye color) are specified in the source data set, the TIBCO Data Science - Team Studio algorithm automatically converts them into "levels" behind the scenes before running the logistic regression training.

The values of categorical variables are often referred to as levels. In TIBCO Data Science - Team Studio, each level is treated as a Boolean value. For example, the "eye color" variable might be represented by three Boolean levels, *IsBlue?*, *IsGreen?*, and *IsBrown?*.

## Input

A data set that contains the dependent and independent variables for modeling.

### Bad or Missing Values

Predictions are made only for rows that contain data. Rows with missing data are skipped.

## Restrictions

Multinomial regressions are not currently supported in the database implementation.

## Configuration

As of TIBCO Data Science - Team Studio 6.3, the Logistic Regression operator searches the parameter space of  $\lambda$  and  $\alpha$  and automatically selects the highest performing model. To use this feature, provide either a comma-separated list (for example, .1,.2,.3) for  $\lambda$  or  $\alpha$ , or start:end:step (for example, 0:1:.1). The operator computes all possible  $\lambda$  and  $\alpha$  combinations, and the output from the operator is the model with the highest classification performance. The results of every parameter combination are visible in the results console under the **Parameter optimization** results tab.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	<p>The quantity to model or predict. A dependent column must be specified for the logistic regression. Select the data column to be considered the dependent variable for the regression.</p> <p>The dependent column is often a categorical type, such as eye color = blue, green, or brown.</p> <p><b>Note:</b> For binomial logistic regression, the <b>Dependent Column</b> must be able to be categorized for a "Yes", "No" prediction (that is, it cannot have more than two distinct values) while for multinomial logistic regression, the <b>Dependent Column</b> can have multiple categorical values to predict.</p>
<b>Value to Predict</b>	<p>Required for binomial logistic regression only. You must specify a <b>Value to Predict</b> that represents the value stored in the dependent variable column that should be the event to analyze.</p> <p>For example, the <b>Value to Predict</b> could be Active vs. Inactive. This specifies the value that the dependent variable must have to be considered a "successful" event in the logistic regression.</p> <p>For binomial logistic regression, the value of the <b>Dependent Column</b> that indicates a positive event to predict is required input. For example, the value to predict could be "Yes" for defaulting on a loan.</p> <p><b>Note:</b> The value of this column must match the data as it is stored in the database that matches how it appears in the data explorer. If you define a Boolean dependent column with 1s and 0s, you must use 1 or 0 as the <b>Value to Predict</b>. If the column uses Trues and Falses, you must use "True" or "False" as the <b>Value to Predict</b>.</p>
<b>Maximum Number of Iterations</b>	<p>The total number of regression iterations that are processed before the algorithm stops if the coefficients do not converge, or show relevance. This parameter must be an integer value <math>\geq 1</math>.</p> <p>Default value: <b>10</b>.</p>

Parameter	Description
<b>Tolerance</b>	<p>Logistic regression requires a tolerance value to be specified. This is used to determine the maximum allowed error value for the IRLS calculation method. When the error is smaller than this value, the logistic regression model training stops. This parameter must be a decimal value <math>\geq 0</math>.</p> <p>Default value: <b>0.0001</b>.</p>
<b>Columns</b>	<p>Specifies the independent variable data columns to include for the regression analysis or model training. At least one column or one interaction variable must be specified.</p> <p>Click the <b>Columns</b> button to open the dialog for selecting the available columns from the input data set for analysis. For more information, see <a href="#">Select Columns dialog</a>.</p>
<b>Interaction Parameters</b>	<p>Enables the selection of available independent variables as those data parameters thought to have combined effect on the dependent variable.</p> <p>Creating interaction parameters is useful when the modeler believes the combined interaction of two independent variables is not additive.</p> <p>To define an interaction parameter, click the <b>Interaction Parameters</b> button and select the suspected interacting data columns.</p> <p>If you have feature A and feature B, selecting * uses both A, B, and the interaction A*B as independent features. Selecting : means that only A*B is used in the model.</p>
<b>Stepwise Feature Selection</b>	<p>Specifies the implementation of Stepwise Regression methodology. Setting this option to <b>true</b> specifies that one of the possible Stepwise Type regression methods defined below is used, and that the <b>CriterionType</b> and <b>Check Value</b> must be specified.</p> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p><b>Note:</b> Stepwise allows the system to find a subset of variables that works just as well as the larger, original variable set. A smaller model is generally considered by data scientists to be safer from the danger of overfitting a model with too many variables.</p> </div> <p>Default value: <b>false</b>, meaning that all the independent variables are considered at once when running the Regression analysis and included in</p>

Parameter	Description
	the model.
<b>Stepwise Type</b>	<p>Required if <b>Stepwise Feature Selection</b> is set to <b>Yes</b>. Specifies the different ways to determine which of the independent variables are the most predictive to include in the model.</p> <ul style="list-style-type: none"> <li>• <b>FORWARD</b> (the default): For a Forward Regression analysis process, the feature selection begins with no variables in the model and adds one variable at a time. Each potential independent variable's contribution to the model is calculated individually. The most significant variable - as defined by the approach selected for <b>Criterion Type</b> - is first added to the model. This process is repeated until none of the remaining unused variables meets a minimum significance level. Once included, a variable remains in the model.</li> <li>• <b>BACKWARD</b>: Use this method if there is a large set of variables and it is suspected that only a few variables are needed. For a Backward Regression analysis process, the feature selection begins with all variables included in the model. The significance of the variables is calculated and the variable with the least significance - as defined by the approach selected for <b>Criterion Type</b> below - is removed from the model. The process is repeated until the least significant variable meets a minimum significance level. Use this method if you are starting with a small set of variables and only a few need to be eliminated.</li> <li>• <b>STEPWISE</b>: For a Stepwise Regression analysis process, the same <b>FORWARD</b> method steps are taken except that after a variable is added to the model, the included variables are re-evaluated for significance. If an included variable no longer meets the significance criteria, it is removed from the model. Feature selection of variables to include terminates when none of the remaining variables meet the selection criteria or the last variable to be included has also just been removed. This is the most powerful and typically used stepwise type.</li> </ul> <p>For these <b>Stepwise Type</b> methods, the minimum significance value is defined by the operator's <b>Check Value</b> parameter specified, and the approach for determining the significance is defined by <b>Criterion Type</b>.</p>

Parameter	Description
<b>Criterion Type</b>	<p>Required if <b>Stepwise Feature Selection</b> is set to <b>Yes</b>. Specifies the approach to use for evaluating a variable's significance in the Regression Model.</p> <ul style="list-style-type: none"> <li>• <b>AIC:</b> A popular criterion, the Akaike Information Criterion, is a specific measure of the relative goodness of fit of a statistical model. Selecting this AIC criterion type applies a function of the number of features or variables included and the maximized likelihood function for the model.</li> <li>• <b>SBC:</b> The Schwarz Bayesian Information Criterion is similar to the AIC significance function, except it includes a larger penalty term for the number of selected features (that is, included variables).</li> </ul> <p><b>Note:</b> The <b>SBC</b> Criterion is recommended because it prevents over-fitting of a model by not trying to analyze too many variables.</p>
<b>Check Value</b>	<p>Required if <b>Stepwise Feature Selection</b> is set to <b>Yes</b>. Specifies the minimal significance level value to use as feature selection criterion in <b>FORWARD</b>, <b>BACKWARD</b>, or <b>STEPWISE</b> regression analysis.</p> <p>Default value: <b>0.05</b>. Alternatively, set <b>Check Value</b> to 10% of the resulting AIC value without a stepwise approach.</p>
<b>Group By</b>	<p>Specifies a column for categorizing or subdividing the model into multiple models based on different groupings of the data. A typical example is using Gender to create two different models based on the data for males versus females. A modeler might do this to determine if there is a significant difference in the correlation between the dependent variable and the independent variable based on whether the data is for a male or a female.</p> <p><b>Note:</b> The <b>Group By</b> column cannot already be selected as a dependent or independent variable in the model.</p>

## Output

**i Note:** In addition to the required logistic regression Prediction operator, it is helpful for the modeler to add Model Validation operators to get further model accuracy statistics (from the Goodness of Fit operator) and/or visual outputs (from the ROC and LIFT operators). See the [Model Validation Operators](#) operators section for more details.


## Visual Results

The **Summary** output displays the **Number of iterations**, **Deviance**, **Null deviance**, **Chi-squared value**, and **Fraction of variance explained** statistical values.

RESULTS - Logistic Regression- Database Config	
<b>Summary</b>	Number of iterations: 6
Data	Deviance: 14366.2950
	Null deviance: 18528.9167
	Chi-squared value: 4162.6217
	Fraction of variance explained: 0.2247

- **Number of iterations:** Indicates the number of times the logistic regression re-weighting process was run. When Iteration = **Maximum Number of Iterations**, it flags that the regression might not have yet converged or there was a fit failure (that is, no correlation pattern was uncovered).
- **Deviance:** Used as a statistic for overall fit of a logistic regression model. However, this number is meaningless by itself - it should be compared against the **Null deviance** value below or with its own value from previous runs of the regression.
  - **Deviance** is the comparison of the observed values,  $Y$ , to the expected values  $Y$  predicted.
  - The bigger the difference or **Deviance** of the observed values from the expected values, the poorer the fit of the model.



- As more independent variables are added to the model, the deviance should get smaller, indicating improvement of fit.
- **Null deviance:** Indicates the deviance of a "dumb" model - a random guess of yes/no without any predictor variables.
  - It is used as a comparison of performance against the model **Deviance** above.
  - The smaller the model **Deviance** (using predictors) can be made relative to the **Null deviance** (no predictors), the better the logistic regression model.
- **Chi-squared value:** The difference between the **Null deviance** and **Deviance**. **Chi-square** technically represents the "negative two log likelihood" or -2LL deviance statistic for measuring the logistic regression's effectiveness. **Chi square = Null deviance minus Deviance**.
  - The hope is for the **Deviance** to be less than the **Null deviance**. Another flag for the logistic regression model not converging or there being a fit failure is having the **Deviance > Null deviance**, or a negative chi square. This might indicate that there is a subset of the data that is over fit on - the modeler could try removing variables and rerunning the regression.
  - 

**Note:** Since the chi square is a measure of the difference between predicted and actual, it is similar to looking at the residuals for a Linear Regression.
- **Fraction of variance explained:** The **Chi-squared value** divided by the **Null deviance**. This ratio provides a very useful diagnostic statistic representing the percentage of the system's variation the model explains (compared to a dumb model). When analyzing logistic regression results, looking at the **Chi-squared/Null deviance** value provides a similar statistic to the R2 value for Linear Regressions. As a rule of thumb, any **Chi-squared/Null deviance** value over .8 (80%) is considered a successful logistic regression model fit.

## Data Results


The **Data** output displays the statistical fit numbers for each independent variable in the model.

RESULTS - Logistic Regression- Database Config							
Summary	Attribute	beta	Odds Ratio	SE	Z-value	P-value	Wald
Data	Bias (offset)	-4.3961	N/A	0.1219	-36.0738	1.372e-67	1,301.3193
	times90dayslate	2.1256	8.3777	0.0573	37.1248	9.733e-69	1,378.2501
	revolving_util	-0.1255	0.8821	0.1475	-0.8511	0.3947	0.7244
	debt_ratio	2.3933	10.9495	0.0667	35.8949	2.169e-67	1,288.4431
	credit_lines	-0.0298	0.9706	0.0195	-1.5276	0.1266	2.3335
	monthly_income	0	1	0	-3.4634	0.0005	11.9952
	times30dayslate_2years	-0.0096	0.9904	0.0477	-0.202	0.8399	0.0408

RESULTS - Logistic Regression - Hadoop Config							
Summary	Attribute	Dependent Value	Coefficients	P-Value	Standard Error	Wald Statistic	Z-Value
Data	Bias	Iris-setosa	0.8785	0.2315	0.7342	1.4319	1.1966
Coefficients	sepal_length	Iris-setosa	-0.0533	0.1852	0.0402	1.755	-1.3248
P-Value	sepal_width	Iris-setosa	0.0254	0.5319	0.0406	0.3907	0.6251
Standard Error	petal_length	Iris-setosa	-0.1502	0.0001	0.0381	15.5375	-3.9418
Wald Statistic	petal_width	Iris-setosa	-0.0625	0.1213	0.0403	2.4009	-1.5495
Z-Value	x0	Iris-setosa	-0.001	0.9807	0.0407	0.0006	-0.0242
Heat Map	x1	Iris-setosa	-0.0002	0.9958	0.0407	0	-0.0053
	x2	Iris-setosa	-0.006	0.8822	0.0407	0.022	-0.1482
	x3	Iris-setosa	-0.0015	0.9701	0.0407	0.0014	-0.0375
	x4	Iris-setosa	-0.0002	0.9961	0.0407	0	-0.0049

- **Attribute:** Displays the name of the independent variable.
- **Dependent Value:** Displays for multinomial logistic regression only. **Dependent Value** shows the specific categorical value for the given regression statistical data. Note that the results have a row for each **Attribute/Dependent Value** pair.
- **Beta/Coefficient:** Also represented as  $\beta$ , **Beta** is the value of the linear model Correlation Coefficient for the natural logarithms of the probability of occurrence of each independent variable in the logistic regression. Note: The Beta is also referred to as the "log scale".
- **Odds Ratio:** The **Odds Ratio** is the primary measure of the strength of a variable's impact on the results in logistic regression (that is, the "odds" of the event happening given the value of the independent variable). It represents a probability ratio of  $P/(1-P)$ , where  $P$  is the probability of an event happening and  $1-P$  is the probability of it not happening. Note that it is actually calculated by taking the  $\beta$  coefficients and finding  $\exp(B)$  or  $e^B$ , which provides useful measure for the strength of the logistic regression's independent variable impact on the outcome result. For example, a  $\beta = .75$  gives an **Odds Ratio** of  $e^{.75} = 2.72$   $.75 = 2.12$  indicating that the probability of a success is twice as likely as the independent variable's value is increased by 1 unit.
  - The **Odds Ratio** is always greater than zero.

- An **Odds Ratio** value of exactly 1 indicates the variable is not predictive or that the odds of a case outcome are equally likely for both groups under comparison.

-  **Note:** Note: The greater the **Odds Ratio** is than 1, the stronger the relationship between the dependent and independent variable in the logistic regression model.

- **SE/Standard Error, or SE:** Represents the standard deviation of the estimated **Coefficient** values from the actual **Coefficient** values for the set of variables. It is best practice to commonly expect + or - 2 standard errors, meaning the actual value should be within two standard errors of the estimated coefficient value. Therefore, the **SE** value should be much smaller than the forecasted **beta/Coefficient value**.
- **Z-value:** Very similar to the T-value displayed for Linear Regressions. As the data set size increases, the T and Z distribution curves become identical. The **Z-value** is the value related to the standard normal deviation of the variable distribution. It compares the **beta/Coefficient** size to the **SE** size of the **Coefficient** and is calculated as follows:  $Z = \beta / SE$ , where  $\beta$  is the estimated beta coefficient in the regression and **SE** is the standard error value for the **Coefficient**. The **SE** value and **Z-value** are intermediary calculations used to derive the following, more interesting **P-value**, so they are not necessarily interesting in and of themselves.
- **P-value:** Calculated based on the **Z-value** distribution curve. It represents the level of confidence in the associated independent variable being relevant to the model, and it is the primary value used for quick assessment of a variable's significance in a logistic regression model. Specifically, it is the probability of still observing the dependent variable's value if the **Coefficient** value for the independent variable is zero (that is, if **P-value** is high, the associated variable is not considered relevant as a correlated, independent variable in the model).
  - A low **P-value** is evidence that the estimated **Coefficient** is not due to measurement error or coincidence, and therefore, is more likely a significant result. Thus, a low **P-value** gives the modeler confidence in the significance of the variable in the model.
  - Standard practice is to not trust **Coefficients** with **P-values** greater than

0.05 (5%). Note: a **P-value** of less than 0.05 is often conceptualized as there being over 95% certainty that the **Coefficient** is relevant. In actuality, this **P-value** is derived from the distribution curve of the Z-statistic - it is the area under the curve outside of + or - 2 standard errors from the estimated **Coefficient** value.

- Note: The smaller the **P-value**, the more meaningful the coefficient or the more certainty over the significance of the independent variable in the logistic regression model.
- **Wald Statistic:** Used to assess the significance of the Correlation **Coefficients**. It is the ratio of the square of the regression coefficient to the square of the standard error of the coefficient as follows. The **Wald Statistic** tends to be biased when the data is sparse. It is analogous to the t-test in Linear Regression.

**i Note:** When assessing the **Data** tab of the Logistic Regression operator, a modeler mostly cares about the odds ratios, which indicate strength of the correlation between the dependent and independent variables, and the **P-values**, which indicate how much not to trust the estimated coefficient measurements.

### Coefficient Results (multinomial logistic regression)

For multinomial logistic regression results, the Correlation **Coefficient** value for each of the specific dependent variable's categorical values is displayed.

RESULTS - Logistic Regression - Hadoop Config							
Summary	Dependent Value	Bias	sepal_length	sepal_width	petal_length	petal_width	x0
Data	Iris-setosa	0.8785	-0.0533	0.0254	-0.1502	-0.0625	-0.001
Coefficients	Iris-versicolor	0.0226	-0.0012	-0.0204	0.0189	0.0012	0.0028
P-Value							
Standard Error							
Wald Statistic							
Z-Value							
Heat Map							

### P-Value Results (multinomial logistic regression)

For multinomial logistic regression results, the **P-Value** for each of the specific dependent variable's categorical values is displayed.

RESULTS - Logistic Regression - Hadoop Config								
Summary	Dependent Value	Bias	sepal_length	sepal_width	petal_length	petal_width	x0	x1
Data	Iris-setosa	0.2315	0.1852	0.5319	0.0001	0.1213	0.9907	0.9958
Coefficients	Iris-versicolor	0.9756	0.9764	0.6155	0.6217	0.9758	0.9444	0.9371
P-Value								
Standard Error								
Wald Statistic								
Z-Value								
Heat Map								

## Standard Error Results (multinomial logistic regression)

For multinomial logistic regression results, the **SE** value for each of the specific dependent variable's categorical values is displayed.

RESULTS - Logistic Regression - Hadoop Config								
Summary	Dependent Value	Bias	sepal_length	sepal_width	petal_length	petal_width	x0	x1
Data	Iris-setosa	0.7342	0.0402	0.0406	0.0381	0.0403	0.0407	0.0407
Coefficients	Iris-versicolor	0.739	0.0402	0.0406	0.0382	0.0403	0.0407	0.0407
P-Value								
Standard Error								
Wald Statistic								
Z-Value								
Heat Map								

## Wald Statistic Results (multinomial logistic regression)

For multinomial logistic regression results, the **Wald Statistic** for each of the specific dependent variable's categorical values is displayed.

RESULTS - Logistic Regression - Hadoop Config								
Summary	Dependent Value	Bias	sepal_length	sepal_width	petal_length	petal_width	x0	x1
Data	Iris-setosa	1.4319	1.755	0.3907	15.5375	2.4009	0.0006	0
Coefficients	Iris-versicolor	0.0009	0.0009	0.2522	0.2436	0.0009	0.0049	0.0062
P-Value								
Standard Error								
Wald Statistic								
Z-Value								
Heat Map								

## Z-Value Results (multinomial logistic regression)

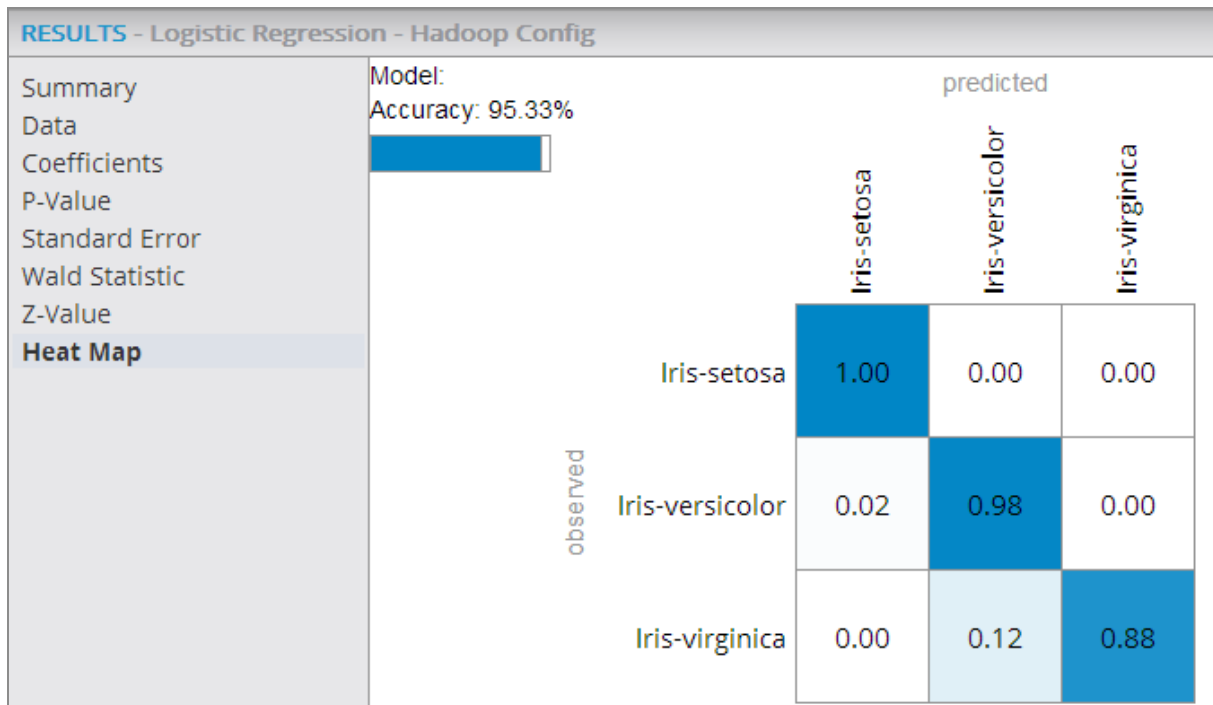
For multinomial logistic regression results, the **Z-Value** for each of the specific dependent variable's categorical values is displayed.

RESULTS - Logistic Regression - Hadoop Config								
Summary	Dependent Value	Bias	sepal_length	sepal_width	petal_length	petal_width	x0	x1
Data	Iris-setosa	1.1966	-1.3248	0.6251	-3.9418	-1.5495	-0.0242	-0.0053
Coefficients	Iris-versicolor	0.0306	-0.0295	-0.5022	0.4935	0.0303	0.0697	0.0789
P-Value								
Standard Error								
Wald Statistic								
Z-Value								
Heat Map								

## Heat Map Results (multinomial logistic regression):

For multinomial logistic regression results, the **Heat Map** displays information about actual vs. predicted counts of a classification model and helps assess the model's accuracy for each of the possible class values.

In the following example, the **Heat Map** shows an overall model accuracy of 95.33% with the highest prediction accuracy being for the class value "Iris-setosa" (100% accurate predictions) versus the lowest being for the "Iris-virginica" (88% accurate predictions).



To learn more about the visualization available in this operator, see [Explore Visual Results](#).

### Data Output

A file with structure similar to the visual output structure is available.

## Logistic Regression (HD)

The Logistic Regression operator fits an s-curve logistic or logit function to a data set to calculate the probability of the occurrence of a specific categorical event based on the values of a set of independent variables.



## Information at a Glance

Parameter	Description
Category	Model
Data source type	HD
Send output to other operators	Yes
Data processing tool	n/a



**Note:** The Logistic Regression (HD) operator is for Hadoop data only. For database data, use the [Logistic Regression \(DB\)](#) operator.

For more detailed information on logistic regression, use cases, and this operator, see [Probability Calculation Using Logistic Regression](#).

## Algorithm

The Hadoop implementation for logistic regression implements a multinomial logistic regression (MLOR) algorithm (and a Regularization Penalizing Parameter to avoid over fitting a model with too many variables). Multinomial logistic regression refers to the instance in which the criterion can take on three or more possible outcomes (for example, "better" vs. "no change" vs. "worse"). TIBCO Data Science - Team Studio can run a multinomial regression when there are more than two categorical values for the dependent variable.

For multinomial logistic regression, the Logistic Regression operator computes a probability for a categorical dependent variable, indicating the probability of each of the class values events occurring (such as chances of having blue eyes). The algorithm uses the statistical fitting of a multinomial logit function to a data set to calculate the probability of the occurrence of a multi-category dependent variable that allows two or more discrete outcomes. For multinomial logistic regression, the dependent variable must be nominal (or categorical, meaning that it falls into any one of a set of categories that cannot be ordered in any meaningful way).

The TIBCO Data Science - Team Studio multinomial logistic regression implementation allows for a Regularization Penalty Parameter that can be applied to prevent the chances of over fitting the model.

**i Note:** Multinomial regressions are not currently supported in the database implementation, only in the Hadoop implementation.

## Input

A data set that contains the dependent and independent variables for modeling.

### Bad or Missing Values

Predictions are made only for rows that contain data. Rows with missing data are skipped.

## Configuration

As of TIBCO Data Science - Team Studio 6.3, the Logistic Regression operator searches the parameter space of  $\lambda$  and  $\alpha$  and automatically select the highest performing model. To use this feature, give either a comma-separated list (for example, .1,.2,.3) for  $\lambda$  or  $\alpha$ , or start:end:step (for example, 0:1:.1). The operator computes all possible  $\lambda$  and  $\alpha$  combinations, and the output from the operator is the model with the highest classification performance. The results of every parameter combination are visible in the results console under the **Parameter optimization** results tab.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	The quantity to model or predict. A dependent column specified for the logistic regression. Select the data column to be considered the dependent variable for the regression. The Dependent Column is often a categorical type, such as eye color = blue, green, or brown.



Parameter	Description
	<p><b>Note:</b> For binomial logistic regression, the <b>Dependent Column</b> must be able to be categorized for a "Yes", "No" prediction (that is, it cannot have more than two distinct values) while for multinomial logistic regression, the <b>Dependent Column</b> can have multiple categorical values to predict.</p>
<b>Maximum Number of Iterations</b>	<p>Specifies the total number of regression iterations that are processed before the algorithm stops if the coefficients do not converge, or show relevance. The possible range for Maximum Number of Iterations must be a non-decimal value <math>\geq 1</math>.</p> <p>Default value: <b>10</b>.</p>
<b>Tolerance</b>	<p>Logistic regression requires a tolerance value to be specified. This is used to determine the maximum allowed error value for the IRLS calculation method. When the error is smaller than this value, the logistic regression model training stops. The possible range for tolerance must be a decimal value <math>\geq 0</math>.</p> <p>Default value: <b>0.0001</b>.</p>
<b>Columns</b>	<p>Specifies the independent variable data columns to include for the regression analysis or model training. At least one <b>Column</b> or one <b>Interaction</b> variable must be specified.</p> <p>Click the <b>Columns</b> button to open the dialog for selecting the available columns from the input data set for analysis. For more information, see <a href="#">Select Columns dialog</a>.</p>
<b>Interaction Parameters</b>	<p>Enables the selection of available independent variables as those data parameters thought to have combined effect on the dependent variable.</p> <p>Creating <b>Interaction Parameters</b> is useful when you believe the combined interaction of two independent variables is not additive.</p> <p>To define an <b>Interaction Parameter</b>, click the <b>Interaction Parameters</b> button and select the suspected interacting data columns.</p> <p>If you have feature A and feature B, selecting * uses both A, B, and the</p>

Parameter	Description
	interaction A*B as independent features. Selecting : means that only A*B are used in the model.
<b>Use approximation (faster)</b>	<p>Implements a single pass of the Hadoop Management System, allowing for a faster, although not as accurate, modeling process.</p> <p><b>Note:</b> Approximation process essentially partitions the data into smaller amounts of data in order to churn the model in memory and average out the outcome.</p> <p>Default value: <b>no</b>.</p>
<b>Use Regularization</b>	<p>Indicates that you can use an optimization parameter for logistic regression as with the <b>Penalizing Parameter</b> below.</p> <p>Default value: <b>no</b>.</p>
<b>Penalizing Parameter (<math>\lambda</math>)</b>	<p>Represents an optimization parameter for logistic regression. It implements regularization of the trade-off between model bias (significance of loss function) and the regularization portion of the minimization function (variance of the regression correlation coefficients). The value can be any number greater than 0.</p> <p>The higher the Lambda, the lower the chance of overfitting with too many redundant variables. (Overfitting is the situation in which a model does a good job "learning" or converges to a low error for the training data but does not do as well for new, non-training data.)</p> <p>For Linear Regression equations, it is possible to use cross-validation process to pick the best Lambda value. However, for logit non-linear regression equations, Lambda should be defined based on user experience with the data. For example, if a model does well on the training data but does not perform well on the predictions, you might want to increase the Lambda value.</p> <p>Default value: <b>0</b>, which indicates no penalty.</p>
<b>Use Regularization</b>	To retrain your model for feature selection and variable importance, enable this option as well as <b>Use Regularization</b> .

Parameter	Description
<b>for Feature Selection Only</b>	<ol style="list-style-type: none"> <li>1. We train a model on the data set with regularization, from which we determine what features to drop.</li> <li>2. A new ordinary regression is trained with those chosen features, obtaining a model with p-values.</li> </ol> <p>The output summary lists any features that were removed and the p-values are included for the second (re-trained) model.</p> <p>Default value: <b>No</b>.</p>
<b>Elastic Parameter (<math>\alpha</math>)</b>	<p>A constant value between 0 and 1 that controls the degree of the mix between L1 (Lasso) and L2 (Ridge) regularization. It is the <math>\alpha</math> parameter in the Elastic Net Regularization loss function:</p> $\lambda \sum_{j=1}^p \left[ \frac{(1-\alpha)}{2} \beta_j^2 + \alpha  \beta_j  \right]$ <p>The <b>Elastic Parameter</b> combines the effects of both the Ridge and Lasso penalty constraints. Both types of penalties shrink the values of the correlation coefficients.</p> <ul style="list-style-type: none"> <li>• When <b>Elastic Parameter (<math>\alpha</math>)</b> = 1, it becomes pure L1 Regularization (Lasso). <ul style="list-style-type: none"> <li>◦ A Lasso constraint tends to remove redundant variables and therefore results in a sparse coefficient model, which is useful when there is high dimensionality in the data.</li> </ul> </li> <li>• When the <b>Elastic Parameter (<math>\alpha</math>)</b> = 0, it becomes pure L2 Regularization (Ridge). <ul style="list-style-type: none"> <li>◦ A Ridge constraint tends to make the variables have similar correlation coefficients and is useful when the data has few variables, or low dimensionality.</li> </ul> </li> <li>• When the <b>Elastic Parameter (<math>\alpha</math>)</b> falls between 0~1, it implements a mix of both L1 (Lasso) and L2 (Ridge) constraints on the coefficients.</li> </ul> <p>Default value:1 (<b>L1</b> or Lasso Regularization). A value of 0.5 would implement a compromise between the Lasso and Ridge constraints.</p>

Parameter	Description
	The <b>Elastic Parameter (<math>\alpha</math>)</b> parameter only applies when <b>Use Spark</b> is set to <b>Yes</b> .
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

**i Note:** In addition to the required logistic regression Prediction operator, it is helpful for the modeler to add Model Validation operators to get further model accuracy statistics (from the Goodness of Fit operator) and/or visual outputs (from the ROC and LIFT operators). See the [Model Validation Operators operators](#) section for details.

## Visual Results

The **Summary** tab displays the Iteration count, Deviance, Null Deviance, Chi-squared value, and Fraction of Variance Explained statistical values.

RESULTS - Logistic Regression- Database Config	
<b>Summary</b>	Number of iterations: 6
Data	Deviance: 14366.2950
	Null deviance: 18528.9167
	Chi-squared value: 4162.6217
	Fraction of variance explained: 0.2247

- **Number of iterations:** Indicates the number of times the logistic regression re-

weighting process was run. When **Iteration = Maximum Number of Iterations**, it flags that the regression might not have yet converged or there was a fit failure (that is, no correlation pattern was uncovered).

- **Deviance:** Used as a statistic for overall fit of a logistic regression model. However, this number is meaningless by itself - it should be compared against the **Null deviance** value below or with its own value from previous runs of the regression.
  - **Deviance** is the comparison of the observed values,  $Y$ , to the expected values  $Y$  predicted.
  - The bigger the difference or **Deviance** of the observed values from the expected values, the poorer the fit of the model.
  - As more independent variables are added to the model, the deviance should get smaller, indicating improvement of fit.
- **Null deviance:** Indicates the deviance of a "dumb" model - a random guess of yes/no without any predictor variables.
  - It is used as a comparison of performance against the model **Deviance** above.
  - The smaller the model **Deviance** (using predictors) can be made relative to the **Null deviance** (no predictors), the better the logistic regression model.
- **Chi-squared value:** The difference between the **Null deviance** and **Deviance**. **Chi-square** technically represents the "negative two log likelihood" or -2LL deviance statistic for measuring the logistic regression's effectiveness. **Chi square = Null deviance minus Deviance**.
  - The hope is for the **Deviance** to be less than the **Null deviance**. Another flag for the logistic regression model not converging or there being a fit failure is having the **Deviance > Null deviance**, or a negative chi square. This might indicate that there is a subset of the data that is over fit on - the modeler could try removing variables and rerunning the regression.

○

**Note:** Since the chi square is a measure of the difference between predicted and actual, it is similar to looking at the residuals for a Linear Regression.

- **Fraction of variance explained:** The **Chi-squared value** divided by the **Null deviance**. This ratio provides a very useful diagnostic statistic representing the percentage of the system's variation the model explains (compared to a dumb model). When analyzing logistic regression results, looking at the **Chi-squared/Null deviance** value provides a similar statistic to the R2 value for Linear Regressions. As a rule of thumb, any **Chi-squared/Null deviance** value over .8 (80%) is considered a successful logistic regression model fit.

## Data Results


The **Data** tab displays the statistical fit numbers for each independent variable in the model.

RESULTS - Logistic Regression- Database Config							
Summary	Attribute	beta	Odds Ratio	SE	Z-value	P-value	Wald
<b>Data</b>	Bias (offset)	-4.3961	N/A	0.1219	-36.0738	1.372e-67	1,301.3193
	times90dayslate	2.1256	8.3777	0.0573	37.1248	9.733e-69	1,378.2501
	revolving_util	-0.1255	0.8821	0.1475	-0.8511	0.3947	0.7244
	debt_ratio	2.3933	10.9495	0.0667	35.8949	2.169e-67	1,288.4431
	credit_lines	-0.0298	0.9706	0.0195	-1.5276	0.1266	2.3335
	monthly_income	0	1	0	-3.4634	0.0005	11.9952
	times30dayslate_2years	-0.0096	0.9904	0.0477	-0.202	0.8399	0.0408

RESULTS - Logistic Regression - Hadoop Config							
Summary	Attribute	Dependent Value	Coefficients	P-Value	Standard Error	Wald Statistic	Z-Value
<b>Data</b> Coefficients P-Value Standard Error Wald Statistic Z-Value Heat Map	Bias	Iris-setosa	0.8785	0.2315	0.7342	1.4319	1.1966
	sepal_length	Iris-setosa	-0.0533	0.1852	0.0402	1.755	-1.3248
	sepal_width	Iris-setosa	0.0254	0.5319	0.0406	0.3907	0.6251
	petal_length	Iris-setosa	-0.1502	0.0001	0.0381	15.5375	-3.9418
	petal_width	Iris-setosa	-0.0625	0.1213	0.0403	2.4009	-1.5495
	x0	Iris-setosa	-0.001	0.9807	0.0407	0.0006	-0.0242
	x1	Iris-setosa	-0.0002	0.9958	0.0407	0	-0.0053
	x2	Iris-setosa	-0.006	0.8822	0.0407	0.022	-0.1482
	x3	Iris-setosa	-0.0015	0.9701	0.0407	0.0014	-0.0375
	x4	Iris-setosa	-0.0002	0.9961	0.0407	0	-0.0049

- **Attribute:** Displays the name of the independent variable.
- **Dependent Value:** Displays for multinomial logistic regression only. **Dependent Value** shows the specific categorical value for the given regression statistical data. Note that the results have a row for each **Attribute/Dependent Value** pair.
- **Beta/Coefficient:** Also represented as  $\beta$ , **Beta** is the value of the linear model

Correlation Coefficient for the natural logarithms of the probability of occurrence of each independent variable in the logistic regression. Note: The **Beta** is also referred to as the "log scale".

- **Odds Ratio:** The **Odds Ratio** is the primary measure of the strength of a variable's impact on the results in logistic regression (that is, the "odds" of the event happening given the value of the independent variable). It represents a probability ratio of  $P/(1-P)$ , where  $P$  is the probability of an event happening and  $1-P$  is the probability of it not happening. Note that it is actually calculated by taking the  $\beta$  coefficients and finding  $\exp(B)$  or  $e^B$ , which provides useful measure for the strength of the logistic regression's independent variable impact on the outcome result. For example, a  $\beta = .75$  gives an **Odds Ratio** of  $e^{.75} = 2.72$   $.75 = 2.12$  indicating that the probability of a success is twice as likely as the independent variable's value is increased by 1 unit.
  - The **Odds Ratio** is always greater than zero.
  - An **Odds Ratio** value of exactly 1 indicates the variable is not predictive or that the odds of a case outcome are equally likely for both groups under comparison.
  -  **Note:** Note: The greater the **Odds Ratio** is than 1, the stronger the relationship between the dependent and independent variable in the logistic regression model.
- **SE/Standard Error, or SE:** Represents the standard deviation of the estimated **Coefficient** values from the actual **Coefficient** values for the set of variables. It is best practice to commonly expect + or - 2 standard errors, meaning the actual value should be within two standard errors of the estimated coefficient value. Therefore, the **SE** value should be much smaller than the forecasted **beta/Coefficient value**.
- **Z-value:** Very similar to the T-value displayed for Linear Regressions. As the data set size increases, the T and Z distribution curves become identical. The **Z-value** is the value related to the standard normal deviation of the variable distribution. It compares the **beta/Coefficient** size to the **SE** size of the **Coefficient** and is calculated as follows:  $Z = \beta/SE$ , where  $\beta$  is the estimated beta coefficient in the regression and **SE** is the standard error value for the **Coefficient**. The **SE** value

and **Z-value** are intermediary calculations used to derive the following, more interesting **P-value**, so they are not necessarily interesting in and of themselves.

- **P-value:** Calculated based on the **Z-value** distribution curve. It represents the level of confidence in the associated independent variable being relevant to the model, and it is the primary value used for quick assessment of a variable's significance in a logistic regression model. Specifically, it is the probability of still observing the dependent variable's value if the **Coefficient** value for the independent variable is zero (that is, if **P-value** is high, the associated variable is not considered relevant as a correlated, independent variable in the model).
  - A low **P-value** is evidence that the estimated **Coefficient** is not due to measurement error or coincidence, and therefore, is more likely a significant result. Thus, a low **P-value** gives the modeler confidence in the significance of the variable in the model.
  - Standard practice is to not trust **Coefficients** with **P-values** greater than 0.05 (5%). Note: a **P-value** of less than 0.05 is often conceptualized as there being over 95% certainty that the **Coefficient** is relevant. In actuality, this **P-value** is derived from the distribution curve of the Z-statistic - it is the area under the curve outside of + or - 2 standard errors from the estimated **Coefficient** value.
  - Note: The smaller the **P-value**, the more meaningful the coefficient or the more certainty over the significance of the independent variable in the logistic regression model.
- **Wald Statistic:** Used to assess the significance of the Correlation **Coefficients**. It is the ratio of the square of the regression coefficient to the square of the standard error of the coefficient as follows. The **Wald Statistic** tends to be biased when the data is sparse. It is analogous to the t-test in Linear Regression.

**i Note:** When assessing the Data tab of the Logistic Regression operator, a modeler mostly cares about the Odds Ratios, which indicate strength of the correlation between the dependent and independent variables, and the P-values, which indicate how much not to trust the estimated coefficient measurements.



## Coefficient Results (multinomial logistic regression)

For multinomial logistic regression results, the Correlation **Coefficient** value for each of the specific dependent variable's categorical values is displayed.

RESULTS - Logistic Regression - Hadoop Config							
Summary	Dependent Value	Bias	sepal_length	sepal_width	petal_length	petal_width	x0
Data	Iris-setosa	0.8785	-0.0533	0.0254	-0.1502	-0.0625	-0.001
Coefficients	Iris-versicolor	0.0226	-0.0012	-0.0204	0.0189	0.0012	0.0028
P-Value							
Standard Error							
Wald Statistic							
Z-Value							
Heat Map							

## P-Value Results (multinomial logistic regression)

For multinomial logistic regression results, the **P-Value** for each of the specific dependent variable's categorical values is displayed.

RESULTS - Logistic Regression - Hadoop Config								
Summary	Dependent Value	Bias	sepal_length	sepal_width	petal_length	petal_width	x0	x1
Data	Iris-setosa	0.2315	0.1852	0.5319	0.0001	0.1213	0.9807	0.9958
Coefficients	Iris-versicolor	0.9756	0.9764	0.6155	0.6217	0.9758	0.9444	0.9371
P-Value								
Standard Error								
Wald Statistic								
Z-Value								
Heat Map								

## Standard Error Results (multinomial logistic regression)

For multinomial logistic regression results, the **Standard Error** value for each of the specific dependent variable's categorical values is displayed.

RESULTS - Logistic Regression - Hadoop Config								
Summary	Dependent Value	Bias	sepal_length	sepal_width	petal_length	petal_width	x0	x1
Data	Iris-setosa	0.7342	0.0402	0.0406	0.0381	0.0403	0.0407	0.0407
Coefficients	Iris-versicolor	0.739	0.0402	0.0406	0.0382	0.0403	0.0407	0.0407
P-Value								
Standard Error								
Wald Statistic								
Z-Value								
Heat Map								

## Wald Statistic Results (multinomial logistic regression)

For multinomial logistic regression results, the **Wald Statistic** for each of the specific dependent variable's categorical values is displayed.

RESULTS - Logistic Regression - Hadoop Config								
Summary	Dependent Value	Bias	sepal_length	sepal_width	petal_length	petal_width	x0	x1
Data	Iris-setosa	1.4319	1.755	0.3907	15.5375	2.4009	0.0006	0
Coefficients	Iris-versicolor	0.0009	0.0009	0.2522	0.2436	0.0009	0.0049	0.0062
P-Value								
Standard Error								
Wald Statistic								
Z-Value								
Heat Map								

## Z-Value Results (multinomial logistic regression)

For multinomial logistic regression results, the **Z-Value** for each of the specific dependent variable's categorical values is displayed.

RESULTS - Logistic Regression - Hadoop Config								
Summary	Dependent Value	Bias	sepal_length	sepal_width	petal_length	petal_width	x0	x1
Data	Iris-setosa	1.1966	-1.3248	0.6251	-3.9418	-1.5495	-0.0242	-0.0055
Coefficients	Iris-versicolor	0.0306	-0.0295	-0.5022	0.4935	0.0303	0.0697	0.0789
P-Value								
Standard Error								
Wald Statistic								
<b>Z-Value</b>								
Heat Map								

## Heat Map Results (multinomial logistic regression):

For multinomial logistic regression results, the **Heat Map** displays information about actual vs. predicted counts of a classification model and helps assess the model's accuracy for each of the possible class values.

In this example, the **Heat Map** shows an overall model accuracy of 95.33%, with the highest prediction accuracy being for the class value "Iris-setosa" (100% accurate predictions) versus the lowest being for the "Iris-virginica" (88% accurate predictions).

RESULTS - Logistic Regression - Hadoop Config					
Summary Data Coefficients P-Value Standard Error Wald Statistic Z-Value Heat Map	Model:		predicted		
	Accuracy: 95.33%				
	<div></div>				
			Iris-setosa	Iris-versicolor	Iris-virginica
	observed	Iris-setosa	1.00	0.00	0.00
		Iris-versicolor	0.02	0.98	0.00
		Iris-virginica	0.00	0.12	0.88

**Important:** To learn more about the visualization available in this operator, see [Explore Visual Results](#).

## Data Output

A file with structure similar to the visual output structure is available.

## Logistic Regression - MADlib

The binomial Logistic Regression (MADlib) operator models the relationship between a dichotomous dependent variable and one or more predictor variables.



## Information at a Glance

Parameter	Description
Category	Model
Data source type	DB
Send output to other operators	Yes
Data processing tool	MADlib

## Algorithm

- The dependent variable is a Boolean value that can be represented with a Boolean expression.
- (Binomial) logistic regression refers to a stochastic model in which the conditional mean of the dependent dichotomous variable is the logistic function of an affine function of the vector of the independent variables.

- Logistic regression finds the vector of coefficients that maximizes the likelihood of the observations.
- Currently, logistic regression in MADlib can use one of the following three algorithms.
  - Iteratively Reweighted Least Squares
  - A conjugate-gradient approach, also known as Fletcher-Reeves method in the literature, where the Hestenes-Stiefel rule is used to calculate the step size.
  - Incremental gradient descent, also known as incremental gradient methods or stochastic gradient descent in the literature.

See the [Official MADlib documentation](#) for more information.

## Input

A data set that contains the dependent and independent variables for modeling.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>MADlib Schema Name</b>	<p>Schema where MADlib is installed in the database. MADlib must be installed in the same database as the input data set.</p> <p>If a "madlib" schema exists in the database, this parameter defaults to madlib.</p>
<b>Model Output Schema Name</b>	Name of the schema where the output is stored.
<b>Model Output Table</b>	<p>Name of the table that is created to store the regression model. Specifically, the model output table stores: [ group_col_1   group_col_2   ...  ] coef   log_likelihood   std_err   z_stats   p_values   odds_ratios   condition_no   num_iterations</p> <p>See the <a href="#">official MADlib logistic regression documentation</a> for more</p>

Parameter	Description
	information.
<b>Drop If Exists</b>	<ul style="list-style-type: none"> <li>• If <b>Yes</b> (the default), drop the existing table of the same name and create a new one.</li> <li>• If <b>No</b>, stop the flow and alert the user that an error has occurred.</li> </ul>
<b>Dependent Variable</b>	Must be a Boolean value to model or predict. The list of the available data columns for the Regression operator is displayed. Select the data column to be the dependent variable for the regression.
<b>Independent Variables</b>	Specifies the independent variable data columns to include for the regression analysis or model training. You must specify at least one column. Click <b>Select Columns</b> to open the dialog for selecting the available columns from the input data set for analysis.
<b>Grouping Columns</b>	Specifies at least one column to group the input data and build separate regression models for each group. Click <b>Select Columns</b> to open the dialog for selecting the available columns from the input data set for grouping.
<b>Maximum Iterations</b>	The computation stops after the number of iterations is greater than the <b>Maximum Iterations</b> or the difference between log-likelihood values in successive iterations is less than the <b>Convergence Tolerance</b> .
<b>Optimizer</b>	<p>Computes the model, which can be one of the following algorithms.</p> <ul style="list-style-type: none"> <li>• <b>Iteratively Reweighted Least Squares</b></li> <li>• <b>Conjugate-Gradient</b>, also known as Fletcher-Reeves method in the literature, where the Hestenes-Stiefel rule for calculating the step size is used.</li> <li>• <b>Incremental Gradient Descent</b>, also known as incremental gradient methods or stochastic gradient descent in the literature.</li> </ul>
<b>Convergence Tolerance</b>	The difference between log-likelihood values in successive iterations that indicate convergence. A zero disables the convergence criterion, so that execution stops after the maximum number of iterations is complete, as set in <b>Maximum Iterations</b> .

Parameter	Description
<b>Verbosity</b>	Set to <b>true</b> (the default) to log all SQL console output of the results of training.

## Output

### Visual Output

Output is displayed in a single tab. For further output and assessment of the quality of the Logistic Regression model, add [ROC](#) and [Lift \(DB\)](#) operators, in addition to the required Logistic Regression Prediction operator.

The Logistic Regression (MADlib) operator output includes the coefficients (**beta**) of the model, the **Odds Ratio**, the standard error (**SE**), the **Z-value**, and the **P-value** statistics.

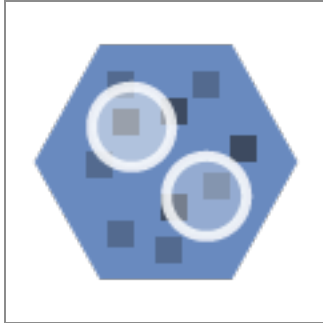
Result Log	dt_abalone	Variable	Logistic Regression (MADlib)		
Iteration times: 5 Deviance: 5239.5380					
Attribute	beta	Odds Ratio	SE	Z-value	P-value
length	-7.55920313	0.00052129	1.62595139	-4.64909541	0.00000333
diameter	3.39060897	29.68402348	2.15625731	1.57245101	0.11584599
height	-0.88396967	0.41313962	1.61484824	-0.54740108	0.58410321
whole	0.29380939	1.34152817	0.70360231	0.41757877	0.67625512
shucked	2.95286016	19.16067808	0.84200727	3.50692954	0.00045331
viscera	1.36640784	3.92123966	1.23855243	1.10322972	0.26992742
shell	0.13432277	1.14376193	1.06600208	0.12600611	0.89972709
rings	0.05785448	1.05956080	0.01452990	3.98175432	0.00006841

### Data Output

None. This is a terminal operator.

## Naive Bayes (DB)

The Naive Bayes operator calculates the probability of a particular event occurring. It can be used to predict the probability of a certain data point being in a particular classification.



## Information at a Glance

Parameter	Description
Category	Model
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

**i Note:** The Naive Bayes (Database) operator is for database data only. For Hadoop data, use the [Naive Bayes \(HD\)](#) operator.

## Algorithm

The Naive Bayes classifier calculates the probability of an event occurring. It combines Bayes' theorem with an assumption of strong independence among the predictors. Bayes' theorem calculates the probability of occurrence given a prior event has occurred. Regardless of actuality, a Naive Bayes classifier considers the influence of predictors on the outcome independently.

- The TIBCO Data Science - Team Studio Naive Bayes Operator computes the dependent variable's class priors and each of the independent variables' probability distributions using the Naive Bayes' conditional probability theorem, with the independence assumption.

- As an overview, the Naive Bayes conditional probability theorem says that, given a data set,  $X$ , and an outcome Hypothesis,  $H$ , the posterior probability that the Hypothesis is true is proportional to the product of the likelihood multiplied by the prior probability.
- Depending on the precise nature of the probability model, Naive Bayes classifiers can be trained very efficiently in a supervised learning setting.
- Given some data and some hypothesis, the posterior probability that the hypothesis is true is proportional to the product of the likelihood multiplied by the prior probability.
- For simplicity, the "prior probability" is often abbreviated as the "prior" and the "posterior probability" as the "posterior".
- The likelihood brings in the effect of the data, while the prior specifies the belief in the hypothesis before the data was observed.

More formally, the Bayes' formula for conditional probability is represented as

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}, \text{ where}$$

- $P(H|X)$  is the conditional probability of outcome  $H$  happening given condition  $X$
- $P(X|H)$  is the conditional probability of the outcome  $X$  happening given condition  $H$
- $P(H)$  is the prior observed probability of the outcome  $H$  happening
- $P(X)$  is the prior observed probability of the outcome  $X$  happening.

This Bayes formula is helpful because it provides a way to calculate the Posterior Probability,  $P(H|X)$ , from  $P(H)$ ,  $P(X|H)$ , and  $P(X)$ , which can be calculated from historic data.

$$P(X|C_i) = \prod_{k=1}^n P(x_k|C_i)$$

This is the Naive Bayes conditional independence assumption formula.

If the feature is a continuous value, the conditional distribution over the class variable  $C$  is expressed as follows:

$$P(X|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-(x_k - \mu_{C_i})^2 / 2\sigma_{C_i}^2}$$



- This formula describes the ideal normal distribution curve for each independent variable's value. Note: This is a simplification assumption since most of the independent variables are likely to have exactly normal distributions.
- However, the Naive Bayes model predictions are still quite accurate with an acceptable level of confidence.
- The Naive Bayes Operator can accept a dependent column that has two or more discrete categories. Note: if the dependent variable is a numeric integer, each integer is treated as a separate category.
- The independence assumption treats all the predictors or variables as independently related to the outcome.
- The Naive Bayes theorem results give the normal probability curve of each possible categorical value occurring for that variable.

## Input

A data set that contains the dependent and independent variables for modeling. The dependent column must be a text type. To use numeric values, pass the data through the [Numeric to Text \(DB\)](#) operator first.

## Configuration

The Dependent Column must be a categorical (non-numeric) variable. Naive Bayes analysis predicts the odds of an outcome of a categorical variable based on one or more predictor variables. A categorical variable is one that can take on a limited number of values, levels, or categories, such as *valid* or *invalid*.

Unlike Logistic Regression and Decision Tree classifiers, Naive Bayes does not require a Value To Predict specification, because the output for the Naive Bayes operator provides the probability of the event for each observed classification value.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent</b>	A <b>Dependent Column</b> must be specified for the Naive Bayes classifier.

Parameter	Description
<b>Column</b>	<p>Select the data column to consider the dependent variable or class to predict. The <b>Dependent Column</b> must be a categorical (non-numeric) type, such as eye color = blue, green or brown.</p> <p>Integers are accepted, with each integer being treated as a category.</p>
<b>Independent Columns</b>	Select the independent variable data columns to include for the regression analysis or model training. At least one column or one interaction variable must be specified.

## Output

### Visual Output

#### Summary Results

The **Summary** results display the class priors, as follows.

RESULTS - Naive Bayes	
Summary	Class Priors
Data	Prior (1): 0.0456
	Prior (0): 0.9544

**Class Priors** - The priors define the observed historical probability of the various possible classification outcome events for the dependent variable based on the training data for the model. This is helpful information because it shows an overall trend of the data for each possible outcome and allows a quick, intuitive check of the source data.

The modeler can see which of the possible dependent variable values occurred the most and least frequently in the training data.

In the example above, the training data showed a prior 4.56% occurrence of the Dependent Variable Class value being 1, and a 95.44% occurrence of the value being 0.

#### Data Results

The **Data** results display calculated standard deviation curve fit numbers (Means and Standard Deviations) for each independent variable (per dependent variable

outcome) in the model.

RESULTS - Naive Bayes				
Summary	Attribute	"Class"	Mean	Standard Deviation
Data	times90dayslate	1	0.6858	0.6675
	times90dayslate	0	0.1065	0.3276
	revolving_util	1	0.469	0.228
	revolving_util	0	0.3255	0.1597
	debt_ratio	1	0.549	0.3673
	debt_ratio	0	0.3184	0.2538
	credit_lines	1	4.3967	1.275
	credit_lines	0	4.9771	1.3099
	monthly_income	1	3,299.8051	1,746.9892
	monthly_income	0	3,479.7434	1,864.3675
	times30dayslate_2years	1	0.2848	0.5822
	times30dayslate_2years	0	0.1162	0.3746
	age	1	36.7964	12.4095
	age	0	36.7933	11.9944
	num_dep	1	1.3374	1.0471
	num_dep	0	1.3488	1.0565
	edu	1	2.8745	1.5699
	edu	0	2.8896	1.5643

Column	Description
<b>Attribute</b>	<p>The name of the independent variable whose normal distribution curve is being described.</p> <p>When assessing Naive Bayes modeling results, each row in the <b>Data</b> table describes the normal distribution curve for each independent variable, given the specified class value specified for the dependent variable.</p> <p>It provides the Class value for the observed curve and the associated</p>

Column	Description
	<b>Mean</b> and <b>Standard Deviation</b> values of the curve.
<b>Class</b>	<p>Represents each possible value for the dependent variable being predicted. For every possible dependent variable outcome value, the independent variable's normal distribution curve, as defined by its <b>Mean</b> and <b>Standard Deviation</b> values, is presented.</p> <p>For example, for the independent variable that represents <i>times90dayslate</i>, when the dependent variable that represents credit delinquency (<i>srsdlnqncy</i>) is 1 (for true), the <i>times90dayslate</i> has a Mean value of .6785 times 90 days late, but when delinquency is false, it only has a Mean value of .1077 times 90 days late.</p> <p>This makes sense in that the more times a person is 90 days late in paying their credit card bill, there is a better chance they have serious credit card delinquency.</p>
<b>Mean</b>	<p>Represents the average value of the independent variable given the specified Dependent Value class outcome. The modeler should compare the Means of an independent variable across the different Class values. If there is a big difference in the Means (with respect to the Standard Deviation value), that particular variable is a stronger predictor of the dependent variable.</p> <p><b>Caution:</b> If the Standard Deviation exceeds the Mean, then the Mean value, although the best possible, is not significant. However, as a rule of thumb, if the Standard Deviation is less than the square root of the Mean, the Mean is a useful measure of a variable's significance.</p> <p><b>Note:</b> if there is little or no difference in Means across Class values (that is, full overlap of the normal distribution curves), assuming small standard deviations, then that associated independent variable is likely not significant in the model.</p> <p>If, for example, an age variable in the credit delinquency model above</p>

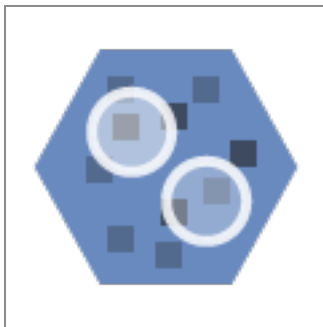
Column	Description
	<p>were to have the same Mean value whether the person is delinquent or not delinquent (with only a slight difference in the standard deviation of the curves), it would seem that a person's age was not a strong predictor of whether that person is delinquent on their credit card payment.</p>
<b>Standard Deviation</b>	<p>Represents the standard deviation of the independent variable value from the Mean given the Dependent Value class outcome specified. It tells how spread out the normal distribution curve is for that particular variable and a given class outcome.</p> <p>Smaller Standard Deviations indicate a smaller range of independent variable values for a given Class value.</p> <p>For each variable, the modeler should understand how the Standard Deviation values compare to the Mean values. A variable's Means might be different for different Class outcomes, but if the Standard Deviation is large, the normal distribution curves significantly overlap for any Class outcome, and therefore the variable is not really a good predictor in the Naive Bayes model.</p> <p>Therefore, smaller Standard Deviations make the Means value a stronger indicator of whether a variable is relevant to the model. Additionally, the larger the Standard Deviation is in comparison to the Mean, the less confidence there is in the Mean.</p> <p>For example, for the <i>monthly_income</i> variable, the Standard Deviation is over half the value of the Mean income, which seems to indicate that there is a large fluctuation in a person's monthly income when the person is both delinquent and not delinquent. The conclusion might be that <i>monthly_income</i> is a weak predictor of credit delinquency.</p> <p>Large standard deviations could also be caused by random error (that is, natural noise) or systematic error (that is, poor data quality).</p> <p>In summary, the more overlap between a given variable's normal distribution curves for all the different possible Class outcomes, the less predictive the variable is in a Naive Bayes model.</p>

## Data Output

Naive Bayes model. When creating a Naive Bayes model, the modeler should add Model Validation Operators<sup>1</sup> to get further Naive Bayes model accuracy statistics (from the Goodness of Fit Operator) and/or visual outputs (from the ROC and Lift Operators). The ROC Curve in particular is a useful visual tool for comparing classification models.

## Naive Bayes (HD)

The Naive Bayes operator calculates the probability of a particular event occurring. It can be used to predict the probability of a certain data point being in a particular classification.



## Information at a Glance

Parameter	Description
Category	Model
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark



**Note:** The Naive Bayes (Hadoop) operator is for Hadoop data only. For database data, use the [Naive Bayes \(DB\)](#) operator.

<sup>1</sup>See the Model Validation Operators section for more details.

## Algorithm

The Naive Bayes classifier calculates the probability of an event occurring. it combines Bayes' theorem with an assumption of strong independence among the predictors. Bayes' theorem calculates the probability of occurrence given a prior event has occurred. Regardless of actuality, a Naive Bayes classifier considers the influence of predictors on the outcome independently.

- The TIBCO Data Science - Team Studio Naive Bayes Operator computes the dependent variable's class priors and each of the independent variables' probability distributions using the Naive Bayes' conditional probability theorem, with the independence assumption.
- As an overview, the Naive Bayes conditional probability theorem says that, given a data set,  $X$ , and an outcome Hypothesis,  $H$ , the posterior probability that the Hypothesis is true is proportional to the product of the likelihood multiplied by the prior probability.
- Depending on the precise nature of the probability model, Naive Bayes classifiers can be trained very efficiently in a supervised learning setting.
- Given some data and some hypothesis, the posterior probability that the hypothesis is true is proportional to the product of the likelihood multiplied by the prior probability.
- For simplicity, the "prior probability" is often abbreviated as the "prior" and the "posterior probability" as the "posterior".
- The likelihood brings in the effect of the data, while the prior specifies the belief in the hypothesis before the data was observed.

More formally, the Bayes' formula for conditional probability is represented as

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}, \text{ where}$$

- $P(H|X)$  is the conditional probability of outcome  $H$  happening given condition  $X$
- $P(X|H)$  is the conditional probability of the outcome  $X$  happening given condition  $H$
- $P(H)$  is the prior observed probability of the outcome  $H$  happening
- $P(X)$  is the prior observed probability of the outcome  $X$  happening.

This Bayes formula is helpful because it provides a way to calculate the Posterior Probability,  $P(H|X)$ , from  $P(H)$ ,  $P(X|H)$ , and  $P(X)$ , which can be calculated from historic data.

$$P(X|C_i) = \prod_{k=1}^n P(x_k|C_i)$$

This is the Naive Bayes conditional independence assumption formula.

If the feature is a continuous value, the conditional distribution over the class variable C is expressed as follows:

$$P(X|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-(x_k - \mu_{C_i})^2 / 2\sigma_{C_i}^2}$$

- This formula describes the ideal normal distribution curve for each independent variable's value. Note: This is a simplification assumption since most of the independent variables are likely to have exactly normal distributions.
- However, the Naive Bayes model predictions are still quite accurate with an acceptable level of confidence.
- The Naive Bayes Operator can accept a dependent column that has two or more discrete categories. Note: if the dependent variable is a numeric integer, each integer is treated as a separate category.
- The independence assumption treats all the predictors or variables as independently related to the outcome.
- The Naive Bayes theorem results give the normal probability curve of each possible categorical value occurring for that variable.

## Input

A data set that contains the dependent and independent variables for modeling.

## Configuration

In Naive Bayes configuration, **Dependent Column** is the value in the data set that is the predicted dependent variable, or "class" variable. **Column(s)** are the expected independent variable data columns, or parameters, to use for model training.

The **Dependent Column** must be a categorical (non-numeric) variable. Naive Bayes analysis predicts the odds of an outcome of a categorical variable based on one or more



predictor variables. A categorical variable is one that can take on a limited number of values, levels, or categories, such as *valid* or *invalid*.

Unlike Logistic Regression and Decision Tree classifiers, Naive Bayes does not require a Value To Predict specification, because the output for the Naive Bayes operator provides the probability of the event for each observed classification value.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	<p>A <b>Dependent Column</b> must be specified for the Naive Bayes classifier. Select the data column to consider the dependent variable or class to predict. The <b>Dependent Column</b> must be a categorical (non-numeric) type, such as eye color = blue, green or brown.</p> <p>Integers are accepted, with each integer being treated as a category.</p>
<b>Columns</b>	<p>Select the independent variable data columns to include for the regression analysis or model training.</p> <p>At least one column or one interaction variable must be specified.</p> <ul style="list-style-type: none"> <li>• Either sparse columns (output from the Collapse operator) or non-sparse columns are supported, but not a mix of the two.</li> <li>• The limitations on the data dimensionality are 10 million values for a column of datatype = sparse and a total 4000 independent data columns.</li> </ul>
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.

## Output

### Visual Output

## Summary Results

The **Summary** results display the class priors, as follows.

RESULTS - Naive Bayes	
Summary	Class Priors
Data	Prior (1): 0.0456
	Prior (0): 0.9544

**Class Priors** - The priors define the observed historical probability of the various possible classification outcome events for the dependent variable based on the training data for the model. This is helpful information because it shows an overall trend of the data for each possible outcome and allows a quick, intuitive check of the source data.

The modeler can see which of the possible dependent variable values occurred the most and least frequently in the training data.

In the example above, the training data showed a prior 4.56% occurrence of the Dependent Variable Class value being 1, and a 95.44% occurrence of the value being 0.

## Data Results

The **Data** results display calculated standard deviation curve fit numbers (Means and Standard Deviations) for each independent variable (per dependent variable outcome) in the model.

RESULTS - Naive Bayes				
Summary	Attribute	"Class"	Mean	Standard Deviation
Data	times90dayslate	1	0.6858	0.6675
	times90dayslate	0	0.1065	0.3276
	revolving_util	1	0.469	0.228
	revolving_util	0	0.3255	0.1597
	debt_ratio	1	0.549	0.3673
	debt_ratio	0	0.3184	0.2538
	credit_lines	1	4.3967	1.275
	credit_lines	0	4.9771	1.3099
	monthly_income	1	3,299.8051	1,746.9892
	monthly_income	0	3,479.7434	1,864.3675
	times30dayslate_2years	1	0.2848	0.5822
	times30dayslate_2years	0	0.1162	0.3746
	age	1	36.7964	12.4095
	age	0	36.7933	11.9944
	num_dep	1	1.3374	1.0471
	num_dep	0	1.3488	1.0565
	edu	1	2.8745	1.5699
	edu	0	2.8896	1.5643

Column	Description
<b>Attribute</b>	<p>The name of the independent variable whose normal distribution curve is being described. When assessing Naive Bayes modeling results, each row in the <b>Data</b> table describes the normal distribution curve for each independent variable, given the specified class value specified for the dependent variable.</p> <p>It provides the Class value for the observed curve and the associated <b>Mean</b> and <b>Standard Deviation</b> values of the curve.</p>
<b>Class</b>	Represents each possible value for the dependent variable being

Column	Description
	<p>predicted.</p> <p>For every possible dependent variable outcome value, the independent variable's normal distribution curve, as defined by its <b>Mean</b> and <b>Standard Deviation</b> values, is presented.</p> <p>For example, for the independent variable that represents <i>times90dayslate</i>, when the dependent variable that represents credit delinquency (<i>srsdlnqncy</i>) is 1 (for true), the <i>times90dayslate</i> has a Mean value of .6785 times 90 days late, but when delinquency is false, it only has a Mean value of .1077 times 90 days late.</p> <p>This makes sense in that the more times a person is 90 days late in paying their credit card bill, there is a better chance they have serious credit card delinquency.</p>
<b>Mean</b>	<p>Represents the average value of the independent variable given the specified Dependent Value class outcome.</p> <p>The modeler should compare the Means of an independent variable across the different Class values. If there is a big difference in the Means (with respect to the Standard Deviation value), that particular variable is a stronger predictor of the dependent variable.</p> <p><b>Note:</b> If the Standard Deviation exceeds the Mean, then the Mean value, although the best possible, is not significant. However, as a rule of thumb, if the Standard Deviation is less than the square root of the Mean, the Mean is a useful measure of a variable's significance.</p> <p><b>Note:</b> If there is little or no difference in Means across Class values (that is, full overlap of the normal distribution curves), assuming small standard deviations, then that associated independent variable is likely not significant in the model.</p> <p>If, for example, an age variable in the credit delinquency model above were to have the same Mean value whether the person is delinquent or</p>

Column	Description
	<p>not delinquent (with only a slight difference in the standard deviation of the curves), it would seem that a person's age was not a strong predictor of whether that person is delinquent on their credit card payment.</p>
<b>Standard Deviation</b>	<p>Represents the standard deviation of the independent variable value from the Mean given the Dependent Value class outcome specified. It tells how spread out the normal distribution curve is for that particular variable and a given class outcome.</p> <p>Smaller Standard Deviations indicate a smaller range of independent variable values for a given Class value.</p> <p>For each variable, the modeler should understand how the Standard Deviation values compare to the Mean values. A variable's Means might be different for different Class outcomes, but if the Standard Deviation is large, the normal distribution curves significantly overlap for any Class outcome, and therefore the variable is not really a good predictor in the Naive Bayes model.</p> <p>Therefore, smaller Standard Deviations make the Means value a stronger indicator of whether a variable is relevant to the model. Additionally, the larger the Standard Deviation is in comparison to the Mean, the less confidence there is in the Mean.</p> <p>For example, for the <i>monthly_income</i> variable, the Standard Deviation is over half the value of the Mean income, which seems to indicate that there is a large fluctuation in a person's monthly income when the person is both delinquent and not delinquent. The conclusion might be that <i>monthly_income</i> is a weak predictor of credit delinquency.</p> <p>Large standard deviations could also be caused by random error (that is, natural noise) or systematic error (that is, poor data quality).</p> <p>In summary, the more overlap between a given variable's normal distribution curves for all the different possible Class outcomes, the less predictive the variable is in a Naive Bayes model.</p>

## Data Output

Naive Bayes model. When creating a Naive Bayes model, the modeler should add Model Validation Operators<sup>1</sup> to get further Naive Bayes model accuracy statistics (from the Goodness of Fit Operator) and/or visual outputs (from the ROC and Lift Operators). The ROC Curve in particular is a useful visual tool for comparing classification models.

## Neural Network

Implements the Spark MLlib MultiLayer Perceptron Classifier (MLPC), a feedforward neural network that consists of multiple layers of nodes in a directed graph, each layer fully connected to the next one in the network.



## Information at a Glance

Parameter	Description
Category	Model
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark



**Note:** This page refers to the Neural Network operator developed in TIBCO Data Science - Team Studio 6.3. The previous version of this operator was deprecated and removed in version 6.1.

The Neural Network operator can be exported and used in other workflows in the same workspace.

---

<sup>1</sup>See the Model Validation Operators section for more details.

Nodes in the input layer represent the input features. All other nodes map inputs to outputs using a linear combination of the inputs, with the node's weights ( $w$ ) and bias ( $b$ ). The nodes apply a nonlinear activation function depending on their types of layers: sigmoid function for nodes in hidden layers; softmax function for nodes in the output layer.

The output layer represents the classes to predict. For learning, the model the output layer uses the backpropagation function.

Because the MLPC requires continuous features as input nodes, categorical features can be used as predictors by first applying One-Hot Encoding (converting categorical columns to columns of binary vectors with a single 1-value).

For more information about the MLPC, see <https://spark.apache.org/docs/1.6.1/ml-classification-regression.html#multilayer-perceptron-classifier> and [https://en.wikipedia.org/wiki/Multilayer\\_perceptron](https://en.wikipedia.org/wiki/Multilayer_perceptron).

## Input

An HDFS data set input, including predictor columns and a dependent column to predict.

### Bad or Missing Values

If a row contains a null value in at least one of the independent columns or the dependent column, the row is removed from the data set. The number of null values removed can be listed in the Summary section of the output (depending on the chosen option for **Write Rows Removed Due to Null Data To File**).

## Restrictions

If the model is exported using the Export operator to the TIBCO Data Science - Team Studio Model format, then this model can be loaded in other workflows in the current workspace using the Load Model operator. Scoring HDFS inputs is supported, but the model is not SQL compatible and does not score against a relational database connection.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	<p>A dependent column that contains the label to predict. This column can be categorical or numeric (each number is considered as a distinct label).</p> <p><b>Note:</b> The dependent column should not contain more than 20 distinct labels; otherwise, an error occurs at runtime).</p>
<b>Columns</b>	<p>The columns to use as independent features to train the model. Both numeric and categorical are supported.</p> <p><b>Note:</b> Because the MLPC requires numeric input features, all categorical features are transformed to binary columns using One Hot Encoder before training the Neural Network.</p> <p>Each categorical column should not contain more than 250 categories; otherwise, a runtime error occurs.</p>
<b>Hidden Layers (Comma-Separated)</b>	<p>The number of hidden layers and number of neurons associated with each layer using an integer comma-separated list. (for example, "6,3" means two hidden layers with six and three neurons respectively).</p> <p>If the default value (-1) is used, the algorithm automatically sets one hidden layer.</p> <p>number of neurons = (nb_neurons_input_layer + nb_neurons_output_layer) / 2.</p> <p><b>Note:</b> If the Info field is left blank, no hidden layers are used: the network consists of an input layer directly connected to the output layer (which is equivalent to a Logistic Regression).</p>
<b>Maximum</b>	The maximum number of iterations.



Parameter	Description
<b>Iterations</b>	Default value: <b>100</b> .
<b>Tolerance</b>	<p>The convergence tolerance of iterations. Smaller values lead to higher accuracy at the cost of more iterations.</p> <p>Default value: <b>10E-4</b>.</p>
<b>Random Seed</b>	The random seed to use to initialize the weights of the Neural Network. Range: $-10000 < \text{seed} < 10000$ . Default value: <b>-1</b> (random initialization).
<b>Compute Classification Metrics on Training Set</b>	<b>Yes</b> (the default) - performance metrics of the model on the input are displayed in the output <b>Tab Training Metrics</b> . The model accuracy is provided as well as for each label to predict the associated recall, precision, and F1 measure.
<b>Write Rows Removed Due to Null Data To File</b>	<p>Rows with null values in at least one of the independent columns or the dependent column are removed from the analysis. Use this parameter to specify that the data with null values be written to a file.</p> <p>The file is written to @default_tempdir/tsds_out/@user_name/@flow_name/@operator_name_uuid/bad_data</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

A visual output with three tabs - **Summary**, **Training Metrics**, and **Neural Network Weights**.

### Summary:

Summary	Parameters Selected	
Training Metrics	Dependent Column:	srslqncy
Neural Network Weights	Input Layer (Number of Features):	{8}
	Hidden Layers:	{5} (auto mode)
	Output Layer (Number of Classes):	{2}
	Maximum Number of Iterations:	100
	Tolerance:	1.0E-4
	Trained Model Statistics	
	Training Time:	51.58 seconds
	Output	
	Input data size:	95460 rows
	Input size after removing rows due to null values in column 'srslqncy' or independent variables selected:	95460 rows
	No data removed due to null values in column 'srslqncy' or independent variables selected	

**Training Metrics** (This tab is displayed only if the option **Compute Classification Metrics on Training Set** was set to **Yes**):

Summary	Label	Accuracy	Recall	Precision	F1
Training Metrics	1	79.42	79.79	79.37	79.58
Neural Network Weights	0	79.42	79.05	79.48	79.26

### Neural Network Weights

This table provides the final input weights of each neuron of the network. The **neuron\_input\_weights** column is shows a list of all neurons connected to the neuron **neuron\_nb** of the layer **layer\_type**, with their associated weight (for example, (1, 0.09) means a weight of 0.09 associated with neuron 1 of the previous layer).

Summary	layer_type	neuron_nb	neuron_input_weights
Training Metrics	hidden_layer_1	1	{{(1,0.09), (2,-0.84), (3,0.07), (4,0.18), (5,0.01), (6,4.64), (7,-1.03), (8,-6.16), (Bias,-5.23)}}
Neural Network Weights	hidden_layer_1	2	{{(1,1.0), (2,-0.97), (3,-0.37), (4,0.59), (5,0.0), (6,0.11), (7,0.23), (8,-1.11), (Bias,-0.43)}}
	hidden_layer_1	3	{{(1,0.4), (2,3.42), (3,1.46), (4,-0.34), (5,0.2), (6,-0.42), (7,1.17), (8,0.54), (Bias,1.71)}}
	hidden_layer_1	4	{{(1,0.19), (2,0.55), (3,0.07), (4,0.04), (5,0.01), (6,-2.62), (7,-1.04), (8,2.96), (Bias,0.42)}}
	hidden_layer_1	5	{{(1,-1.59), (2,-5.96), (3,-0.46), (4,1.98), (5,-2.11), (6,1.67), (7,-2.5), (8,-0.34), (Bias,-1.17)}}
	output_layer	1	{{(1,-3.89), (2,0.01), (3,-0.66), (4,0.94), (5,-0.92), (Bias,0.96)}}
	output_layer	2	{{(1,3.24), (2,0.56), (3,0.2), (4,-1.7), (5,1.11), (Bias,-0.27)}}

### Data Output

A neural network model than can be used for classifying new input data using the Classifier or Predictor operator. Alternatively, it can be connected to classification evaluation operators such as Confusion Matrix, ROC, and Goodness of Fit. It can be

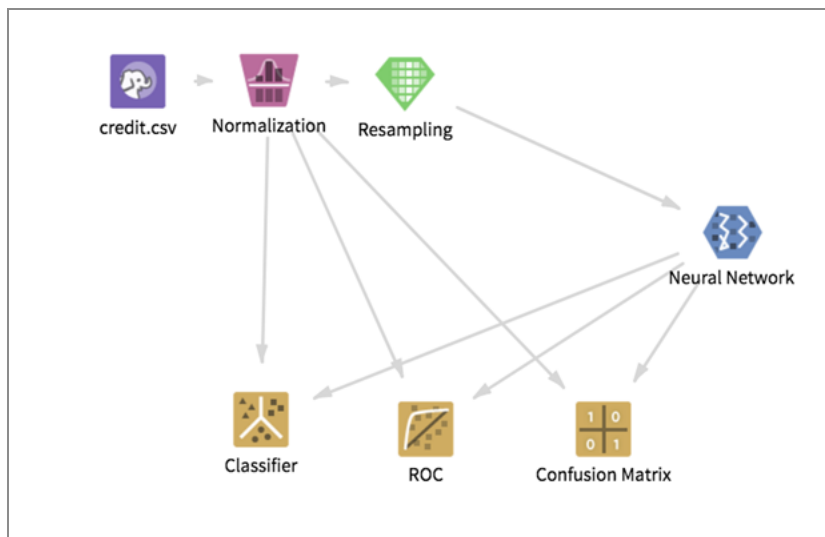
exported using the Export operator (to be reused in other workflows in the current workspace).

## Additional Notes

To improve Neural Network performance, try the following.

- Scale and normalize your input features before training the model to equally distribute the importance of each input and avoid saturating the hidden layers (otherwise the naturally large values become dominant). This also helps the algorithm to converge faster.
- If your training data set is very imbalanced (that is, one label is much more represented than another), resample it so all classes are well represented (you can use the Resampling operator). This is likely to improve your model accuracy.

## Example



## PCA (DB)

Uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of uncorrelated variables (principal components).



## Information at a Glance

Parameter	Description
Category	Model
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

## Algorithm

PCA (Principal Component Analysis) is an orthogonal linear transformation that transforms data into a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, the third on the third coordinate, continuing until the number of rows has been reached or a preset maximum principal component threshold has been reached.

The Alpine PCA operator implements an eigenvalue decomposition of a data covariance matrix  $\Sigma$  (or, correlation matrix  $R$ ).

- Each principal component is a linear combination of the original variables.
- The coefficients (loadings) are the eigenvectors ( $v_1, v_2, \dots, v_p$ ) of covariance matrix  $\Sigma$  (or, correlation matrix  $R$ ) with unit length.
- The eigenvalues ( $\lambda_1, \lambda_2, \dots, \lambda_p$ ) denote the contribution of the principal component associated with it.
- The principal components are sorted by descending order according to their variance contribution.

- The user can choose the number of principal components according to the accumulation contribution ( $\sum_{j=1}^p \lambda_j / \sum_{k=1}^K \lambda_k$ ).

More details are available in *Principal Component Analysis*, (1986), Joliffe, I.T.

Additional references:

- Jerome Friedman, Trevor Hastie, Robert Tibshirani (2008), *The Elements of Statistical Learning Data Mining, Inference and Prediction* Chapter 3: "Linear Methods for Regression"
- Joliffe, I.T. (1986), *Principal Component Analysis*, New York, Springer
- Wu, W., Massart, D.L., and de Jong, S. (1997), "The Kernel PCA Algorithms for Wide Data. Part I: Theory and Algorithms" *Chemometrics and Intelligent Laboratory Systems*, 36, 165-172.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Schema where MADlib is installed in the database. MADlib must be installed in the same database as the input dataset. If a "madlib" schema exists in the database, this parameter defaults to madlib.
<b>Analysis Type</b>	<p>The type of matrix to use to perform the eigenvalue decomposition.</p> <ul style="list-style-type: none"> <li>• <b>COV-POP</b> (the default): Uncorrected covariance matrix. This implements the PCA algorithm against un-centered, un-scaled data.</li> <li>• <b>COV-SAM</b>: Covariance matrix. This implements the PCA algorithm against centered but un-scaled data.</li> <li>• <b>CORR</b>: Correlation matrix. This implements the PCA algorithm against centered and scaled data.</li> </ul>
<b>Percent</b>	The threshold for the fraction of the variance explained with principal components. This decides the number of principal components.

Parameter	Description
	<ul style="list-style-type: none"> <li>The value expected is between 0 and 1.</li> <li>A larger value directly relates to the number of principal components reported.</li> </ul>
<b>Result Output Schema</b>	The schema name of the result output table transformed from the original table.
<b>Result Output Table</b>	The name of the result output table transformed from the original table.
<b>Result Output Table Storage Parameters</b>	<p>For operators that can generate an output table, the Storage Parameters dialog allows you to specify additional parameters regarding storage method and compression.</p> <p>See: <a href="#">Storage Parameters dialog</a></p>
<b>Drop If Exists (Result)</b>	<ul style="list-style-type: none"> <li>If <b>Yes</b> (the default), drop the existing table of the same name and create a new one.</li> <li>If <b>No</b>, stop the flow and alert the user that an error has occurred.</li> </ul>
<b>Eigenvalues Output Schema</b>	The schema name of the output table in which to save the scores of the principal components.
<b>Eigenvalues Output Table</b>	The name of the output table in which to save the scores of the principal components.
<b>Eigenvalues Output Table Storage Parameters</b>	The storage parameters of the output table in which to save the scores of the principal components.
<b>Drop If Exists (Eigenvalues)</b>	<p>Specifies whether to overwrite the existing eigenvalues.</p> <ul style="list-style-type: none"> <li><b>Yes</b> (the default) - If the entry with the name exists, it is dropped before storing the results.</li> <li><b>No</b> - If the entry with the name exists, the results window displays an error message.</li> </ul>

Parameter	Description
<b>Column Names</b>	Click <b>Columns</b> to open the dialog for selecting the available columns from the procedure of PCA.
<b>Carryover Columns</b>	You can choose to keep columns from the input data untransformed and included in the output. To do this, click <b>Carryover Columns</b> to open the dialog for selecting the columns to retain in the result table.

## Output

### Visual Output

#### Results Table

Provides the eigenvalues used in the matrix transformation.

- Initial variable columns: The initial variable columns passed into the PCA operator are displayed, along with a magnitude value for that variable's contribution to the eigenvector transformation into each derived principal component.
- alpine\_pcadataindex**: Eigenvector index number that provides a unique number for each derived principal component.
- alpine\_pcaevalue**: Eigenvalue for that principal component.
- alpine\_pcacumvl**: The fraction of the variability that this eigenvector explains for the principal component defined.
- alpine\_pcatotalcumvl**: The cumulative fraction of the variability that this eigenvector explains for the principal component defined.

RESULTS - PCA Operator - dbase													— T
Results Table	x6	x7	x8	x9	x10	x11	x12	x13	x14	alpine_pcadataindex	alpine_pcaevalue	alpine_pcacumvl	alpine_pcatotalcumvl
Output Table	0.2236	0.4427	0.0599	-0.2313	-0.0736	0.4381	-0.2506	0.1899	-0.4453	2	0.1356	0.0233	0.7981
	-0.4253	0.0606	0.2547	-0.3157	0.2465	0.0098	0.348	0.4859	0.1074	11	0.0737	0.0126	0.9387
	-0.4627	0.1299	-0.0258	0.1745	0.2121	0.1896	-0.4918	0.1116	0.2382	5	0.1112	0.0191	0.9497
	-0.0085	0.0368	-0.5384	-0.3688	-0.214	0.1075	0.421	-0.1049	0.0043	7	0.0977	0.0168	0.8837
	-0.138	-0.4136	0.5462	-0.2403	-0.1294	0.3504	-0.063	-0.2634	-0.005	10	0.0786	0.0135	0.9261
	0.3263	0.0783	-0.0371	-0.2058	0.5048	0.1236	0.0607	-0.4343	0.3688	3	0.1311	0.0224	0.8106

#### Output Table

Provides an overview of the new reduced Principal Components data set.

**alpine\_pcaattr[0-13]+**: Each of the newly derived Principal Components columns is provided, along with their values for the new transformed data set. In this case, the

source Iris data set with hundreds of variables was reduced to only 13 principal component variables and saved as **pcaOperatorResultsIris**. (See the example flow below.)

Carryover columns: any carryover columns from the original data set that were specified in the PCA operator configuration are displayed here, such as any necessary unique ID key or the dependent variable to predict in the following model. In this example, the "class" column was carried over to be used in the following Alpine Forest model.

RESULTS - PCA Operator - dbase												
Results Table	_pcaattr1	alpine_pcaattr4	alpine_pcaattr5	alpine_pcaattr6	alpine_pcaattr7	alpine_pcaattr8	alpine_pcaattr9	alpine_pcaattr10	alpine_pcaattr11	alpine_pcaattr12	alpine_pcaattr13	"class"
Output Table	1.9165	-0.0311	0.2577	0.1408	0.4082	-0.5046	0.33	-0.3559	0.9984	0.4223		Iris-setosa
	1.855	0.6311	-0.0451	0.359	0.3152	-0.8908	0.1397	-0.6705	0.9912	0.2182		Iris-versicolor
	2.1049	0.7705	-0.0874	0.0301	0.2771	-1.1881	0.093	-0.4988	0.7485	0.1975		Iris-virginica
	2.3419	0.2418	0.4523	0.2414	0.2287	-0.7364	-0.1127	-0.6506	0.6521	-0.0897		Iris-setosa
	1.4109	0.5797	0.0833	0.4229	0.2437	-0.9579	0.0471	-0.4233	0.9062	-0.0484		Iris-setosa
	1.8167	0.5189	0.6611	0.0145	0.3668	-0.7076	-0.7879	-0.7457	1.0343	-0.0841		Iris-setosa

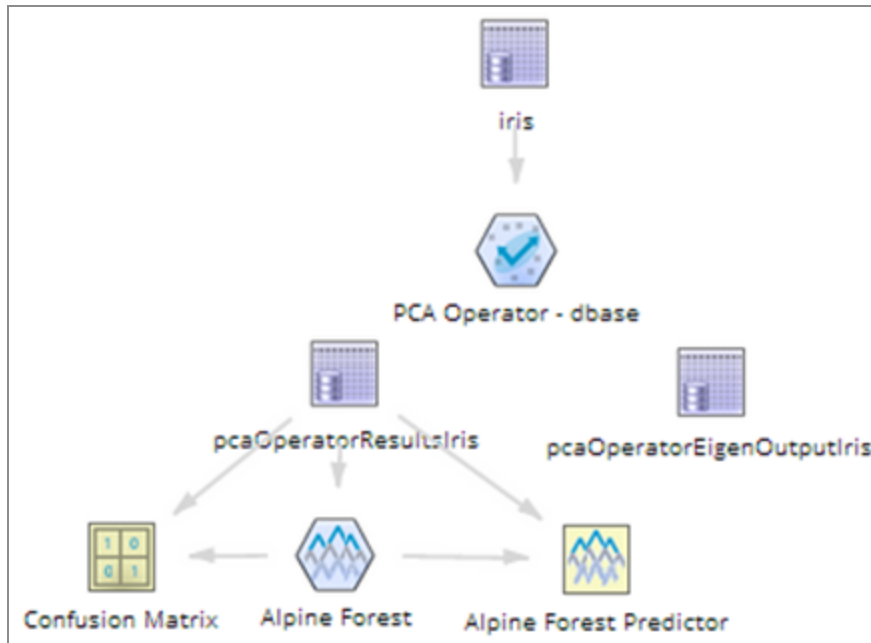
## Data Output

Stored database tables that can be accessed by other operators.

The PCA operator for the database is technically a terminal operator, meaning that no other operator directly follows it in the workflow. However, the PCA operator stores its Principal Component Results (and Eigenvalue Output details) in two database tables that can then be accessed as the data source for a new workflow, if applicable. The example below shows the results of the database PCA operator being saved as **pcaOperatorResultsIris** and **pcaOperatorEigenOutputIris**. The tables can be brought into the workflow and the derived Principal Components can be fed into an Alpine Forest operator, for example, and the classification results analyzed in the Confusion Matrix in order to understand if the reduced set of variables that the PCA operator created provide an accurate enough model.



## Example



## PCA (HD)

Uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of uncorrelated variables (principal components).



## Information at a Glance

Parameter	Description
Category	Model
Data source type	HD

Parameter	Description
Send output to other operators	Yes
Data processing tool	In-memory: MapReduce or Spark, depending on input configuration

## Algorithm

PCA (Principal Component Analysis) is an orthogonal linear transformation that transforms data into a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, the third on the third coordinate, continuing until the number of coordinates has been reached or a preset maximum principal component threshold has been reached.

The Alpine PCA operator implements an eigenvalue decomposition of a data covariance matrix  $\Sigma$  (or, correlation matrix  $R$ ).

- Each principal component is a linear combination of the original variables.
- The coefficients (loadings) are the eigenvectors ( $v_1, v_2, \dots, v_p$ ) of covariance matrix  $\Sigma$  (or, correlation matrix  $R$ ) with unit length.
- The eigenvalues ( $\lambda_1, \lambda_2, \dots, \lambda_p$ ) denote the contribution of the principal component associated with it.
- The principal components are sorted by descending order according to their variance contribution.
- The user can choose the number of principal components according to the accumulation contribution ( $\sum_{j=1}^K \lambda_j / \sum_{p=1}^K \lambda_p$ ).

More details are available in *Principal Component Analysis*, (1986), Joliffe, I.T.

Additional references:

- Jerome Friedman, Trevor Hastie, Robert Tibshirani (2008), *The Elements of Statistical Learning Data Mining, Inference and Prediction* Chapter 3: "Linear Methods for Regression"
- Joliffe, I.T. (1986), *Principal Component Analysis*, New York, Springer

- Wu, W., Massart, D.L., and de Jong, S. (1997), "The Kernel PCA Algorithms for Wide Data. Part I: Theory and Algorithms" *Chemometrics and Intelligent Laboratory Systems*, 36, 165-172.

## Input

A data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	Click <b>Columns</b> to open the dialog for selecting the available variable columns to transform by the PCA algorithm, in order to create a reduced set of variables.
<b>Center</b>	<p>If <b>Yes</b> (the default), the mean value of each variable column is set to 0 before the PCA matrix transformation algorithm is run. In combination with <b>Scale</b>, the following applies.</p> <ul style="list-style-type: none"> <li>• If <b>Yes</b>, and <b>Scale</b> is <b>No</b>, then a Covariance matrix is applied.</li> <li>• If <b>Yes</b>, and <b>Scale</b> is <b>Yes</b>, then a Correlation matrix is applied.</li> <li>• If <b>No</b>, and <b>Scale</b> is <b>No</b>, then an uncorrected covariance matrix is applied.</li> </ul> <p>Usually, the PCA algorithm has meaning only if the data is centered first, so the default is set to <b>Yes</b>.</p> <p>When <b>Use Spark</b> is set to <b>Yes</b>, then this parameter is grayed out.</p>
<b>Scale</b>	<p>When the <b>Scale</b> option is selected, each variable's data values are divided by the Standard Deviation so that all of the columns have the same data spread (that is, to be on an equivalent scale as each other).</p> <p>See <b>Center</b> for more information about its effect on the algorithm.</p>

Parameter	Description
	Default value: <b>Yes</b> . Note that variables with zero Standard Deviation should not be input into PCA. When <b>Use Spark</b> is set to <b>Yes</b> , then this parameter is grayed out.
<b>In Memory Threshold</b>	<p>Determines whether to compute the PCA by a Hadoop MapReduce job or in memory SVD (single machine instead of distributed mode).</p> <p>If the number of rows in the training data set is fewer than the threshold value, the PCA algorithm is run in memory SVD. Otherwise, it is computed by a MapReduce job.</p> <p>When the input parameter <b>Use Spark</b> is set to <b>Yes</b>, then the job is run in Spark regardless of the value of this parameter, which is grayed out.</p> <p><b>Note:</b> The MapReduce option is currently deprecated.</p>
<b>Maximum Number of Components to output.</b>	<p>Determines the upper-limit number of principal components to calculate, starting with the top, high-ranking variance components. This value must be equal to or typically less than the number of columns in the training data set, to help with dimension reduction.</p> <p>The choice of this value depends on the desired cumulative variance to cover. To check whether this number would return enough components, inspect the output tab <b>Variance</b> (when <b>Use Spark</b> is set to <b>Yes</b>) or the output tab <b>Cumulative Variance</b> (when <b>Use Spark</b> is set to <b>No</b>).</p>
<b>Additional Runs for Distributed Mode</b>	<p>Specifies the number of required extra passes of the algorithm to implement when computing in-memory SVD. Typically, a single pass result is sufficient when the number of rows are less than the In Memory Threshold value.</p> <p>Default value: <b>0</b> (no additional runs).</p> <p>When <b>Use Spark</b> is set to <b>Yes</b>, then this parameter is grayed out.</p>
<b>Max JVM Heap Size (MB) (-1=Automatic)</b>	<p>A Java Virtual Machine data storage setting for Hadoop. (Spark option only.)</p> <ul style="list-style-type: none"> <li>The default value is <b>1024</b>.</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>If the value is <b>-1</b>, the system sets the heap size automatically.</li> </ul> <p>When <b>Use Spark</b> is set to <b>Yes</b>, then this parameter is grayed out.</p>
<b>Use Spark</b>	If <b>Yes</b> (the default), uses Spark to optimize calculation time.
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li><b>Yes</b> specifies using the default Spark optimization settings.</li> <li><b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

The visual output for Hadoop provides visualizations of the principal components and their contribution weightings, scaling, centering, variance, and cumulative variance.

### Components

Shows the new principal components as the columns and each row provides the source data variable's contribution to the derived component value.

RESULTS - PCA Operator - Hadoop								
Components	Attribute	Component 1	Component 2	Component 3	Component 4	Component 5	Component 6	Component 7
Variance	sepal_length	0.3612	-0.6236	-0.0743	-0.1977	0.0291	0.1109	-0.0039
Scale	sepal_width	-0.0821	-0.7153	-0.0665	0.0305	-0.1478	-0.0571	0.1008
Center	petal_length	0.8555	0.1709	0.0178	0.0093	0.0193	0.0128	-0.0048
Variances	petal_width	0.3585	0.0647	0.0323	0.134	-0.1403	-0.1415	-0.0027
Cumulative Variance	x0	0.0002	-0.042	0.0188	0.0846	0.3843	0.1764	-0.4545
Features in the Principal Component Space	x1	-0.0015	-0.0017	0.1658	-0.319	0.206	-0.1362	0.3845
	x2	0.0347	-0.0904	0.314	0.1194	0.307	-0.0713	0.1548
	x3	0.0021	0.0923	-0.2101	-0.3132	0.2875	0.0237	0.3988
	x4	0.0008	0.0352	-0.1372	-0.0875	0.0678	-0.4235	-0.277
	x5	-0.0107	-0.0126	-0.004	-0.0925	0.2347	0.2635	0.0749
	x6	-0.0093	-0.0288	-0.2159	-0.2325	-0.1952	-0.4533	-0.1567
	x7	-0.0027	-0.0079	-0.4467	-0.0694	0.1922	0.1096	-0.2495

### Variance

Shows each principal component's contribution to the overall data variance. Variance provides a measure of how much of all the variance in the original data set is captured by a given component. When **Use Spark** is set to **Yes**, this tab also contains the Cumulative Variance as an additional column. The **Variance** column is called **Explained Variance**.

<b>Components</b> <b>Variance</b> <b>Center</b> <b>Scale</b>	Component	Explained Variance	Cumulative Variance
	1	0.8391	0.8391
	2	0.08695	0.926
	3	0.03231	0.9583
	4	0.02075	0.9791
	5	0.01062	0.9897

### Center

Shows if any centering was done to the original data set column to clean up the data before computing the principal components. The data displayed shows the center value of the original data set's columns that were used in the PCA algorithm. Note: If the values are not 0, the data was not centered prior to running the algorithm. It is best to normalize the source data and center it around 0 before running the PCA algorithm.

RESULTS - PCA Operator - Hadoop		
<b>Components</b> <b>Variance</b> <b>Scale</b> <b>Center</b> <b>Variances</b> <b>Cumulative Variance</b> <b>Features in the Principal Component Space</b>	Component	Center
	1	5.8433
	2	3.054
	3	3.7587
	4	1.1987
	5	0.5033
	6	0.5359
	7	0.5196

### Scale

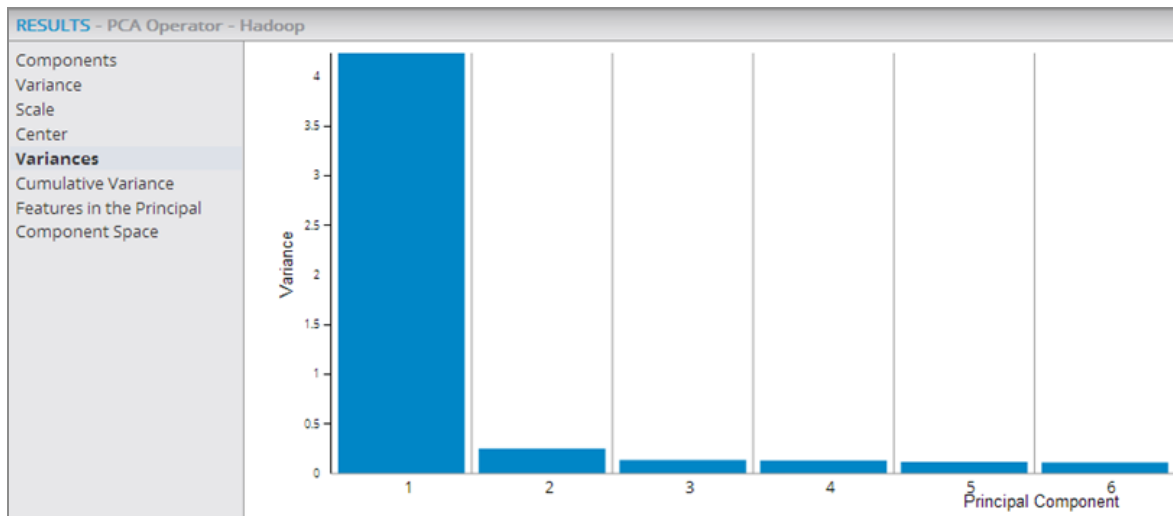
Shows the relative size of the source data set's columns relative to their original values passed in. Note: if the value is not "1" (meaning 100% of its original value), no scaling was done prior to running the PCA algorithm.

RESULTS - PCA Operator - Hadoop		
Components	Component	Scale
Variance	1	1
<b>Scale</b>	2	1
Center	3	1
Variances	4	1
Cumulative Variance	5	1
Features in the Principal Component Space	6	1
	7	1
	8	1
	9	1

### Variances

Shows each principal component's contribution to the overall data variance (for the Variance data provided above). Variance provides a measure of how much of all the variance in the original data set is captured by a given component. The visualization output is helpful to quickly see how many of the principal components explain most of the data set's variance. If the first few principal components have higher Variance values and then the values drop off, those are the components that should be used as the reduced dimension data set.

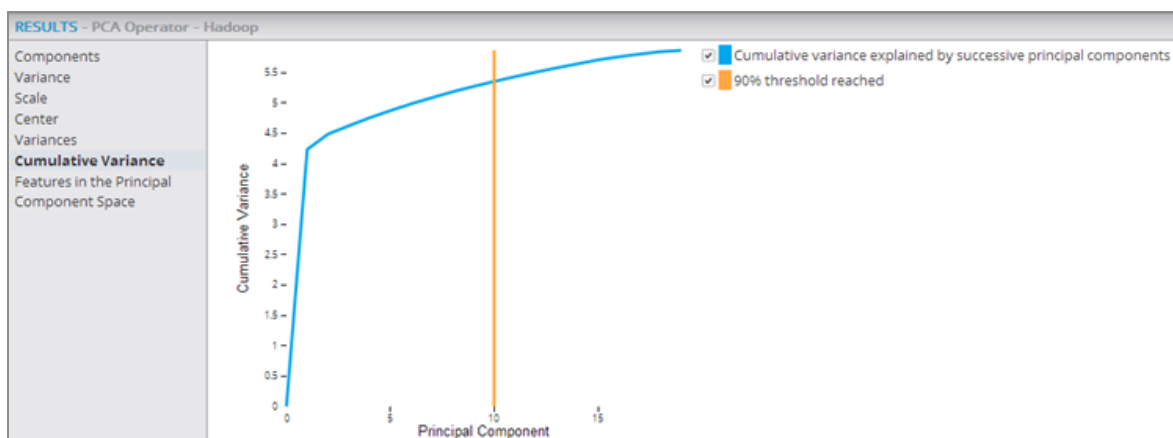
**i Note:** This tab is not output when **Use Spark** is set to **Yes**.



### Cumulative Variance

This graph provides a visualization of the cumulative importance of each principal component (starting with the most significant principal component) and of how effectively the components explain the data set. The 90% Threshold Reached line shows the cumulative principal component point at which 90% of the data variance is explained. This helps know how many of the principal components explain the bulk of the data variance. In the example below, it is the first 10 principal components that explain 90% of the data variance.

**i Note:** This tab is not output when **Use Spark** is set to **Yes**. The Cumulative Variance is reported as an extra column in the **Variance** output tab.

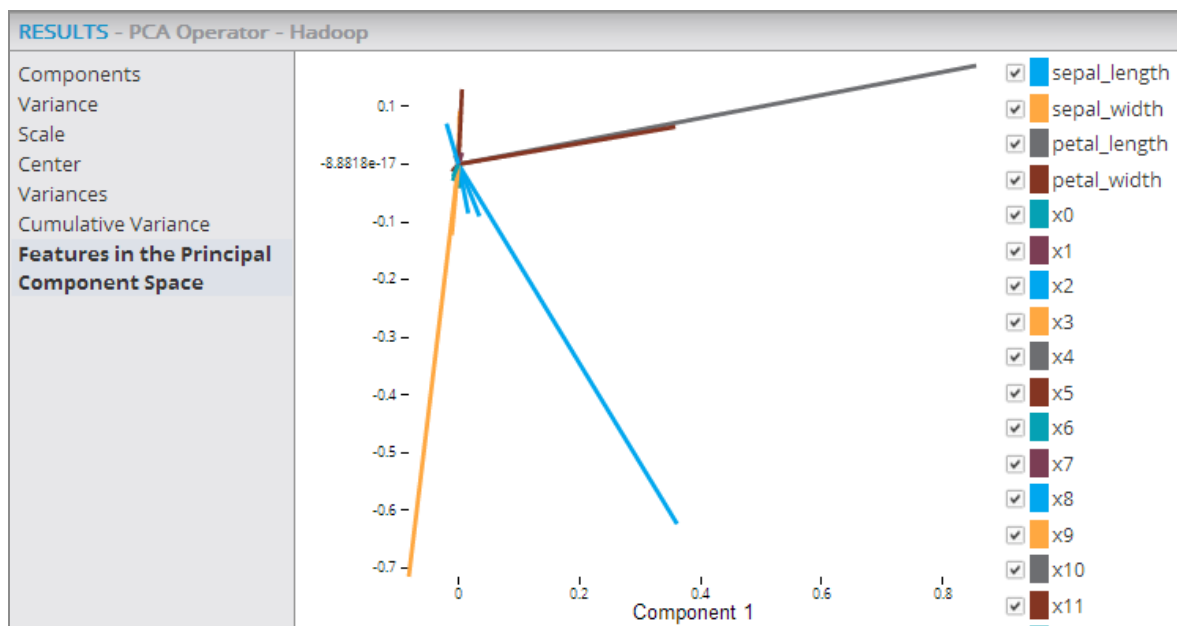




## Features in the Principal Component Space

This graph provides a visualization of the source variables' (columns) axis from the original data space into the two-dimensional space spanned by the first and second principal components. As a result, those variables (columns) whose axes are near each other in the Principal component space have higher correlation to each other and the longer vectors (axes) have higher relevance in explaining the overall variance. In the example below, the three axes for *sepal\_width*, *sepal\_length*, and *petal\_length* are the longest and most relevant to the datavariance. Also, the *sepal\_width* and *sepal\_length* variables are more closely correlated to each other than the *sepal\_length* and *petal\_length* are.

**Note:** This tab is not output when **Use Spark** is set to **Yes**.



**Note:** To learn more about the visualization available in this operator, go to [Explore Visual Results](#).

## Data Output

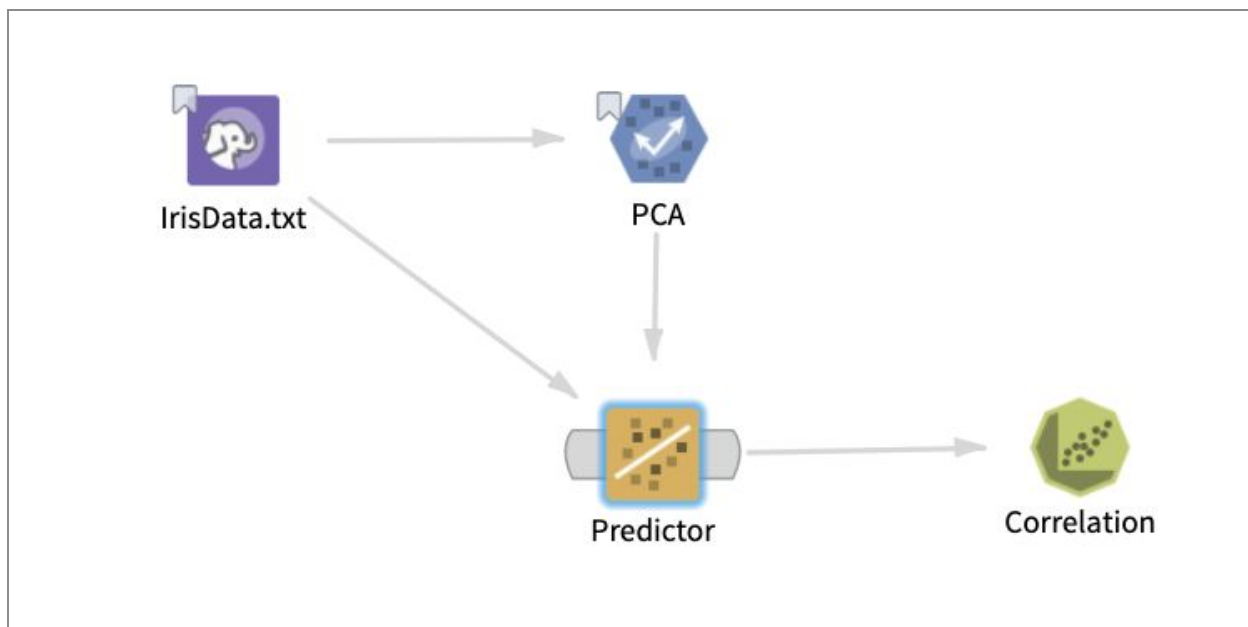
The PCA Operator for Hadoop outputs the principal components, not the transformed data set itself.

**i Note:** To perform the transformation against a data set, the Hadoop PCA operator must be succeeded by a Predictor operator. This operator adds the transformed columns  $y_0\_PCA, \dots, y_p\_PCA$ , where  $(p+1)$  is the minimum between the number of selected columns and the value of the input parameter Maximum Number of Components to output. The transformation can then be processed against the source training data set or a new input data set (with the same variables).

The following example shows the PCA and Predictor operators within a Hadoop workflow, with their output being passed into the Correlation operator, which confirms that the added transformed variables are uncorrelated.

See the [Predictor \(HD\)](#) operator for more details.

## Example



## SVM Classification

Classifies data (both linear and nonlinear) by clustering it into the most distant and distinct groups possible.



## Information at a Glance

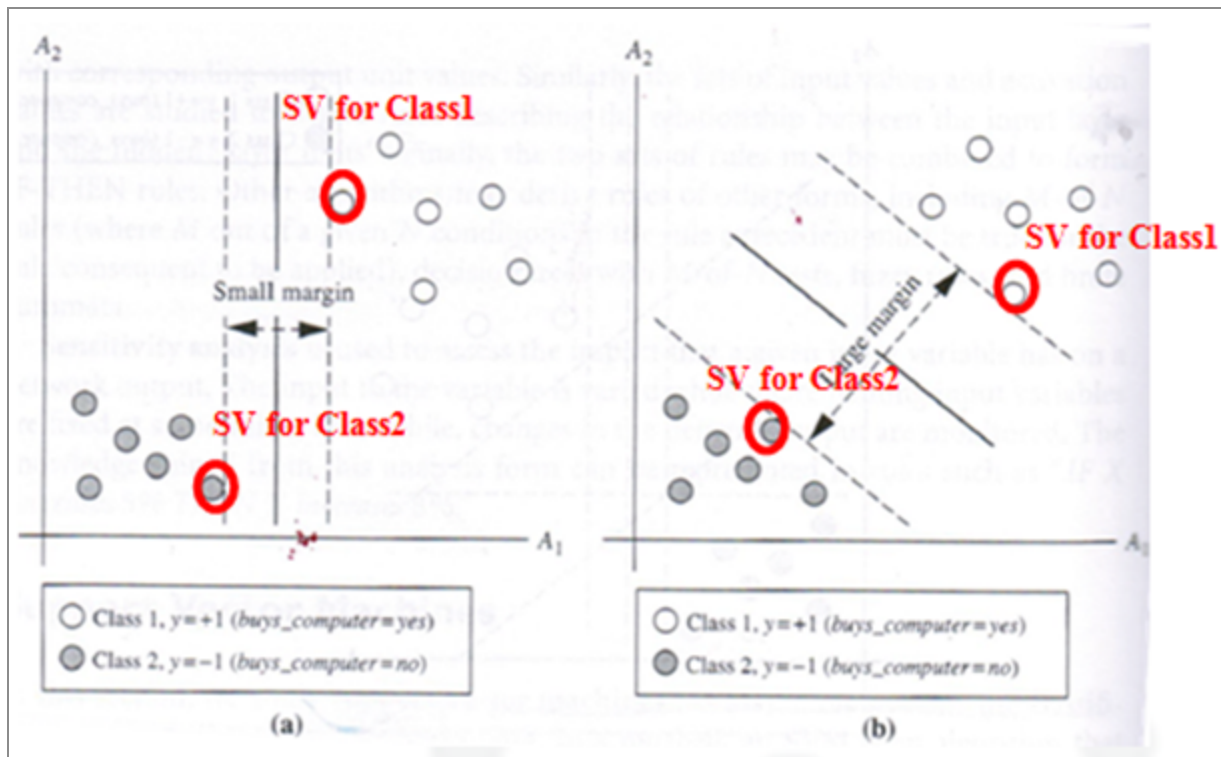
Parameter	Description
Category	Model
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce

## Algorithm

The basic premise behind SVM Classification is following the idea of finding the maximum-margin hyperplane (MMH) in order to minimize the chances of misclassification for the training points. The SVM algorithm runs through the possible ways to divide a data set into classifications using a "hyperplane"; it then chooses the classification with the largest margin, or separation, between the classes' boundary points, called support vectors.

- Support vectors are the closest data points to the optimal decision boundary between classes.
- The margin is defined as distance from the hyperplane to the support vectors.

The following shows a simplified example of this maximal margin division process, with the defining support vector data points circled.



Figures (a) & (b): Support Vector Machine Algorithm Overview Diagram

In Figure (a), the data is separated by a vertical "hyperplane," resulting in a small margin between the two classes.

In Figure (b), the data is separated by a diagonal "hyperplane," resulting in a larger margin between the two classes.

The SVM Classification algorithm would choose the classification of Figure (b) over the classification of Figure (a) due to the larger margin.

The algorithm mathematically calculates the geometric distances of the data points to the boundaries (iteratively), which becomes a problem of solving well-understood mathematical quadratic equations.

The goal of the common SVM formulation is to find the optimal linear decision boundary in term of available training points. The above example shows a data set that is considered to be linearly separable. However, the following diagram shows a data set where it is not possible to draw a straight line to separate the classes. This is referred to as the data set having a "nonlinear decision boundary."

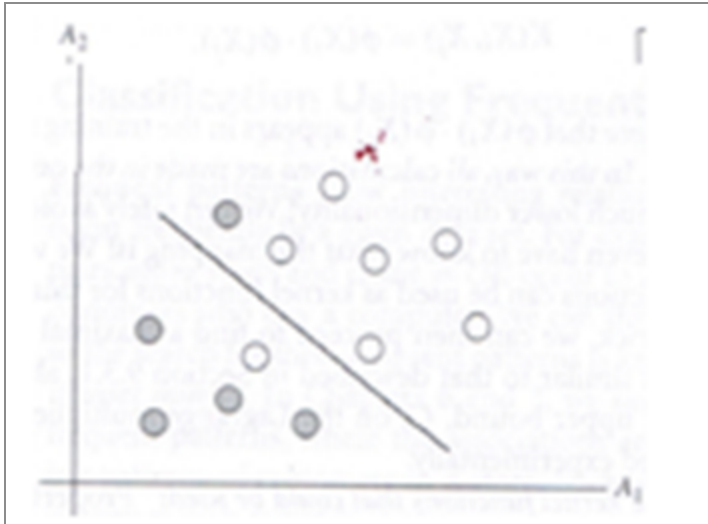


Figure ©): Example of non-linearly separable data, meaning it is not possible to draw a straight line to separate the classes.

In addition to acting as a linear model, the SVM algorithm is able to handle noisy and non-linear separable situations using the concepts of Soft Margin and Kernel Trick.

- The Soft Margin idea manipulates the original SVM optimization problem (so-called primal hard margin) to behave more flexibly for adapting to tolerable non-linearity (noise).
- Kernel Tricks are easy and efficient mathematical transformations of the data to higher dimensional space.
- The Kernel Trick is an easy and efficient mathematical way of mapping data sets into a higher dimensional space; where it finds a "hyperplane" with the hope of make the linear separable representation of the data. This is simply the method of mapping observations from a general set,  $S$ , into an inner product space,  $V$  (equipped with its natural norm), without ever having to compute the mapping explicitly.
- The three Kernel Methods currently supported by the TIBCO Data Science - Team Studio SVM Classification operator are the Dot Product/Linear, the Polynomial Kernel, and the Gaussian Kernel.

### Normalization Type

Within the SVM algorithm, the Normalization type sets the mathematical approach to normalizing the SVM Classification data. The following formulas are used for normalization options:

Option	Description	Formula
Standardization	Standardization normalization approach is the default approach	FeatureValue
Unit Vector	Unit Vector normalization approach is helpful when the modeler needs the data to be in a smaller scale (that is, have smaller values).	FeatureValueSampleNorm
Mixed	Performs both Standardization and Unit Vector normalization on the data.	-
None	No normalization.	-

The SVM Classification algorithm is currently only supported for Hadoop data sources, as follows:

- For the TIBCO Data Science - Team Studio Hadoop (MapReduce) version, the TIBCO Data Science - Team Studio SVM Classification operator implements the primal form of the L2- Regularized, multi-class, cost sensitive Hinge loss function Introduced by Crammer and Singer (2001) [3] along with the idea of explicit random feature mappings for approximate the kernel trick process. The random feature mappings are established based on some classical theorems of harmonic analysis (Bochner 1994 and Schoenberg 1942).
- The supported Kernel functions for Hadoop sources are the Linear/Dot Product, Polynomial, and Gaussian functions.
- For the Gaussian function, as a shift invariant kernel, TIBCO Data Science - Team Studio implements the Random Fourier transformation presented by Rahimi and Recht (2007).
- For the Polynomial function, TIBCO Data Science - Team Studio implements the Random Maclaurin Feature Maps presented by Kar and Karnick (2012).

## Input

A Hadoop data set that contains the dependent and independent variables for modeling.

## Configuration

The following configuration parameters are the minimum settings required to use SVM Classification.

- **Dependent Column**
- **Maximum Number of Iterations**
- **Columns**
- **Kernel Type**

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	<p>A dependent column must be specified for the SVM Classification. Select the data column to be considered the dependent variable for the classification. This is the quantity to model or predict. The dependent column can be either categorical or continuous.</p> <p>The cardinality of the dependent column should not be greater than 1000. Therefore, there can be at most 1000 classes in the model.</p>
<b>Maximum Number of Iterations</b>	<p>Controls the number of times the SVM Classification algorithm is run against the data.</p> <ul style="list-style-type: none"> <li>• When choosing the dot product (linear) kernel, having the higher number of iterations generally brings higher training accuracy. For other types of kernels, the smaller number of iterations generally generates good results both in terms of accuracy and computational complexity.</li> <li>• When using a Kernel Type other than Linear (Gaussian or Polynomial), the modeler should not typically use a Number of Iterations over 40 because they converge quickly and more iterations do not typically make the model any more accurate. When using the Linear Type it is okay to set Maximum Number of Iterations higher, such as 100.</li> </ul> <p>Default value: <b>100</b>.</p>

Parameter	Description
<b>Columns</b>	<p>Sets the Independent Variable data columns to include for the decision tree training. You must specify at least one column.</p> <ul style="list-style-type: none"> <li>• Either sparse columns (output from the Collapse operator) or non-sparse columns are supported, but not a mix of the two.</li> <li>• The limitations on the data dimensionality are having 10 million values for a column of datatype = "sparse" and a total 4000 independent data columns.</li> </ul> <p>Click <b>Select Columns</b> to open the dialog for selecting the available columns from the input data set for analysis. For more information, see <a href="#">Select Columns dialog</a>.</p>
<b>Kernel Type</b>	<p>Specifies which implementation of a data transformation algorithm, or "kernel trick", to use to represent data in a higher dimensional space. Methods (KMs) are a class of data transformation algorithms (equations) for pattern analysis, where the goal is to find and study general types of relations (for example, clusters, rankings, principal components, correlations, and classifications) in general types of data (such as sequences, text documents, sets of points, vectors, images, etc.).</p> <p>According to Mercer's Theorem (1909), the inner product of two data vectors in higher-dimensional space can be represented in linear space by a mathematical formula, or kernel representation. Specifically, it states that every semi-positive definite symmetric function is a kernel. A kernel is a primal space representation of an inner product of two data vectors in higher dimensional space: <math>f(x_i) * f(x_j)</math></p> <p><b>Note:</b> If the independent variable is a sparse column format, the only choice for <b>Kernel Type</b> is <b>Linear</b>.</p> <p>Kernel type options are as follows.</p> <ul style="list-style-type: none"> <li>• <b>Gaussian</b> - Implements the Gaussian formula of <math>\exp(-1.0 * \gamma(x-y)(x-y))</math>.</li> </ul>



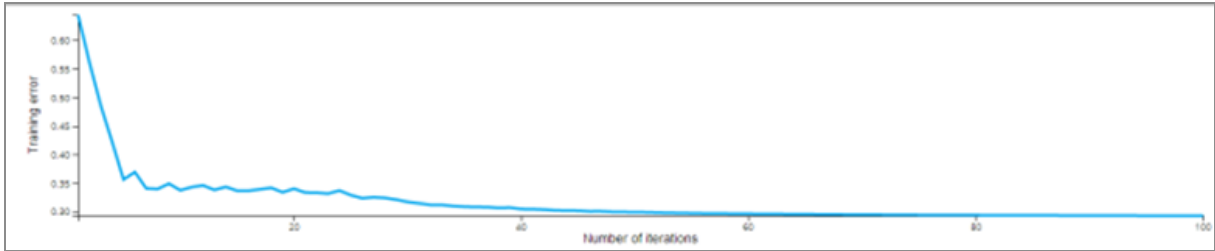
Parameter	Description
	$K(x_i, x_j) = e^{-\frac{\ x_i - x_j\ ^2}{2\sigma^2}}$ <ul style="list-style-type: none"> <li>• <b>Linear</b> (the default) - For the Hadoop MapReduce data sources, this is the explicit default kernel method used even for linearly separable data in order to save on computational cost of solving the optimization problem. This implements the standard dot product of two data vectors. This option is chosen when the modeler believes the data source to be linearly separable (or to have "smooth" decision boundaries).</li> </ul> $K(x_i, x_j) = x_i \cdot x_j$ <ul style="list-style-type: none"> <li>• <b>Polynomial</b> - Represents the dot product of two vectors raised to some exponential degree, <math>d</math>, <math>(x \cdot y)^d</math>. The Polynomial Kernel Method is popular for natural language processing (NLP) applications.</li> </ul> $K(x_i, x_j) = (ax_i^T x_j + b)^d$ <p>The <b>Polynomial</b> and <b>Gaussian</b> kernel types should be chosen when the modeler believes the data source is not linearly separable, or has a rough Decision Boundary. When this is the case, it is recommended that the modeler try implementing the SVM Classification model with both Polynomial and Gaussian Kernel Types and comparing the results for best fit.</p>
<b>Polynomial Degree</b>	Specifies the value of the exponent, $d$ , for the polynomial kernel method implementations.

Parameter	Description
	<div data-bbox="469 289 1031 455" data-label="Equation-Block"> <math display="block">K(x_i, x_j) = (ax_i^T x_j + b)^d</math> </div> <p>For Natural Language Processing (NLP) problems, the most common degree is 2, because using larger degrees tends to lead to over fitting (<a href="http://en.wikipedia.org/wiki/Polynomial_kernel">http://en.wikipedia.org/wiki/Polynomial_kernel</a>). However, for other very sparse, large data sources, such as Forest Cover use case, a good degree is 10 - many support vectors are needed to represent the non-smooth Decision Boundaries.</p> <p>Default value: <b>2</b>.</p> <p>This parameter is enabled only when the <b>Polynomial</b> option of the <b>Kernel Type</b> parameter is selected.</p>
<b>Gamma (γ)</b>	<p>Specifies the value of the γ variable in the following Gaussian Kernel Type radial basis function:</p> <div data-bbox="1044 1031 1182 1094" data-label="Equation-Block"> <math display="block">\gamma = \frac{1}{2\sigma^2}</math> </div> <p>Gamma and sometimes parameterized using <math>\gamma = \frac{1}{2\sigma^2}</math>.</p> <p>Changing the Gamma value changes the accuracy of the Gaussian model. The modeler should use trial and error with cross validation in order to assess what value is best for the given data set.</p> <p>This variable is typically set relative to 1 over the number of independent variables, or dimensionality, of the model. For example, if there were 50 variables, it might be set to <math>1/50 = 0.02</math>.</p> <p>Default value: <b>1</b>.</p> <p>This parameter is enabled only when the <b>Gaussian</b> option of the <b>Kernel Type</b> parameter is selected.</p>
<b>Lambda (λ)</b>	<p>Represents an optimization parameter for SVM Classification. It controls the tradeoff between model bias (significance of loss function) and the regularization portion of the minimization function (variance).</p> <p>The higher the Lambda, the lower the chance of over-fitting. Over-fitting</p>

Parameter	Description
	<p>is the situation where the model does a good job "learning" or converges to a low error for the training data but does not do as well for new, non-training data. If a model does well on the training data but does not perform well on the predictions, the modeler might seek to increase the Lambda value.</p> <p>Default value: <b>0.0001</b>.</p>
<b>Normalization Type</b>	<p>Sets the mathematical approach to normalizing the SVM Classification data.</p> <p>The options are as follows.</p> <ul style="list-style-type: none"> <li>• <b>Mixed:</b> Perform both Standardization and Unit Vector normalization on the data.</li> <li>• <b>None:</b> Do not perform any normalization.</li> <li>• <b>Standardization.</b> The default.</li> <li>• <b>Unit Vector</b></li> </ul> <p>The Standardization normalization implements the normalization equation <math display="block">\frac{\text{FeatureValue} - \text{Mean}}{\text{StdDev}}</math>.</p> <p>The Unit Vector normalization approach is helpful when the modeler needs the data to be in a smaller scale (that is, have smaller values). It implements the equation <math display="block">\frac{\text{FeatureValue}}{\text{SampleNorm}}</math>.</p>
<b>Max JVM Heap Size (MB) (-1=Automatic)</b>	<p>A Java Virtual Machine data storage setting for Hadoop.</p> <ul style="list-style-type: none"> <li>• The default value is <b>1024</b>.</li> <li>• If the value is -1, the system automatically sets the Heap Size.</li> </ul>

## Output

### Visual Output



**Important:** To learn more about the visualization available in this operator, see [Explore Visual Results](#).

The ideal results curve would show the Training Error gradually reducing until it plateaus and flattens out at the minimal attained error rate. This shows the number of iterations at which the model has stopped getting more accurate overall.

In the above example, the Training Error seems to minimize after 60 Iterations. The modeler could rerun the model reducing Number of Iterations to 60 since going beyond that doesn't seem to improve the accuracy of the model.

**Note:** If the results curve continues to oscillate, it is an indication of too much noise in the model. The modeler might try adjusting the Lambda configuration parameter or choosing a different Kernel Type option.

SVM Classification models need additional operators added to them in order to analyze their effectiveness. Often, they are followed by an SVM Predictor which provides the prediction value for each data row compared against the actual data set training value and the associated confidence level. It is also common to add a ROC operator to quickly assess the quality of the model. See the Example section for an illustration.

The output from the SVM Predictor operator shows the predicted Dependent Value for each data instance, along with its associated confidence values.

- The **P(Diagnosis)** column uses a threshold assumption of > than 50% confidence that the prediction is happen. For example, if the confidence  $C(M)$  is over .50, the prediction value displayed is "Malignant," indicating a cancerous cell.
- The **C(M)** column indicates the confidence that the dependent value is M for malignant cell. Note: usually confidence values are a decimal value that represents the % of confidence in its result.

- The **C(B)** column indicates the confidence that the Dependent value is B for Benign cell.

SVM Predictor

Help

Edit Operator

Copy

Paste

Rename

Delete

Step Run

Explore

RESULTS - SVM Predictor

ture	worst.perimete	worst.area	worst.smoothn	worst.compact	worst.concav	worst.concave	worst.symmetr	worst.fractaldit	P(Diagnosis)	C(M)	C(B)
184.6000	2019.0000	0.1622	0.6656	0.7119	0.2654	0.4601	0.1189	0.1189	M	1.0000	0.0000
158.8000	1956.0000	0.1238	0.1866	0.2416	0.1860	0.2750	0.0890	0.0890	M	1.0000	0.0000
152.5000	1709.0000	0.1444	0.4245	0.4504	0.2430	0.3613	0.0876	0.0876	M	1.0000	0.0000
98.8700	567.7000	0.2098	0.8663	0.6869	0.2575	0.6638	0.1730	0.1730	M	1.0000	0.0000
152.2000	1575.0000	0.1374	0.2050	0.4000	0.1625	0.2364	0.0768	0.0768	M	1.0000	0.0000
103.4000	741.6000	0.1791	0.5249	0.5355	0.1741	0.3985	0.1244	0.1244	M	1.0000	0.0000
153.2000	1606.0000	0.1442	0.2576	0.3784	0.1932	0.3063	0.0837	0.0837	M	1.0000	0.0000
110.6000	897.0000	0.1654	0.3682	0.2678	0.1556	0.3196	0.1151	0.1151	M	1.0000	0.0000
106.2000	739.3000	0.1703	0.5401	0.5390	0.2060	0.4378	0.1072	0.1072	M	1.0000	0.0000

Predicted  
Dependent Values

Confidence  
Values

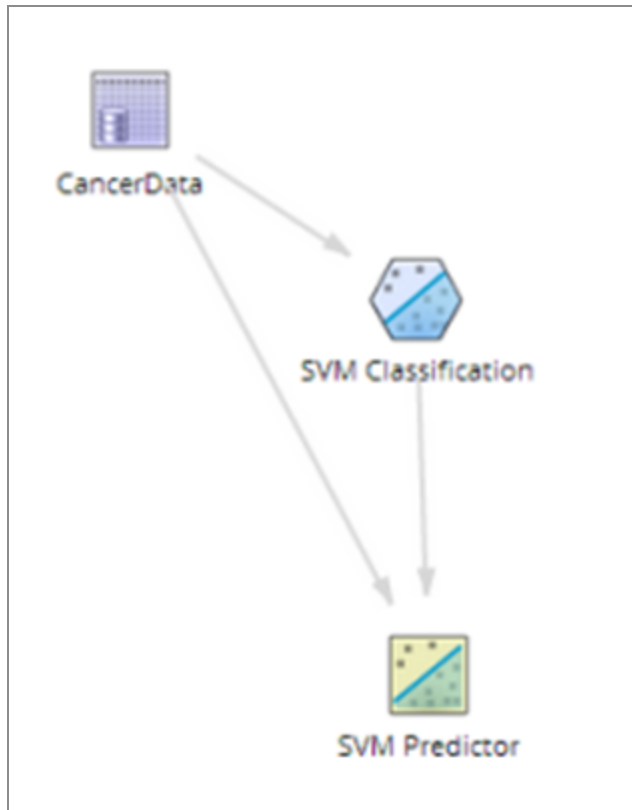
Adding additional Scoring operators, such as a ROC graph, are also helpful in immediately assessing how predictive the Random Forest model is. For the ROC graph, any AUC value over .80 is typically considered a "good" model. A value of 0.5 just means the model is no better than a "dumb" model that can guess the right answer half the time.

**i Note:** When analyzing the results of a SVM Classification Model, the most important assessment is that it scores well outside of the training data (indicated by the ROC, LIFT and Goodness of Fit Scoring Operators).

## Data Output

None.

## Example



## NLP Operators

You can use the NLP (Natural Language Processing) operators to process and extract text from human-language documents.

### N-gram Dictionary Builder

A sequence of tokens (one or greater) that might appear in a text corpus. The N-gram Dictionary operator parses each document in the corpus into tokens, and then into all possible n-grams (combinations of sequential tokens).



## Information at a Glance

Parameter	Description
Category	NLP
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

For more detailed information about working with N-gram Dictionary Builder, see [Test Corpus Parsing](#).

## Input

A corpus of documents represented as a data set with one row per document and at least one column of text. You can select the column with the text to analyze from the **Text Column** parameter.

## Configuration

**Important:** The [Text Featurizer](#) operator uses the same parsing rules as the dictionary built in this operator. Therefore, if you are connecting this dictionary builder to the Text Featurizer, be aware that the tokenization settings in this operator (**Case Insensitive**, **Use Sentence Tokenization**, **Use Stemmer**, and **Filter Stop Words**) are used.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Text Column</b>	Select the column with the text of each document.
<b>Maximum Size of N-Gram</b>	Must be <b>1</b> , <b>2</b> (the default), or <b>3</b> (Unigrams, Bigrams, or Trigrams).
<b>Case Insensitive</b>	If <b>yes</b> (the default), all words in corpus are considered lowercase, so "Fish" and "fish" would be considered the same token.
<b>Use Sentence Tokenization</b>	<ul style="list-style-type: none"> <li>If <b>yes</b>, use the Sentence Tokenizer from apache.open.nlp to determine where the ends of sentences are. N-grams are then only calculated from within the sentence units.</li> <li>If <b>no</b> (the default), an n-gram can be formed across sentences. For example, the text "This is sentence one. This is sentence two." could get "one This" as a bigram.</li> </ul>
<b>Use Stemming</b>	<p>Stemming means reducing the forms of a word to a single word. For example: "swims", "swimmer", and "swimming" would all be parsed into the token "swim". Stemming is its own complicated subfield of linguistics, so for simplicity, we use the Porter Stemmer from apache.open.nlp.</p> <ul style="list-style-type: none"> <li>If <b>yes</b>, parse each token into its root word before computing n-grams.</li> <li>Default value: <b>no</b>.</li> </ul> <div> <p><b>Note:</b> The stemmer might not produce real English words. For example, the stemmed version of "parsed" is "pars", because stemming removes the "ed" suffix.</p> </div>
<b>Filter Stop Words</b>	<ul style="list-style-type: none"> <li>If <b>yes</b>, stop words are removed from the data set.</li> <li>If <b>no</b> (the default), stop words are left in the data set.</li> </ul> <p>Stop words are words that are very common or not useful for the analysis;</p>



Parameter	Description
	for example, "a", "the", and "that". You can specify your own list of stop words in the <b>Stop Words File</b> parameter, or use our default list of stop words <a href="#">Stop Words</a> .
<b>Stop Words File</b>	<p>If left at the default value, a standard set of stop words is used. You can find this list <a href="#">Stop Words</a>.</p> <p>Otherwise, choose a file that contains a list of stop words. This list must be one word per line and should be small enough to fit in memory.</p> <ul style="list-style-type: none"> <li>Default value: @default_tmpdir/alpine_out/@user_name/@flow_name</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li><b>Yes</b> - if the path exists, delete that file and save the results.</li> <li><b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li><b>Yes</b> specifies using the default Spark optimization settings.</li> <li><b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Outputs

### Visual Output

Three sections are displayed after the operator is run:

#### Dictionary

A table that shows the first preview of the n-gram dictionary generated by the operator and passed on to future operators.

## Corpus Statistics

Shows aggregate counts for number of documents, n-grams, and unique tokens found, as shown in the following example.

<a href="#">Dictionary (Output)</a> <a href="#">Corpus Statistics</a> <a href="#">Summary</a>	<b>Document Statistics</b>	
	Total Number of Documents	44
	Total number of N-Grams	3308
	Total number of Unique N-Grams	1736
	<b>Total N-Gram Statistics</b>	
	Total Number of Unigrams	1676
	Total Number of Bigrams	1632
	<b>Total Unique N-Gram Statistics</b>	
	Total Number of Unique Unigrams	537
	Total Number of Unique Bigrams	1199

## Summary

Contains some information about which parameters were selected and where the results were stored. Use this information to navigate to the full results data set.

<a href="#">Dictionary (Output)</a> <a href="#">Corpus Statistics</a> <a href="#">Summary</a>	Operator Name	N-Gram Dictionary Builder
	Operator Type	N-Gram Dictionary Builder
	<b>Parameters</b>	
	Selected Text Column Name	Column1
	Convert All Text to Lowercase	Yes
	Stemming	No Stemmer
	Filter Stop Words	no
	The N-Gram Dictionary is stored at:	
	/user/alpine_out/allison/DocTests/N_Gram_Dictionary_Builder_1463500338847	

## Additional Notes

The data output of this operator is written to HDFS as a delimited file, but is not recognized by TIBCO Data Science - Team Studio as a tabular data set. This is because it is a special n-gram dictionary type that is only recognized by the [Text Featurizer](#) operator.

Although you cannot connect a transformation operator such as [Summary Statistics \(DB\)](#) to this operator directly, you can go to the location of the results on HDFS (specified in the **Summary** tab of the results pane), drag the file(s) onto your workflow, and use that as input to other TIBCO Data Science - Team Studio operators. However, keep in mind that, if you use this method, the files might be stored in multiple parts.

## N-gram Dictionary Loader

Creates an N-gram dictionary object from a dictionary data set input (with the exact same columns as the output dictionary data set created by the N-gram Dictionary Builder operator), and the location of the N-gram dictionary builder configuration file (which is always stored in HDFS when training an N-gram Dictionary Builder operator and has the output suffix `_dictInfo`).



### Information at a Glance

Parameter	Description
Category	NLP
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

For more information, see [N-gram Dictionary Builder](#).

With this operator, you can reuse an N-gram dictionary without having to retrain the N-Gram Dictionary Builder operator each time. You can filter a dictionary created by an N-Gram Dictionary Builder operator in a custom way, and then use it as the new dictionary data set to create an N-gram dictionary object that can be used with Text Featurizer or LDA Trainer operators.

### Input

A tabular data set that represents an N-Gram dictionary (most commonly the output dictionary of an N-Gram Dictionary Builder operator, which has been filtered out), with the exact same column names and types as the N-Gram Dictionary Builder data set output.

<b>ngram</b> chararray ▼	<b>size_of_ngram</b> int ▼	<b>total_count_in_corpus</b> long ▼	<b>number_of_documents</b> long ▼
-----------------------------	-------------------------------	--	--------------------------------------

## Restrictions

This operator requires an input with the exact same column names and types as the N-Gram Dictionary Builder data set output; otherwise, an error occurs.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>N-Gram Dictionary Builder Configuration</b>	<p>Select the HDFS directory where the configuration parameters and corpus statistics of a trained N-Gram Dictionary Builder operator is stored.</p> <div> <p><b>Note:</b> It should have been created when running an N-Gram Dictionary Builder operator in the first place, and stored at the same output path of the N-Gram dictionary data set, with the <code>_dictInfo</code> suffix appended.</p> <p>This configuration file contains information on the training corpus of documents as well as user-specified options when training the N-Gram Dictionary Builder in the first place (stemming, case sensitivity, stop words, sentence tokenization, and so on).</p> </div>

## Output

### Visual Output

Visual output includes **Dictionary**, **Corpus Statistics**, and **Summary** sections.

### Dictionary

A table that shows the first preview of the n-gram dictionary loaded by the operator and passed on to future operators.

<b>Dictionary (Output)</b> <a href="#">Corpus Statistics</a> <a href="#">Summary</a>	ngram	size_of_ngram	total_count_in_corpus	number_of_documents
	someone	1	8	8
	google	1	5	5
	book	1	6	6
	process	1	4	4
	find	1	8	8
	email	1	5	5
	people	1	8	8
	india	1	11	11
	one	1	8	8

### Corpus Statistics

Shows aggregate counts for number of documents, n-grams, and unique tokens found.

Results - N-Gram Dictionary Loader-2	
<b>Dictionary (Output)</b> <b>Corpus Statistics</b> <a href="#">Summary</a>	<b>Document Statistics</b>
	Total Number of Documents 300
	Total number of N-Grams 1606
	Total number of Unique N-Grams 914
	<b>Total N-Gram Statistics</b>
	Total Number of Unigrams 1606
	<b>Total Unique N-Gram Statistics</b>
	Total Number of Unique Unigrams 914

## Summary

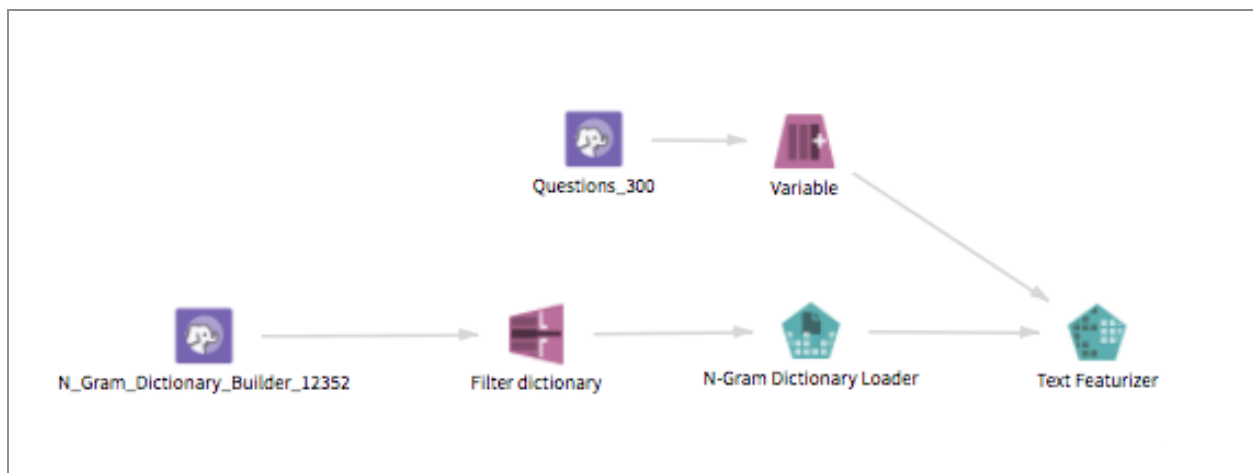
Contains some information about which parameters were selected and where the results were stored. Use this information to navigate to the full results data set.

Dictionary (Output) Corpus Statistics Summary	Operator Name	N-Gram Dictionary Loader-2
	Operator Type	N-Gram Dictionary Builder
	Parameters	
	Convert All Text to Lowercase	Yes
	Stemming	No Stemmer
	Filter Stop Words	from the file/tmp/English_Stop_Words_And_Two_Letters_Or_Less.csv
	Note: Stop words are cleaned according to the same parameters (lower case and stemming) as dictionary tokens.	

## Data Output

The N-gram dictionary object that can be connected to a Text Featurizer or LDA Trainer operator (in combination to a data set input).

## Example



## Text Extractor

Using the Text Extractor, users can select an HDFS input directory that contains a set of documents, and then parse that content to create a new data set that contains the parsed text.



## Information at a Glance

Parameter	Description
Category	NLP
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

## Input

No input needs to be directly connected. Select an input directory from the parameter configuration dialog.

## Bad or Missing Values

If the operator encounters an error while reading or parsing a document, it flags as true in the **read\_or\_parse\_error** column, and the text of the error is displayed in the **text\_content** column. Such an error can occur if the user does not have read permissions on the selected directory (or specific files), or if a file is corrupted.

## Restrictions

Text Extractor accepts only the following file types.

- .doc
- .docx
- .html

- .log
- .pdf
- .ppt
- .pptx
- .rtf
- .txt
- .xml

Text Extractor does not preserve the structure of the document; it only parses the text data. Thus, the structure of the original document might be lost.

If the fonts in your document use a non-standard encoding and the document structure does not contain a /ToUnicode table associated with these fonts, the text content extracted might be garbled. Many different encodings and fonts exist, and it is not possible to predict all of them. Some files are produced without this important metadata. Even though you can display and print the file properly, the file does not contain information about the meaning of the font/letter shapes. In this case, you must recreate the file or use OCR. ([source](#))

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Data Source (HD)</b>	The Hadoop data source.
<b>Input Directory</b>	The input directory that contains the files to parse (wildcards and patterns are supported, as well as single file selection).  <b>Tip:</b> The input directory path can be entered manually, and the user can enter a regular expression as path pattern (for example, /dir/user*/projectA*)



Parameter	Description
	<p>The operator parses only files with selected extensions in the chosen directory its tree of subdirectories (other files are skipped).</p> <p>If no files with the selected extensions are found, the output is empty and the following error message is displayed in the addendum:</p> <pre>"No files with selected extension were found in the input directory and subdirectories"</pre> <p><b>Caution:</b> Filenames with the following characters: {},  are not supported and cause the job to fail.</p>
<b>File Formats to Parse</b>	<p>Extensions of the files to parse from the available options.</p> <p><b>Note:</b> The filenames must explicitly include the extension. For example, a PDF file titled mydoc is not read, but a PDF file titled mydoc.pdf is read.</p>
<b>Maximum Number of Characters per File</b>	<p>If a file has more characters than this limit, the file is not parsed. The default limit is 10,000,000 characters. The column read_parse_error is set to true and an error is displayed in the output column text_content.</p> <p><b>Caution:</b> This limit is set to avoid the Spark job hanging because the directory contains huge files that a user could try to parse by mistake. To parse these large files, increase this limit. Doing so may require tuning the Spark memory settings.</p>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

Operation created output with 6 columns.

file_index	file_path	file_extension	text_content	read_parse_error	empty_file
2	/Emilie/file2.txt	txt	7w01Standard01Aw01741w0100934 11w01Silver01Aw01733w0101075 15w01Standard01Aw01697w0101330 19w01Standard01Aw01701w0101463 23w01Silver01Aw01677w0101588 32w01Standard01Aw01696w0102125 40w01Silver01Aw01740w0102476 44w01Standard01Aw01654w0102738 48w01Silver01Aw01663w0102837 65w01Silver01Aw01677w0103841 69w01Silver01Aw01731w0104043 73w01Standard01Aw01741w0104103 98w01Standard01Aw01648w0104953 121w01Silver01Aw01671w0106420 125w01Standard01Aw01662w0106518 146w01Standard01Aw01714w0107607 158w01Stand	false	false
3	/Emilie/header.txt	txt	N/A	false	true
4	/Emilie/multi_record.txt	txt	D1 35.00 D2 40.00 D3 45.50 T3 MAI_SCHEDULED2 LongType PIM_NEW_M - DEBT_SCHEDULED - COL_AMT MAI_SCHEDULED3	false	false

### Data Output

This operator outputs a tabular data set (.TSV) with the following six columns.

- **doc\_index** - a unique index created to identify the document.
- **file\_path** - the original file path.
- **file\_extension** - the extension of the file.
- **text\_content** - the text content parsed from the document.
- **read\_or\_parse\_error** - a boolean value that determines whether an error occurred while reading/parsing this document.
  - **true** - an error occurred while reading or parsing. If an error occurred, it appears in the **text\_content** column.
  - **false** - no errors occurred while parsing this document.
- **is\_empty** - a boolean value that is set to true if the file to be read is empty (or does not contain any alphanumeric characters).

## Text Featurizer

Parses a corpus of text into numeric features. You can select which metric(s) to compute for each document and for each of the selected n-grams or hashed features.



### Information at a Glance

Parameter	Description
Category	NLP
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

## Input

The operator takes output from the [N-gram Dictionary Builder](#) and a data set of documents (one for each row). Use the operator to select which n-grams to use as features based on a set of criteria.

### Bad or Missing Values

Rows with null values are not removed. Instead, an empty value is just considered a document with no n-grams.

## Restrictions

If you click **Yes** for **Use N-gram Values as Column Names**, you cannot connect the output to other operators.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Text Column</b>	The column that contains the documents to analyze. Each row is treated as one document.
<b>Columns to Keep</b>	The columns to pass through as features. These columns do not change; they are sent through to the output as is.
<b>N-Gram Selection Method</b>	<p>The criteria used to choose the n-grams to pull out as features. These preferences are applied to the training corpus input by the N-gram Dictionary Builder.</p> <ul style="list-style-type: none"> <li>• <b>Appear in the Most Documents</b> (the default)</li> <li>• <b>Appear in the Fewest Documents</b></li> <li>• <b>Feature Hashing</b> - Use all of the n-grams, but reduce the dimensionality of the feature space by storing each n-gram as a hash value using the number of buckets specified by the <b>Max Number of</b></li> </ul>

Parameter	Description
	<p><b>Unique N-grams</b> parameter below. Then the columns represent the summed valued of all the n-grams associated with one hash value.</p> <ul style="list-style-type: none"> <li>• <b>Occur Least Frequently (Entire Corpus).</b></li> <li>• <b>Occur Most Frequently (Entire Corpus).</b></li> </ul>
<b>Max Number of Unique N-grams to Select (Feature Hashing Size)</b>	<p>The number of n-grams to select from the dictionary to use as features. The default value is <b>500</b>.</p> <p>If, for <b>N-Gram Selection Method</b>, you selected <b>Feature Hashing</b>, this parameter represents the size of the hash set.</p> <p>In either case, the total number of features in the new data set is no greater than the number of pass-through columns selected + the value of this parameter multiplied by the number of values selected in the <b>For each N-gram and Document Calculate</b> parameter (1,2,3).</p>
<b>For Each N-gram and Document Calculate</b>	<p>Select any or all of the following metrics:</p> <p><b>Raw N-Gram Count</b> - The number of times the n-gram appears in the document.</p> <p><b>Normalized N-Gram Count</b> - Normalize word count against the original corpus. This is calculated using the following equation:</p> $\frac{\text{Number of times n-gram appears in the document}}{\text{Number of tokens in the document}}$ <p>The <b>Number of tokens in the document</b> metric is reported in the first column output of the featurized data set <code>number_of_tokens</code>. With custom tokenization and stop-word removal, the notion of a token can be complicated. We use the number of unigrams found in the document after stop words, and special characters have been removed.</p> <p><b>tf-idf</b> - Computes tf-idf value for each n-gram.</p>

Parameter	Description
	<p><b>Note:</b> tf-idf (term frequency - inverse document frequency) is a common algorithm used for feature generation in natural language processing. It calculates the relative importance of a term <math>t</math> in a document. Tf-idf lowers the weight of terms that are used more frequently in the corpus. To calculate the tf-idf score for a term, we use the following formula:</p> $TF(t) = \text{Number of times n-gram appears in the document}$ <p>While TF is often calculated as the term frequency per token (using the same calculation as the Normalized N-gram Count, we use the number of times it appears here as an approximation for performance reasons.</p> $IDF(t) = \log_e \left( \frac{\text{Total number of documents in training corpus}^*}{\text{Docs in the training corpus containing the n-gram}^{**}} \right)$ <p>*This metric is the "Total Number of Documents" in the training corpus. The number is reported in the Corpus Statistics section of the N-Gram Dictionary Builder output.</p> <p>**This metric corresponds to the "document count" for the n-gram, which is reported in the dictionary output of the N-Gram Dictionary Builder.</p> <p>tf-idf = <math>TF(t) * IDF(t)</math></p> <p>To learn more about tf-idf metrics, see <a href="http://www.tfidf.com/">http://www.tfidf.com/</a>.</p>
Use N-gram Values as Column Names	<ul style="list-style-type: none"> <li>• <b>Yes</b> - The column names are not of the form ngram1_raw_count. Instead, they are the names of the actual n-gram. Instead of seeing ngram1_raw_count and having to refer to a table to see what ngram1 actually is, you see the column name swim_raw_count for the token "swim".</li> </ul> <p><b>Important:</b> If this option is selected, you can connect the operator to another operator, but the columns with the n-grams do not show up. Instead you see only the "Pass Through Columns" and the four columns with document level statistics: number_of_tokens, normalized_number_tokens, number_unique_tokens, and normalized_number_unique_tokens.</p>

Parameter	Description
	<p><b>Tip:</b> If you want to leverage the full output (with n-gram value columns) for further analysis, then you can drag the output from HDFS onto the canvas and connect it to subsequent operators.</p> <p>The "Addendum" specifies the output location of the results. Navigate to that location and drag the entire directory onto the canvas. See <a href="#">Using the Results of Text Featurizer</a>.</p> <ul style="list-style-type: none"> <li>• <b>No</b> (the default) - The operator is non-terminal, meaning that you can connect it to a subsequent operator and transmit all the output columns, but the column names are the number of the n-gram. This means that the columns in the output have names such as ngram1_raw_count and ngram2_raw_count, but the value of those n-grams are mapped to those numbers in a separate table in the output.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	<p>The location to store the output files.</p>

Parameter	Description
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Outputs

### Visual Output

There are three sections of output:

#### Featurized data set (Output)

The output that is passed on to the next operator. Contains a row for each document and columns for the features generated for the n-grams.

Here is an example (shortened) for analyzing Martin Luther King's speech "I Have A Dream":

Text	number_of_tokens	normalized_number_tokens	number_unique_token	normalized_number_unique_token	ngram1_count	ngram1_count_normalized	ngram1_TF_IDF
And as we walk, we must make the pledge that we shall always march ahead.	15	0.00894988	13	0.02420857	0	0	0
We cannot turn back.	4	0.00238663	4	0.00744879	0	0	0
We cannot walk alone.	4	0.00238663	4	0.00744879	0	0	0
And so even though we face the difficulties of today and tomorrow, I still have a dream. It is a dream deeply rooted in the American dream.	27	0.01610979	22	0.04096834	1	0.03703704	0.31845373
Let us not wallow in the valley of despair, I say to you today, my friends.	16	0.00954654	16	0.02979516	1	0.0625	0.31845373
I am happy to join with you today in what will go down in history as the greatest demonstration for freedom in the history of our nation.	27	0.01610979	23	0.04283054	1	0.03703704	0.31845373

### N-Grams to Column Names

The dictionary of the actual value of the n-gram (for example, "freedom") to the name of the n-gram used in the column names such as ngram5. If feature hashing was used, this is essentially empty.



N-Gram Label	N-Gram Value	N-Gram Size	Total N-Grams In Corpus	Total Number of Documents
ngram1	of	1	99	32
ngram2	the	1	103	30
ngram3	and	1	54	22
ngram4	to	1	59	20
ngram5	a	1	37	19
ngram6	freedom	1	20	19

## Summary

Some information about where the results were written and which parameters were chosen.

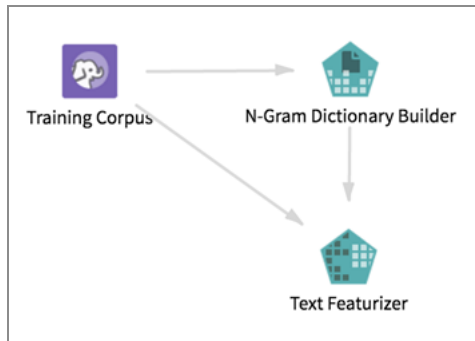
Operator Name	Text Featurizer
Operator Type	Text Featurizer
<b>Parameters:</b>	
Selected Text Column Name	Text
N-Gram Selection Method	Appear in the Most Documents
For Each N-Gram and Document Calculate	Raw N-Gram Count, Normalized N-Gram Count, TF-IDF
Number of N-Grams Selected	500
The Featurized Dataset is stored at:	
/tmp/alpine_out/allison/NLP/Text_Featurizer_1463777942532	
The N-Gram to Column Name dictionary is stored at:	
/tmp/alpine_out/allison/NLP/Text_Featurizer_1463777942532_conversionTable	

## Data Output

The data output is in two parts. One data set is the featurized data set, which is shown in the **Featurized data set (Output)** section above. This is what is sent to subsequent operators.

A dictionary mapping the n-gram numbers to their values is also written out, using the information from the **N-grams to Column Names**.

## Example



## Stop Words

Stop words are words that are very common or not useful for an analysis.

---

a

---

about

---

above

---

after

---

again

---

against

---

all

---

am

---

an

---

and

---

any

---

are

---

---

as

---

at

---

be

---

because

---

been

---

before

---

being

---

below

---

between

---

both

---

but

---

by

---

can

---

did

---

do

---

does

---

doing

---

don

---

don't

---

down

---

---

during

---

each

---

few

---

for

---

from

---

further

---

had

---

has

---

have

---

having

---

he

---

her

---

here

---

hers

---

herself

---

him

---

himself

---

his

---

how

---

I

---

---

if

---

in

---

into

---

is

---

it

---

its

---

itself

---

just

---

me

---

more

---

most

---

my

---

myself

---

no

---

nor

---

not

---

now

---

of

---

off

---

on

---

---

once

---

only

---

or

---

other

---

our

---

ours

---

ourselves

---

out

---

over

---

own

---

s

---

same

---

she

---

should

---

so

---

some

---

such

---

t

---

than

---

that

---

---

the

---

their

---

theirs

---

them

---

themselves

---

then

---

there

---

these

---

they

---

this

---

those

---

through

---

to

---

too

---

under

---

until

---

up

---

very

---

was

---

we

---

---

were

---

what

---

when

---

where

---

which

---

while

---

who

---

whom

---

why

---

will

---

with

---

you

---

your

---

yours

---

yourself

---

yourselves

---

## LDA Predictor

Uses both the model trained by the LDA Trainer and a tabular data set to output topic prediction for the new documents in various formats.





## Information at a Glance

Parameter	Description
Category	NLP
Data source type	HD
Send output to other operators	Yes <sup>1</sup>
Data processing tool	Spark

This operator also stores in HDFS the Featurized data set used as input for LDA Prediction (and created from the featurization parameters specified in the LDA Trainer). You can drag this data set onto the canvas for further analysis.

For more information about using LDA, see [Unsupervised Text Mining](#) and [LDA Training and Model Evaluation Tips](#).

## Input

The LDA Predictor requires the following 2 inputs.

- The output model from an LDA Trainer.
- A tabular data set with at least a unique document ID column and a text content column (for example, output of the Text Extractor operator).

**i Note:** The LDA Predictor operator includes the Text Featurization of **Text Column**, which converts raw text to n-gram features.

---

<sup>1</sup>The full output schema is not available until you step-run the operator. After you run this operator, the output schema automatically updates, and subsequent operators either validate or turn red, depending on the structure of the output data.

## Bad or Missing Values

If a row contains a null value in at least one of the **Doc ID** column or **Text** column, the row is removed from the data set. The number of null values removed can be listed in the **Summary** section of the output (depending on the chosen option for **Write Rows Removed Due to Null Data To File**).

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Document ID Column</b> *required	Column that contains the document unique ID (String, Int, or Long).  <b>Note:</b> This column must contain unique IDs. If duplicate values are found, the job fails with an error message.
<b>Text Column</b> *required	Column that contains the document text content (that is featurized based on N-Gram Dictionary Builder and Featurization input parameters).
<b>Other Columns to Keep</b>	Columns to keep in the output data set.
<b>Write Rows Removed Due to Null Data To File</b> *required	<p>Rows with null values (in <b>Doc ID Column</b> or <b>Text Column</b>) are removed from the analysis. This parameter allows you to specify that the data with null values be written to a file.</p> <p>The file is written to the same directory as the rest of the output. The filename is bad_data.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI.</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write</li> </ul>

Parameter	Description
	all removed rows to an external file.
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options are the following.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options are the following.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Outputs

### Visual Output

- A tabular data set output with the following columns (which can be further

filtered, depending on the columns needed for the user's specific use case):

- **Doc ID Column** and **Pass Through** columns selected
- **top\_topics\_summary** column with the top topics and their corresponding weights in a dictionary format (the number of top topics displayed corresponds to the **Max Topics to Describe Documents** parameter set in the LDA Trainer).
- Full topic distribution weights (**weight\_topic\_X** columns)
- Top topic distribution as pairs of columns **topic\_ranked\_X**, **weight\_topic\_ranked\_X**). The number of top topics displayed corresponds to the **Max Topics to Describe Documents** parameter set in the LDA Trainer)

file_index	file_path	top_topics_summary	weight_topic_0	weight_topic_1	weight_topic_2	weight_topic_3	weight_topic_4	weight_topic_5	topic_ranked_1	weight_topic_ranked_1	topic_ranked_2	weight_topic_ranked_2	topic_ranked_3
831	/txt/Dataset_blogdisc	["0":0.69,"1":0.31,"2":0.0]	0.6926	0.0005	0.3062	0.0003	0.0002	0.0002	0	0.6926	2	0.3062	1
1,831	/txt/Dataset_blogdisc	["0":0.74,"1":0.26,"2":0.0]	0.7407	0.2587	0.0002	0.0002	0.0001	0.0001	0	0.7407	1	0.2587	3
1,231	/txt/Dataset_blogdisc	["0":0.75,"1":0.13,"2":0.12]	0.7451	0.1345	0.1182	0.001	0.0005	0.0007	0	0.7451	1	0.1345	2
631	/txt/Dataset_blogdisc	["0":0.87,"1":0.13,"2":0.0]	0.8659	0.0006	0.0004	0.0005	0.0003	0.1324	0	0.8659	5	0.1324	1
31	/txt/Dataset_blogdisc	["0":0.9,"1":0.09,"2":0.0]	0.9013	0.0017	0.0942	0.0013	0.0007	0.0009	0	0.9013	2	0.0942	1

- **Featurized data set** (stored in HDFS): featurized data set used as input for LDA prediction:

file_index	number_of_toks	normalized	number_unique	normalized_number_unique_toks	http	as	n_t	like	get	us
2,428	3,739	0	1,406	0.0007	1	24	27	14	13	53
251	358	0	215	0.0001	1	0	1	0	2	5
1,180	568	0	367	0.0002	1	4	4	1	4	6
1,489	189	0	159	0.0001	1	6	0	0	0	0
1,801	125	0	79	0	1	0	0	0	0	0

- **Summary** tab that describes the parameters selected, model prediction results, output location of the results, and the number of rows processed

Output	Featurized Dataset	Summary
Operator Type		LDA Predictor
Operator Name		LDA Predictor-1
Input Parameters		
Document ID Column:	file_index	
Text Column:	text_content	
Columns to Keep:	file_index (default), file_path	
Prediction Metrics		
Log Likelihood (entire corpus):	-3.2408075225E8	
Average Log Likelihood (per document):	-36623.43	
Log Perplexity (per token):	5.82	
Output		
Input data size:	8850 rows	
Input size after removing rows due to null values in file_index or text_content:	8849 rows	
Rows removed due to null values in file_index or text_content:	1 rows (0%)	
The prediction results are stored at:		
/tmp/alpine_out/emilie/Test_LDA_11340/LDA_Predictor_1_1473953022924		
The featurized dataset (N-Gram counts per doc) is stored at:		
/tmp/alpine_out/emilie/Test_LDA_11340/LDA_Predictor_1_1473953022924_featurized_data		

**Training data Log Likelihood (entire corpus):** Lower bound on the log likelihood of the entire corpus.

**Training data Average Log Likelihood (per document):** Log Likelihood/Total Number of Documents

**Training data Log Perplexity (per token):** Upper bound on the log perplexity per token of the provided documents, given the inferred topics (lower is better). In information theory, [perplexity](#) is a measurement of how well a probability distribution (or probability model) predicts a sample, and is a common measure of model performance in topic modeling. More specifically, it measures how well the word counts of the documents are represented by the word distributions represented by the topics. A low perplexity indicates the probability distribution is good at predicting the sample.

## Data Output

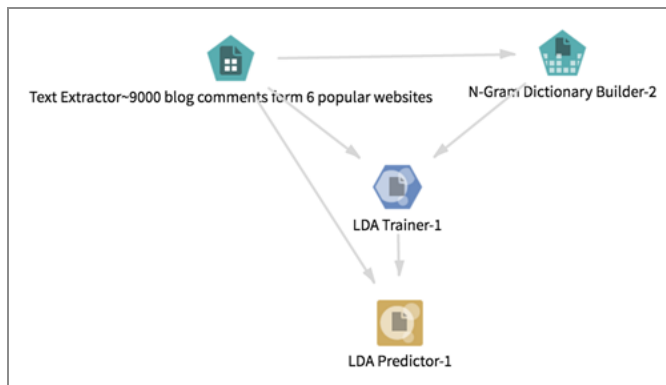
This is a semi-terminal operator. It can be connected to any subsequent operator at design time, but it does not transmit the full output schema until the user runs the operator. The partial output schema at design time is only the first columns of the output (**Doc ID Column**, **Pass Through** columns selected and **top\_topics\_summary** column). After running it, the output schema is automatically updated, and subsequent operators turn red in case the UI parameters selection is no longer valid.

**i Note:** The final output schema of the Transpose operator is cleared you take one of the following actions.

- Change the configuration properties of the Transpose operator.
- Change the input connected to the Transpose operator.
- Clear the step run results of the Transpose operator.

In this case, the output schema transmitted to subsequent operators again becomes the partial schema defined at design time (hence, subsequent operators can turn invalid), and you must run the Transpose operator again to transmit the new output schema.

## Example



## LDA Trainer

LDA (Latent Dirichlet Allocation) is an unsupervised text-mining algorithm used to analyze collections of unstructured documents.



## Information at a Glance

Parameter	Description
Category	NLP
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

For more information about using LDA, see [Unsupervised Text Mining](#) and [LDA Training and Model Evaluation Tips](#).

## Input

The LDA trainer requires two inputs:

- An HDFS tabular data set with at least a unique document ID column and a text content column (for example, the output of the Text Extractor operator). Note: The Text Featurization of 'Text Column' that converts raw text to N-Grams features is included in the LDA Trainer operator.
- An N-Gram Dictionary Builder (most likely created from the same tabular input connected to the LDA Trainer).

## Bad or Missing Values

If a row contains a null value in at least one of the **Doc ID** column or **Text Column**, the row is removed from the data set. The number of null values removed can be listed in the **Summary** section of the output (depending on the chosen option for **Write Rows Removed Due to Null Data To File**).

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Document ID Column</b>	The column that contains the document unique ID (String, Int, or Long). This column must contain unique IDs. If duplicate values are found, the job fails with a meaningful error message.
<b>Text Column</b>	The column that contains the document text content (that is featurized based on N-Gram Dictionary Builder and Featurization input parameters).
<b>N-Gram Selection Method</b>	<p>The criteria used to choose the n-grams to pull out as word count features for the LDA (the criteria is applied to the training corpus; that is, the corpus used to create the N-Gram Dictionary).</p> <p>Choose from the following options.</p> <ul style="list-style-type: none"> <li>• <b>Appear in the Most Documents</b> (the default)</li> <li>• <b>Appear in the Fewest Documents</b></li> <li>• <b>Feature Hashing</b> - Use all of the n-grams, but reduce the dimensionality of the feature space by storing each n-gram as a hash value using the number of buckets specified by the <b>Max Number of N-grams</b> parameter below. The columns then represent the summed valued of all the n-grams associated with one hash value.</li> <li>• <b>Occur Least Frequently (Entire Corpus)</b></li> <li>• <b>Occur Most Frequently (Entire Corpus)</b></li> </ul>



Parameter	Description
	<p><b>Note:</b> If "Feature Hashing" option is selected:</p> <ul style="list-style-type: none"> <li>The n-gram features used for LDA and LDA Predictor (for the Topic Description data set) have the name format "ngramX", where X is the number of the n-gram generated.</li> <li>The list of real n-grams included in each feature used in LDA (from hashing) is stored in the HDFS file <code>conversion_table</code>, and displayed in the LDA Trainer visual output (the <b>N-Grams from Feature Hashing</b> tab) .</li> </ul>
<b>Maximum Number of Unique N-Grams to Select (Feature Hashing Size)</b>	<p>The number of n-grams to select from the dictionary to use as features. If you selected Feature Hashing for the <b>N-Gram Selection Method</b> parameter, this represents the size of the hash set. In either case, the total number of features in the featurized data set (used as input for LDA) is no greater than this number.</p> <p>Default value: <b>500</b>.</p>
<b>Number of Topics (k)</b>	Select the number of topics to train the LDA.
<b>Maximum Terms to Describe Topics</b>	Select the maximum number of n-grams to describe topics. This parameter sets the number of top n-grams (= n-grams with highest weights for each topic) to use in the <b>Topic Description</b> data set output.
<b>Maximum Topics to Describe Documents</b>	<p>Select the maximum number of topics to describe documents. This parameter sets the number of top topics (= topics with highest weights for each document) to use in the <b>Top Topic Distribution</b> data set output.</p> <p><b>Note:</b> This parameter also applies to the LDA Predictor connected to the LDA Trainer for the following columns of the output:</p> <ul style="list-style-type: none"> <li><b>top_topics_summary</b></li> <li>pairs of columns <b>topic_ranked_x</b>, <b>weight_topic_ranked_x</b></li> </ul>
<b>Document Concentration</b>	<p>Dirichlet parameter for prior over documents' distributions over topics. Only symmetric priors are supported, which is the most commonly used</p>

Parameter	Description
<b>(<math>\alpha</math>) (enter -1 for default)</b>	<p>configuration in LDA (= uniform k-dimensional vector with value <math>\alpha</math>).</p> <p>Larger values of <math>\alpha</math> encourage smoother inferred distributions (that is, each document is likely to contain a mixture of most of the topics), whereas lower values encourage sparse distributions.</p> <p>Values should be positive. Entering <b>-1</b> results in default behavior for Online optimizer (=uniformkdimensional vector with value 1.0 / k).</p>
<b>Topic Concentration (<math>\beta</math>) (enter -1 for default)</b>	<p>Dirichlet parameter for prior over topics' distributions over terms. Larger values encourage smoother inferred distributions (that is, each topic is likely to contain a mixture of most of the words), whereas lower values encourage sparse distributions.</p> <p>Values should be positive. Entering <b>-1</b> results in default behavior for Online optimizer (=uniformvector with value 1.0 / k).</p>
<b>Maximum Number of Iterations</b>	<p>Limit on the number of iterations. Default value is <b>30</b>.</p> <p><b>Note:</b> It is important that you set enough iterations. Early iterations often return useless topics, but those topics improve dramatically after more iterations. Using at least 20 and possibly 50-100 iterations is often reasonable, depending on your data set.</p>
<b>Automatic Optimization of <math>\alpha</math></b>	<p>Indicates whether Doc Concentration (<math>\alpha</math>) - Dirichlet parameter for document-topic distribution - is optimized during training.</p> <p><b>Note:</b> Setting this parameter to <b>Yes</b> (the default) might slow down the training, but you might get better topics.</p>
<b>Mini-Batch Fraction</b>	<p>The fraction of the corpus sampled and used at each iteration. This parameter should be set in sync with <b>Maximum Number of Iterations</b> to ensure the entire corpus is used. <b>Maximum Number of Iterations * Mini-Batch Fraction</b> must be <math>\geq 1</math>, or a meaningful error occurs at design time.</p>
<b>Learning Rate (<math>\kappa</math>)</b>	<p>The learning parameter for exponential decay rate. It should be set between <b>[0.5, 1]</b> to guarantee asymptotic convergence. The learning rate at each iteration number (t) is computed with the formula <math>(\tau_0 + t)^{-\kappa}</math></p>

Parameter	Description
	The default value for $\kappa$ is <b>0.51</b> .
<b>Learning Parameter (<math>\tau_0</math>)</b>	A (positive) learning parameter that down-weights early iterations. Larger values make early iterations count less. The default value is <b>1024</b> .
<b>Use Checkpointing</b>	Select <b>Yes</b> (the default) or <b>No</b> . Checkpointing helps with recovery when nodes fail, and also helps with eliminating temporary shuffle files on disk, which can be important when LDA is run for many iterations.
<b>Write Rows Removed Due to Null Data To File</b>	<p>Rows with null values (in the Doc ID Column or the Text Column) are removed from the analysis. This parameter allows you to specify that the data with null values be written to a file.</p> <p>The file is written to the same directory as the rest of the output. The filename is bad_data.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

- A **Model Training Summary**, which describes the parameters selected, model training results, output location of the results, and the number of rows processed.

<b>Operator Type</b>	LDA Trainer
<b>Operator Name</b>	LDA Trainer-1
<b>Input Parameters</b>	
Document ID Column:	file_path
Text Column:	text_content
N-Gram Selection Method:	Appear in the Most Documents
Maximum Number of N-Grams Selected:	1000
Maximum Terms to Describe Topics:	50
Maximum Topics to Describe Documents:	6
<b>Training Data Statistics</b>	
Number of documents in corpus:	8849
Vocabulary size:	1000
<b>Model Parameters</b>	
Number of topics (k):	6
Prior for Document Concentration ( $\alpha$ ):	0.17 (default for Online optimizer)
Prior for Topic Concentration ( $\beta$ ):	0.17 (default for Online optimizer)
Optimizer:	Online Variational
Optimize Document Concentration ( $\alpha$ ):	true
Mini Batch Fraction:	0.1
Learning Rate ( $\kappa$ ):	0.51
Learning Parameter ( $\tau_0$ ):	1024.0
<b>Model Results</b>	
Training time:	366.31 seconds
Optimized Document Concentration ( $\alpha$ ):	[0.09,0.12,0.07,0.09,0.05,0.06]
Training data Log Likelihood (entire corpus):	-3.240807546E8
Training data Average Log Likelihood (per document):	-36623.43
Training data Log Perplexity (per token):	5.82
<b>Output</b>	
Input data size:	8850 rows
Input size after removing rows due to null values in file_path or text_content:	8849 rows
Rows removed due to null values in file_path or text_content:	1 rows (0%)
The LDA model is stored at: /tmp/alpine_out/emilie/Test_LDA_11340/LDA_Trainer_1_1473954098333/lda_model	
The Topic Description dataset is stored at: /tmp/alpine_out/emilie/Test_LDA_11340/LDA_Trainer_1_1473954098333/topics_description	
The Topic Distribution dataset is stored at: /tmp/alpine_out/emilie/Test_LDA_11340/LDA_Trainer_1_1473954098333/doc_topics_distribution_all	
The Top Ordered Topic Distribution dataset is stored at: /tmp/alpine_out/emilie/Test_LDA_11340/LDA_Trainer_1_1473954098333/doc_top_topic_distribution	

**Training data Log Likelihood (entire corpus):** Lower bound on the log likelihood of the entire corpus.

**Training data Average Log Likelihood (per document):** Log likelihood/total number of documents

**Training data Log Perplexity (per token):** Upper bound on the log perplexity per token of the provided documents given the inferred topics (lower is better). In information theory, [perplexity](#) is a measurement of how well a probability distribution (or probability model) predicts a sample, and is a common measure of model performance in topic modeling. More specifically, it measures how well the word counts of the documents are represented by the word distributions that the topics represent. A low perplexity indicates the probability distribution is good at

predicting the sample.

- **Topic Description (top X terms)** data set (stored in HDFS): top X terms per topic with their associated weights:

topic_id	term_ranked_1	weight_term_ranked_1	term_ranked_2	weight_term_ranked_2	term_ranked_3	weight_term_ranked_3	term_ranked_4
0	pm	0.0375	a011	0.0327	as	0.0227	a
1	as	0.0201	n_t	0.0188	peopl	0.0187	would
2	appl	0.0284	a	0.0235	n_t	0.0233	a011
3	instanc	0.0581	a_www	0.0581	a_www_tern	0.058	asemanticweb_org_ontologies_convers

- **Full Topic Distribution** data set (stored in HDFS): full documents' distribution over topics (not ordered, all topics displayed):

file_path	weight_topic_0	weight_topic_1	weight_topic_2	weight_topic_3
/txt/Dataset_blogdiscussions/Slashdot/Slashdot_part_1/10_02_05_1925203.instancedata.txt	0	0.2891	0	0.6359
/txt/Dataset_blogdiscussions/Slashdot/Slashdot_part_2/09_12_17_187248.instancedata.txt	0	0	0.0037	0.6108
/txt/Dataset_blogdiscussions/AndroidCentral/9647.txt	0.418	0.0016	0.5778	0.0011
/txt/Dataset_blogdiscussions/TSN/380115.txt	0.9982	0.0005	0.0003	0.0004

- **Topic Distribution (top X topics)** data set (stored in HDFS): top X topics per document with their associated weights:

file_path	topic_ranked_1	weight_topic_ranked_1	topic_ranked_2	weight_topic_ranked_2	topic_ranked_3	weight_topic_ranked_3
/txt/Dataset_blogdiscussions/Slashdot/Slashdot_part_1/10_02_05_1925203.instancedata.txt	3	0.6359	1	0.2891	4	0.0378
/txt/Dataset_blogdiscussions/Slashdot/Slashdot_part_2/09_12_17_187248.instancedata.txt	3	0.6108	4	0.339	5	0.0465
/txt/Dataset_blogdiscussions/AndroidCentral/9647.txt	2	0.5778	0	0.418	1	0.0016
/txt/Dataset_blogdiscussions/TSN/380115.txt	0	0.9982	1	0.0005	3	0.0004

- **Featurized data set** (stored in HDFS): featurized data set used as input for LDA algorithm:

file_path	number_of_tokens	normalized	number_unique	normalized_n	http	as	n_t	like	grr	us
/txt/Dataset_blogdiscussions/AndroidCentral/11022.txt	358	0	215	0.0001	1	0	1	0	2	5
/txt/Dataset_blogdiscussions/AndroidCentral/8498.txt	130	0	103	0.0001	1	2	1	0	2	1
/txt/Dataset_blogdiscussions/AndroidCentral/9350.txt	211	0	156	0.0001	1	1	2	0	4	0
/txt/Dataset_blogdiscussions/Businessinsider/1223.txt	568	0	367	0.0002	1	4	4	1	4	6
/txt/Dataset_blogdiscussions/Businessinsider/478.txt	189	0	159	0.0001	1	6	0	0	0	0

(Only if **Feature Hashing** option is selected for **N-Gram Selection Method**)

- **N-Grams from Feature Hashing** data set (stored in HDFS): list of n-grams included in each feature used in LDA (from hashing):

N_Gram_Label	N_Grams_in_Feature	Total_N_Grams_in_Corpus	Total_Number_of_Documents
ngram0	if you, ingesting data, credit must, and that, towards, dermtl and, 54 6.14, reserves of, first classify, adaptations, quantiles, awarded, conceptual, more and, 17 3.1, pp.37-50, august 2005, without referring, a whole, likelihood they, and natural, very important, error expected, mile hence, notified, create, html or, a state-of-the-art, are, does and, place of, [5 breiman, data 3.1, approach 77, dv 42, of minimizing, skewed "the, accurate, same country, predict each, side of, paper writing	363	55
ngram1	first", qualitative a.k, publishing' int, result is, improvement compared, beneficial to, research gap, bootstraps 60, disrupt, and illustrative, models significant, paper including, 17 3.2, 0.4128440 table, to assess, elaborated, call or, this system, second-generation, 4screener when, principal advantage, anthony, mistakes are, expanded scope, additional adapters, 238 295, the observation, compton east, rpart, impact on, the d, association, during, discussing and, violate these, 54 6.15, similarities bet	152	65
ngram2	0 35, class assigned, 'diabetes, fit the, care about, broader situations, significance 120, 3 structure, 1984 cross-validation, a normal, groundbreaking new, in all, storing, code samples, 0.1747088 0.1480864, potential, meaningless, two important, "1" ---, data augmentation, he/she can, in a, associations, " for, firsttimec, cutt getting, 4 358-382, introduction can, http://www.acm, all or, "1", is locally, useful hints, constant cm, able, measureqm t, make, made of, wrangler, challenges even, 1993	123	51

## Data Output

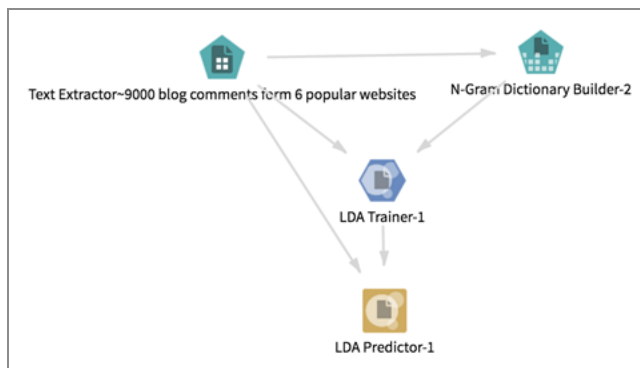
The main output of the LDA Trainer is an LDA model that can be connected to the LDA Predictor operator to predict topics on new documents. Several HDFS tabular data sets created from the training documents (**Topic Description, Full Topic Distribution, Top Topic Distribution, Featurized data set**) also are stored in HDFS and can be dragged onto the canvas for further analysis.

## Additional Notes

### Extra Resources

- [https://en.wikipedia.org/wiki/Latent\\_Dirichlet\\_allocation](https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation)
- <https://github.com/thvasilo/spark/blob/master/docs/mllib-clustering.md>
- <https://databricks.com/blog/2015/03/25/topic-modeling-with-lda-mllib-meets-graphx.html>

## Example



## Prediction Operators

Prediction (Predict) operators are used to apply a particular modeling algorithm operator to a new dataset for prediction purposes.

### Chi Square, Goodness of Fit

Computes a Pearson's Chi Square test for goodness of fit of a distribution.



#### Information at a Glance

Parameter	Description
Category	Predict
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

For more information, see [Pearson's Chi Square Operations](#) for information about the Chi Square operators.

#### Input

This operator requires two inputs: a data set that contains the observed events and their frequency, and a data set that represents the expected frequency of the events. Both data sets should have a categorical column that represents event names and a numeric column with the frequency of events. Frequencies must be absolute frequencies (using relative frequencies - ratios - results in an inaccurate statistic).

The following inputs (pre-aggregated or not) are considered valid (in other words, the frequencies for a unique category can be split into multiple rows that are aggregated at runtime).

Conversion	Count
Churned	500
Retained	250
Inactive	250

Conversion	Count
Churned	499
Churned	1
Inactive	250
Retained	250

The two input data sets, the expected frequencies and the observed frequencies, do not need to have the exact same data format, but they should correspond to the same distinct events, because the chi square statistic is a pairwise comparison between observed and expected frequency for each event outcome.



**Note:** If the sum of absolute frequencies in the **Observed Frequency Column** is not equal to the sum of absolute frequencies in the **Expected Frequency Column**, the MLLib Chi Square rescales automatically the frequencies in the **Expected Frequency Column** using the factor  $\text{observed\_freq\_sum} / \text{expected\_freq\_sum}$  before computing the Chi Square statistic.

### Bad or Missing Values

Missing or null values must be removed from both the even name and event frequency column in each of the two input data sets. If rows are removed in this step, then they are reported in the **Summary** tab of the visual output. Depending on the selected value



of the **Write Rows Removed Due to Null Data To File** parameter, the rows removed due to null data in either of the data sets can be written to a file.

## Restrictions

This operator can accept any two data sets that each have a numeric and categorical column. However, the results are not meaningful unless the categorical columns in the two data sets contain corresponding event names and the numeric columns represent absolute frequencies. For example, frequencies should be non negative and the frequencies in one data set should be (aggregated or not) counts of events.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Observed Dataset</b>	The name of the input data set (both the input data sets appear in the dropdown) that corresponds to the observed frequencies.
<b>Observed Events Column</b>	The event names column in the observed data set selected above. This should be a categorical column. The events might occur more than once in the columns.
<b>Observed Frequency Column</b>	A column in the observed data set with a measure of absolute frequency of the events.
<b>Expected data set</b>	The second data set, which represents the theoretical expected frequencies of the events in the observed data set.
<b>Expected Events Column</b>	The event names column in the specified <b>Expected data set</b> . This column should be a categorical column. The events can occur more than once in the column. In this case, the frequency is the sum of the frequencies of the events with the same name scaled by the sum of all the frequencies. The events in the observed events column should correspond with the events in the expected column. The analysis can be completed only on events that

Parameter	Description
	appear in both data sets.
<b>Expected Frequency Column</b>	The absolute frequency of each event in the expected events column.
<b>Significance Threshold</b>	The confidence level under which the null hypothesis is rejected. Practically, this value is used to determine the <b>Reject Null Hypothesis</b> column in the output. We reject the null hypothesis if the P-Value (which represents the probability that the variance in the event distributions occurred due to chance) is less than this value.
<b>Write Rows Removed Due to Null Data To File</b>	<p>Rows with null values in the event name or frequency columns are removed from each of the input data sets. This parameter allows you to specify that the data with null values are written to a file. The parameter applies to null values in both observed and expected input data set.</p> <p>Null data in the observed data set is written to the following location:</p> <pre>@default_tempdir/tsds_out/@user_name/@flow_name/@operator_name_ uuid_bad_data_observed</pre> <p>Null data in the expected data set is written to the following location:</p> <pre>@default_tempdir/tsds_out/@user_name/@flow_name/@operator_name_ uuid_bad_data_expected</pre> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI.</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	Select the type of compression for the output.

Parameter	Description
	<p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

This operator produces the following three tabs of visual output.

- **Output** - Shows a preview of the output that is passed to the next operator. The output is the results of the Chi Square goodness of fit test. The structure of this

output is the same as the Chi Square, Independence Test Operator; however, the degrees of freedom correspond to the number of event outcomes and there is only one independent variable.

Results - Chi Square, Goodness of Fit-StatePopulation					
Output Event Frequencies Summary	Independent Variable	Degrees of Freedom	Chi Square Statistic	P-Value	Reject Null Hypothesis
	Goodness of Fit	50	0.009627425	1	No

- **Event Frequencies** - Displays a table of the scaled frequencies for the events in both of the data sets. Notice that only events which appear in both the observed and expected data set are included in this table. If events are dropped during the join, a message that lists the first ten of these events appears in the logs. This is the table that forms the basis of the Chi Square analysis.

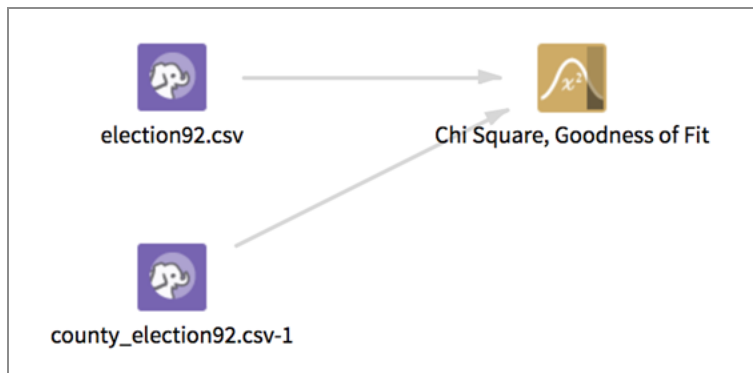
Output Event Frequencies Summary	Event Name	Expected Frequency	Observed Frequency
	cat9	10	45
	cat15	10	23
	cat3	7	18
	cat6	10	10
	cat1	24	35

- **Summary** - Like most of the TIBCO Data Science - Team Studio transformation operators, this operator contains a summary tab that provides the location of the results on HDFS and reports if any rows are dropped due to null values. In the case of this operator, we report rows dropped for both the **Expected data set** input and the **Observed data set** input. The exact format of the bad data reporting is controlled by the **Write Rows Removed Due to Null Data To File** parameter.

## Data Output

The Results of the Chi Square test (previewed in the **Output** tab of the visual results) are passed to succeeding operators. The results are a tabular data set and can connect to any Hadoop operator that expects tabular input.

## Example



## Chi Square, Independence Test

Determines whether categorical columns are statistically independent of a categorical dependent variable column.



## Information at a Glance

Parameter	Description
Category	Predict
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

See [Pearson's Chi Square Operations](#) for information about the Chi Square operators.

## Input

The operator requires a tabular input on Hadoop. The input should contain at least two

categorical columns, one that represents the independent variable, and one that represents the dependent variable. The operator can compute the chi square test on multiple independent columns in one run - in this case, each independent column is compared to the dependent column and forms a row in the output data set of the chi square test metrics.

### Bad or Missing Values

Before computing any of the chi square tests, rows with null values in any of the independent or dependent columns are dropped. These rows are reported and written to a file according to the value of the `Write Rows Removed Due to Null Data` parameter.

### Restrictions

Scalability issues might result if there are many distinct values in the categorical columns (more than 1,000) or if many independent columns are selected.

### Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	A categorical column.
<b>Independent Columns</b>	One or more categorical columns to compare to the dependent columns. The null hypothesis in this case is that the distribution of categories in the independent and dependent columns are statistically independent.
<b>Significance Threshold</b>	The confidence level under which we reject the null hypothesis. Practically, this value is used to determine the <b>Reject Null Hypothesis</b> column in the output. We reject the null hypothesis if the P-Value (which represents the probability that the variance in the event distributions occurred due to chance) is less than this value.

Parameter	Description
<b>Use Fisher's Exact Test instead of Chi Square</b>	<p>Select to use the Fisher's exact test rather than the more common, and more robust, Pearson's Chi Square Test.</p> <div> <p><b>Note:</b> For computational and theoretical reasons, the Fisher's exact test is appropriate only for 2 x 2 tables; that is, when the independent and dependent variables have only two possible outcomes and when the number of observations is extremely small. We cannot perform a Fisher's exact test in cases when the cells in the table have a value greater than five.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - Compute a Fisher's Exact Test for all the independent variables.</li> <li>• <b>No</b> (the default) - Use the Chi Square test.</li> </ul> </div>
<b>Write Rows Removed Due to Null Data To File</b>	<p>Rows with null values in at least one of the independent columns or the dependent column are removed from the analysis. This parameter allows you to specify that the data with null values are written to a file.</p> <p>The file is written to the following location:</p> <p>@default_tempdir/alpine_out/@user_name/@flow_name/@operator_name_uuid_bad_data</p> <p>From the dropdown list, specify one of the following.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI.</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

Results - Chi Square Test					
Output Summary	Independent Variable	Degrees of Freedom	Chi Square Statistic	P-Value	Reject Null Hypothesis
	outlook	2	3.5466667	0.1697662	No
	wind	1	0.9333333	0.3339983	No



## Data Output

The results of the statistical test for each independent variable are output to the next operator. However, the format of these results differ depending on whether a Chi Square test or a Fisher's exact test are used.

In the case of a Chi Square test, we output a table with the following columns and one row for each independent column selected:

- **Independent Variable:** The name of the independent column
- **Degrees of Freedom:** The number of degrees of freedom. The degrees of freedom differ between the Chi Square test of independence and the Chi Square test for Goodness of Fit.
- **Chi Square Statistic:** The test statistic, a decimal. The test is a measure of difference between the observed and expected distribution.
- **P-Value:** The probability that the two samples are from the same distribution. Lower P-Values indicate a greater relationship between the independent and dependent variables. The P-Value is a function of the degrees of freedom and the test statistic. In general, a high chi square statistic for the same degrees of freedom leads to a lower P-Value. By convention, tests that yield P-Values of greater than 0.05 (reject the null hypothesis 0.05 percent of the time) are considered to show significance.
- **Reject Null Hypothesis:** Whether the P-Value was less than the alpha value set in the parameters. (The default alpha value is 0.05).

In the case of the Fisher's exact test, the output is only the **Independent Variable**, **P-Value**, and **Reject Null Hypothesis** columns. This is because the Fisher's exact test does not compute a test statistic from which the probability is estimated, but rather a probability directly and does not use degrees of freedom.

Output Summary	Independent Variable	P-Value	Reject Null Hypothesis
	wind	0.6	No

## Classifier (DB)

Uses any input classification model to apply a classification prediction to the input data set.



### Information at a Glance

Parameter	Description
Category	Predict
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

**i Note:** The Classifier (DB) operator is for database data only. For Hadoop data, use the [Classifier \(HD\)](#) operator.

### Algorithm

The TIBCO Data Science - Team Studio Classifier operator is used to predict the probability of the occurrence of the event based on the model generated by the training of Alpine Forest, Decision Tree, K-Means (Hadoop), Logistic Regression, Naive Bayes, Neural Network, or SVM Classification operator models.

### Input

The input data set must contain the columns such that the names are the same as the columns in the data set selected for model training with the exception of the dependent column. The Classifier operator must have both of the following.

- An input Classification model.
- An input data set against which the model is applied.

The model preceding the Classifier operator can be any of the following. The Classifier operator can take multiple models from the preceding operators, not just one.

- Alpine Forest
- Decision Tree
- K-Means
- Logistic Regression
- Naive Bayes,
- SVM Classification

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	Advanced database settings for the operator output. Available only for <b>TABLE</b> output.  See <a href="#">Storage Parameters dialog</a> for more information.
<b>Drop If Exists</b>	Specifies whether to overwrite an existing table. <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

The Classifier outputs its prediction columns with the columns of the input data set into a prediction table location specified by user.

The data rows of the output table/view displayed (up to 2,000 rows of the data).

For example, the output for a dependent column, `srsdlqncy`, might look like the following.

srsdlqncy	P_srsdlqncy	C_srsdlqncy	C_srsdlqncy_details
0	0	0.9781	{{"1":"0.021943216515517477","0":"0.9780567834844826"}}
0	0	0.9781	{{"1":"0.021943216515517477","0":"0.9780567834844826"}}
0	0	0.9781	{{"1":"0.021943216515517477","0":"0.9780567834844826"}}
0	0	0.9781	{{"1":"0.021943216515517477","0":"0.9780567834844826"}}
0	0	0.9781	{{"1":"0.021943216515517477","0":"0.9780567834844826"}}
0	0	0.9781	{{"1":"0.021943216515517477","0":"0.9780567834844826"}}

### Data Output

The Classifier operator outputs the following standardized three prediction columns:

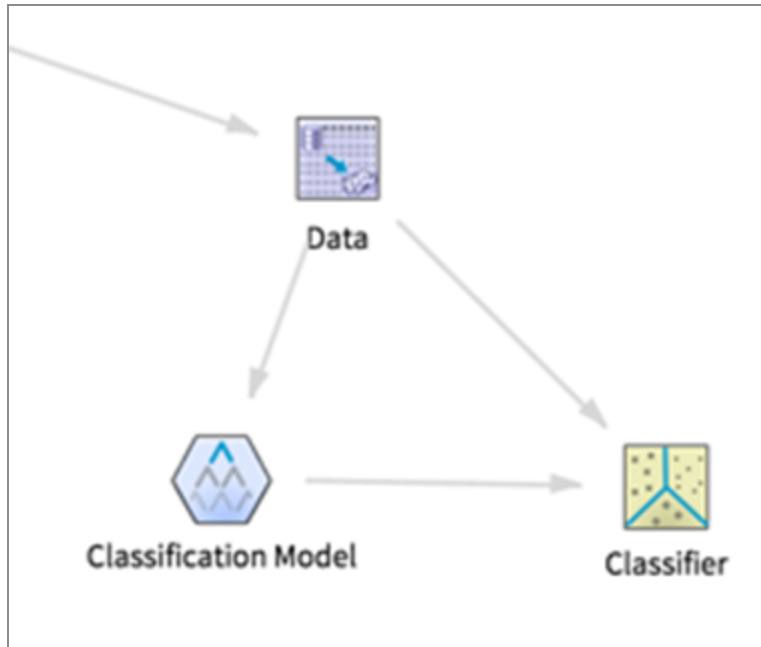
- **P\_dependent\_column\_name**: The predicted value which should be one of the possible returning values of the dependent column.
- **C\_dependent\_column**: The confidence of obtaining the result being the **P\_dependent\_column\_name** predicted value.
- **C\_dependent\_column\_details**: The confidence values associated with the dependent column's possible values.

**i Note:** If the Classifier operator has more than one input model, the resulting output has the three prediction columns per input model, and the column names are prepended with the input model operator's name.

## Data Output

Connect this operator to succeeding operators.

## Example



## Classifier (HD)

Uses any input classification model to apply a classification prediction to the input data set.



## Information at a Glance

Parameter	Description
Category	Predict
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce



**Note:** The Classifier (HD) operator is for Hadoop data only. For database data, use the [Classifier \(DB\)](#) operator.

## Algorithm

The TIBCO Data Science - Team Studio Classifier operator is used to predict the probability of the occurrence of the event based on the model generated by the training of Alpine Forest, Decision Tree, K-Means (Hadoop), Logistic Regression, Naive Bayes, Neural Network, or SVM Classification operator models.

## Input

The input data set must contain the columns such that the names are the same as the columns in the data set selected for model training with the exception of the dependent column.

The Classifier operator must have both of the following.

- an input Classification model.
- an input data set against which the model is applied.

The model preceding the Classifier operator can be any of the following. The Classifier operator can take multiple models from the preceding operators, not just one.

- Alpine Forest
- Decision Tree

- K-Means
- Logistic Regression
- Naive Bayes,
- SVM Classification

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Store Results?</b>	Specifies whether to store the results. <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	The HDFS directory where the results of the operator are stored. This is the main directory, the sub-directory of which is specified in <b>Results Name</b> . Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.
<b>Results Name</b>	The name of the file in which to store the results.
<b>Overwrite</b>	Specifies whether to delete existing data at that path and file name. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Compression</b>	Select the type of compression for the output. Available Parquet compression options are the following. <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>no compression</li> </ul> <p>Available Avro compression options are the following.</p> <ul style="list-style-type: none"> <li><b>Deflate</b></li> <li><b>Snappy</b></li> <li>no compression</li> </ul>

## Output

### Visual Output

The Classifier outputs its prediction columns with the columns of the input data set into a prediction table location specified by user.

The data rows of the output table/view displayed (up to 2000 rows of the data).

For example, the output for a dependent column, srsdlqncy, might look like the following.

srsdlqncy	P_srsdlqncy	C_srsdlqncy	C_srsdlqncy_details
0	0	0.9781	{{"1":"0.021943216515517477","0":"0.9780567834844826"}}
0	0	0.9781	{{"1":"0.021943216515517477","0":"0.9780567834844826"}}
0	0	0.9781	{{"1":"0.021943216515517477","0":"0.9780567834844826"}}
0	0	0.9781	{{"1":"0.021943216515517477","0":"0.9780567834844826"}}
0	0	0.9781	{{"1":"0.021943216515517477","0":"0.9780567834844826"}}
0	0	0.9781	{{"1":"0.021943216515517477","0":"0.9780567834844826"}}

### Data Output

The Classifier operator outputs the following standardized three prediction columns:

- **P\_dependent\_column\_name:** The predicted value which should be one of the possible returning values of the dependent column.
- **C\_dependent\_column:** The confidence of obtaining the result being the **P\_dependent\_column\_name** predicted value.
- **C\_dependent\_column\_details:** The confidence values associated with the



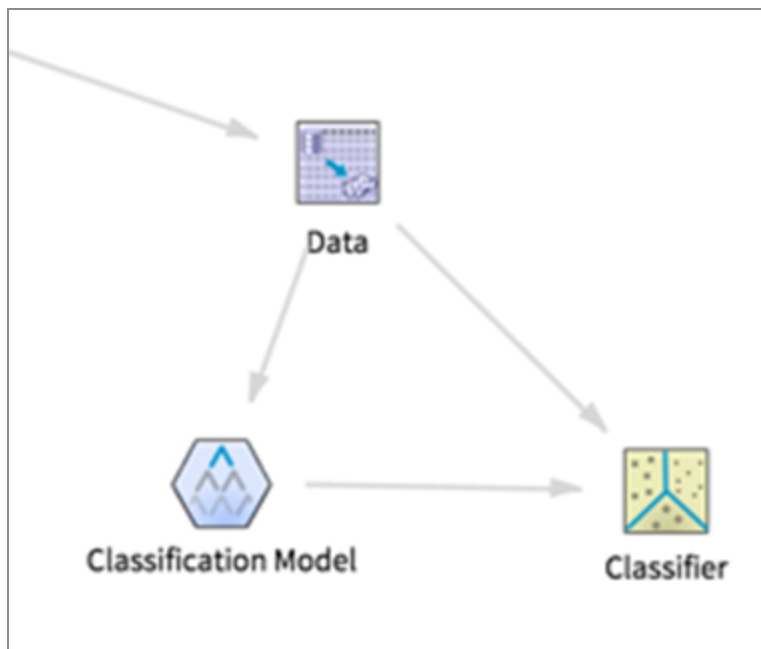
dependent column's possible values.

**i Note:** If the Classifier operator has more than one input model, then the resulting output has the three prediction columns per input model, and the column names are prepended with the input model operator's name.

## Data Output

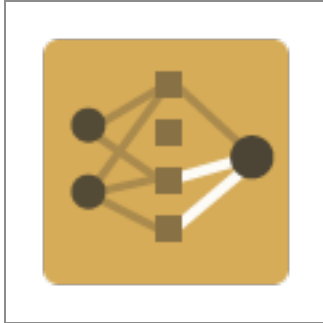
Connect this operator to succeeding operators.

## Example



## Collaborative Filter Predictor

Outputs predicted ratings for products using the Collaborative Filtering model created by the trainer. Uses both the model trained by the Collaborative Filter Trainer and a data set.



## Information at a Glance

Parameter	Description
Category	Predict
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

## Input

Output from a [Collaborative Filter Trainer](#) operator and a tabular data set with a column of user IDs and product IDs.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Users Column</b>	Column in the data set that contains user IDs.

Parameter	Description
<b>Products Column</b>	Column in the data set that contains product IDs.
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

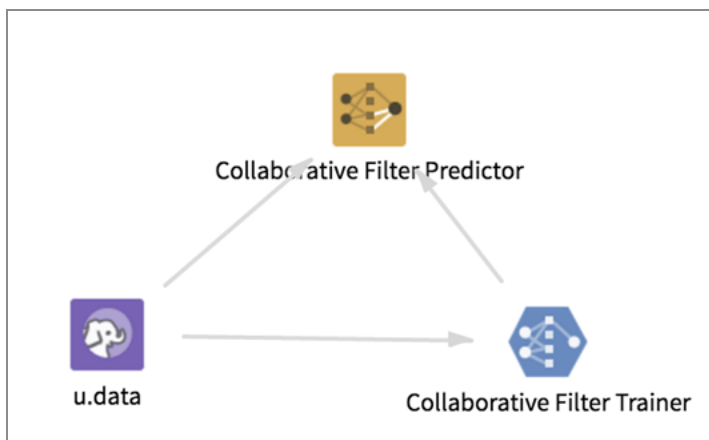
### Visual Output

User	Product	Rating	TimeStamp	Predicted_Rating
396	322	4	8.8464579e+8	3.37412358
21	262	4	8.74950931e+8	3.43727621
670	659	5	8.77974699e+8	3.96859118
532	301	4	8.74999563e+8	3.74628753
323	678	2	8.7873891e+8	2.44337008
642	1,029	3	8.85606271e+8	2.70353099

### Data Output

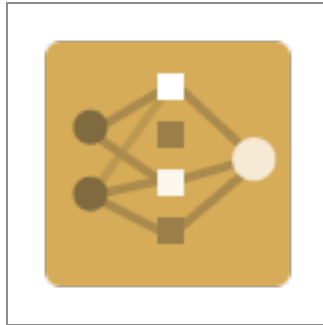
An HDFS data set with an extra column that describes the predicted rating for each product/user.

### Example



## Collaborative Filter Recommender

Using the model trained by the Collaborative Filter Trainer, outputs recommendations for those users or products.



## Information at a Glance

Parameter	Description
Category	Predict
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

## Input

The output from a Collaborative Filter Trainer, as well as an HDFS data set.

## Restrictions

The column selected must represent either user IDs or product IDs.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.

Parameter	Description
<b>Generate Recommendations for</b>	<p>Select a column in your data set that represents either users or product IDs.</p> <p>To see a list of products that a user might like, choose <b>User</b> (the default).</p> <p>To see a list of users who might like a product, choose <b>Product</b>.</p>
<b>This Column Represents</b>	Indicates what the <b>Generate Recommendations for</b> column represents - <b>Users</b> (the default) or <b>Products</b> .
<b>Number to Recommend</b>	<p>Specify the number of recommendations to generate.</p> <p>Range: 1-100.</p> <p>Default value: <b>5</b>.</p>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>

Parameter	Description
	Available Avro compression options. <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

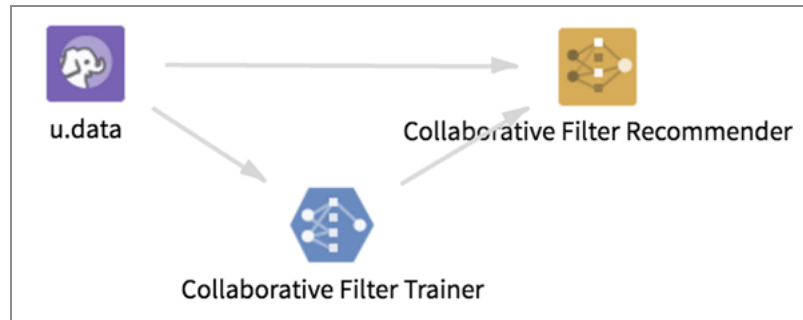
This output shows each user ID in the users column selected, then lists five products they might enjoy and their predicted ratings.

user	product0	product0_rating	product1	product1_rating	product2	product2_rating	product3	product3_rating	product4	product4_rating
297	884	6.95119999	913	6.680994	1,664	6.2205752	899	6.07428008	1,529	5.93130937
153	1,154	10.72211588	868	10.61282178	1,496	10.2423442	1,397	10.21838736	1,607	9.88495922
180	1,643	6.48412191	1,431	6.24544394	1,256	6.16269741	1,585	6.02268534	361	5.91729725
324	850	9.64859589	913	9.5465684	817	8.61214246	1,155	8.32418617	1,288	7.91253723
369	1,631	8.28278607	1,275	8.01184614	1,643	7.07024656	793	6.99015521	867	6.83672082
513	1,131	8.4735594	1,386	8.3847912	1,160	8.26145964	1,463	8.16734635	1,623	7.91249622
171	1,463	6.74967155	1,643	6.60719361	1,367	5.72592704	1,431	5.65710136	1,386	5.62951246
729	1,386	10.57639266	1,427	10.38383751	1,438	10.00473777	1,260	9.24540637	1,319	9.14175452
639	1,233	6.6661531	1,529	6.33510941	899	6.20195577	1,474	6.14804246	1,664	5.87075755
909	850	9.94736148	1,496	9.60310149	1,368	8.40782443	1,418	7.92540091	1,138	7.87962954

### Data Output

This data can be further manipulated by other operators in your workflow. It is passed on as a tabular HDFS data set. You can find the storage location of the recommendation table by referring to the **Summary** section of the results pane, as shown in the following example.

Output  
The recommendations are stored at  
/tmp/alpine\_out/allison/CustomOperator/Collaborative\_Filter\_Recommender\_1464796102154

**Example**

## K-Means Predictor - MADlib

The K-Means predictor (MADlib) operator output is simply the assignment of the input data members to the  $k$  number of clusters, the centroids already predetermined by the K-Means (MADlib) operator.



### Information at a Glance

Parameter	Description
Category	Predict
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

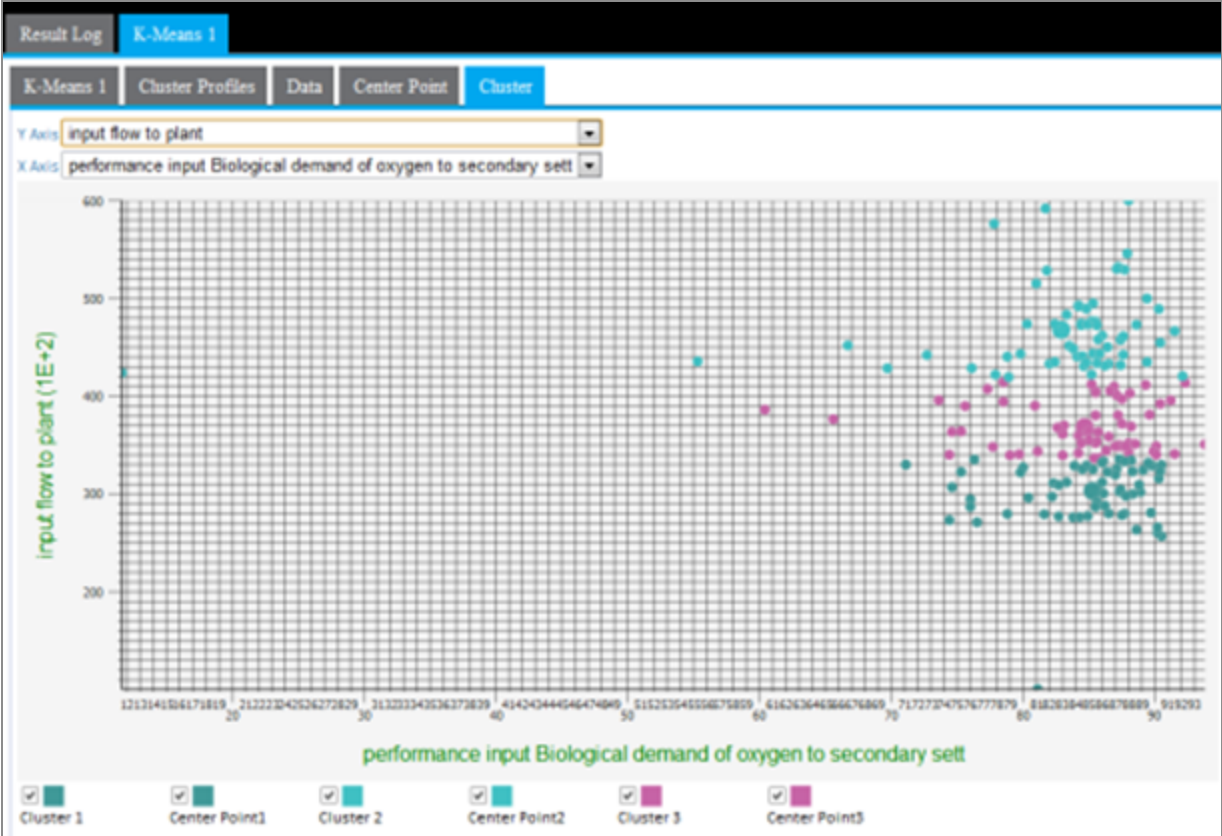
Unlike the Regression or Decision Tree/CART operators, the k-means predictor does not provide a final answer or prediction. Rather, it provides an overall understanding of the inherent structure of the data set the modeler is analyzing. This might be very helpful for understanding the inherent demographic groupings of a consumer data set.



The first results tab in the following image shows the (cluster, distance) cluster assigned to each point and the distance between the point and the cluster centroid, the pid or point ID, and the position of the point itself.

Result Log	km_wine	K-means (MADlib)	K-means Predictor (MADlib)
madlibtestdata.km_wine			
Cluster			
(cluster, distance)	pid	position	
{1,7060.1266737805963}	1	{14.23,1.71,2.430000000000002,15.6,127,2.799999999999998,3.060000000000001,0.280000000000000}	
{1,1519.6418544949054}	3	{13.16,2.359999999999999,2.669999999999999,18.600000000000001,101,2.799999999999998,3.240000000000000}	
{0,29869.621734538905}	5	{13.24,2.589999999999999,2.870000000000000,21,118,2.799999999999998,2.689999999999999,0.280000000000000}	
{1,20730.16916378064}	7	{14.390000000000001,1.870000000000000,2.450000000000002,14.6,96,2.5,2.52,0.299999999999999}	
{1,10389.779633066308}	9	{14.83,1.639999999999999,2.169999999999999,14,97,2.799999999999998,2.98,0.289999999999999}	
{1,132187.26753592354}	11	{14.1,2.160000000000001,2.299999999999998,18,105,2.950000000000002,3.319999999999998,0.280000000000000}	
{1,30431.843103780644}	13	{13.75,1.73,2.410000000000001,16,89,2.600000000000000,2.759999999999998,0.289999999999999}	
{1,160513.03431449496}	15	{14.380000000000001,1.870000000000000,2.379999999999999,12,102,3.299999999999998,3.640000000000000}	
{1,18034.041699494923}	17	{14.300000000000001,1.919999999999999,2.720000000000002,20,120,2.799999999999998,3.140000000000000}	
{1,284713.69688306644}	19	{14.19,1.590000000000001,2.48,16.5,108,3.299999999999998,3.930000000000002,0.320000000000000}	
{0,47759.656429642673}	21	{14.06,1.629999999999999,2.279999999999998,16,126,3.3,1.699999999999999,0.239999999999999}	
{1,12450.287465923448}	23	{13.710000000000001,1.860000000000001,2.359999999999999,16.600000000000001,101,2.609999999999999}	
{0,79249.049500099558}	25	{13.5,1.810000000000001,2.609999999999999,20,96,2.529999999999998,2.609999999999999,0.280000000000000}	

The **Cluster** results tab displays a cluster graph, which is a visualization of each cluster's member values based on two of the variable dimensions used for the k-means analysis.



Input

Configuration

Parameter	Description
Notes	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
MADLib Schema Name	The schema where MADlib is installed in the database. MADlib must be installed in the same database as the input data set. If a madlib schema exists in the database, this parameter defaults to madlib
Points Column	The <b>Points</b> column in the input data set contains the array of attributes for each point.

Parameter	Description
	This parameter must be an array type.
<b>Distance Function</b>	<p>Calculates the difference between the cluster member's values from the cluster's centroid (mean) value. The distance can be calculated in the following different ways:</p> <p><b>Cosine</b> - Measures the cosine of the angle between two vectors. For <math>n</math> dimensional vectors <math>\mathbf{u} = (u_1, u_2, \dots, u_n)</math> and <math>\mathbf{v} = (v_1, v_2, \dots, v_n)</math>, it is calculated using the dot product formula as:</p> $\frac{\mathbf{u} \cdot \mathbf{v}}{\ \mathbf{u}\  \ \mathbf{v}\ } = \frac{\sum u_i v_i}{\sqrt{\sum u_i^2} \sqrt{\sum v_i^2}}.$ <p><b>Euclidean</b> - The square root of the sum of the squares of distances along each attribute axes. For <math>n</math> dimensional vectors <math>\mathbf{u} = (u_1, u_2, \dots, u_n)</math> and <math>\mathbf{v} = (v_1, v_2, \dots, v_n)</math>, it is calculated as:</p> $\sqrt{\sum_{i=1}^n (u_i - v_i)^2}.$ <p><b>Manhattan</b> - The Manhattan, or taxicab, metric measures the distance between two points when traveling parallel to the axes. For <math>n</math> dimensional vectors <math>\mathbf{u} = (u_1, u_2, \dots, u_n)</math> and <math>\mathbf{v} = (v_1, v_2, \dots, v_n)</math>, it is given by</p> $\sum_{i=1}^n  u_i - v_i .$ <p><b>Squared Euclidean</b> (the default) - The default method for calculating the straight line distance between two points. It is the sum of the squares of distances along each attribute axes. For <math>n</math> dimensional vectors <math>\mathbf{u} = (u_1, u_2, \dots, u_n)</math> and <math>\mathbf{v} = (v_1, v_2, \dots, v_n)</math>, it is calculated as</p>

Parameter	Description
	$\sum_{i=1}^n (u_i - v_i)^2$ <p><b>Tanimoto</b> - Measures dissimilarity between sample sets. It is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union. As with Dice similarity, 0-1 vectors are used to represent sets. Then the Jaccard Similarity of sets A and B represented by set-vectors <math>(a_1, a_2, \dots, a_n)</math> and <math>(b_1, b_2, \dots, b_n)</math> is given by:</p> $\frac{ A \cap B }{ A \cup B } = \frac{\sum_{i=1}^n a_i b_i}{\sum_{i=1}^n a_i + \sum_{i=1}^n b_i - \sum_{i=1}^n a_i b_i}$ <p><b>User defined</b> - Specified by the user. See <b>User-Defined Distance</b></p>
<b>User-Defined Distance</b>	<p>If, for <b>Distance Function</b>, you specify <b>User-defined</b>, provide the function.</p> <p>The modeler might try to start with the default <b>Squared Euclidean</b> method, then experiment with the various other calculation methods to determine whether the cluster results seem more intuitive or provide more business insight.</p>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

The **Cluster** results tab displays a cluster graph, which is a way to visualize each cluster based on two of the variable dimensions used in the k-means analysis. Although there is typically some overlap between members of the clusters, the goal is to minimize cluster overlap. For a perfect cluster analysis model, there would be zero overlap between the clusters for each variable analyzed.

The output can only be displayed in two dimensions at a time. Therefore, the modeler must review all the possible clustering diagrams in order to get an overall assessment of which attribute dimensions have the greatest influence on the clustering.

**Note:** The **Cluster Profiles** results tab provides a quick sense of which variables have the most unique distribution profile across clusters, so those variables would be good ones to further analyze in this cluster graph section.

Cluster graphing can be toggled on and off per cluster. Therefore, the graph can be viewed showing one cluster at a time, which helps understand just the spread of members per cluster and visually see their distance from their center, as in the following example only showing cluster3 results.



A lot of cluster overlap for two variables might indicate that they are not as significant in the cluster analysis, or that there is not much variation of the overall population for those particular variables. The following example shows a more intermingled cluster visualization when the y-axis dimension is changed from input flow to plant to output pH.



Another cause of cluster overlap might be that the variable values were not appropriately normalized before the analysis was run. For example, when minimizing the distance in the cluster, a difference in pH of "7" is not as significant as the difference in input flow to plants value of "10,000."

## Data Output

## PCA Apply

Uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of uncorrelated variables (principal components).



## Information at a Glance

Parameter	Description
Category	Predict
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce



**Note:** This operator is deprecated and will be removed in a future release.

The PCA Apply operator is used in conjunction with the PCA operator. PCA, or Principal Component Analysis, is a multivariate technique for examining relationships among several quantitative variables. Depending on your data source, see either [PCA \(DB\)](#) or [PCA \(HD\)](#) for more information about PCA modeling and the PCA operator configuration.

The PCA (HD) operator analyzes the data for determining the principal components matrix transformation, but needs the PCA Apply operator to actually transform the data before it passes the reduced variable set into any following operator.



**Note:** For database workflows, the PCA operator both analyzes the data for principal components and also "applies" the matrix transformation to the original data passed into the PCA Operator. However, for Hadoop workflows, PCA and PCA Apply Operators are separated operators, giving the user the choice to apply the derived matrix transformation either to the original training data set or to a new data set (with the same variables).

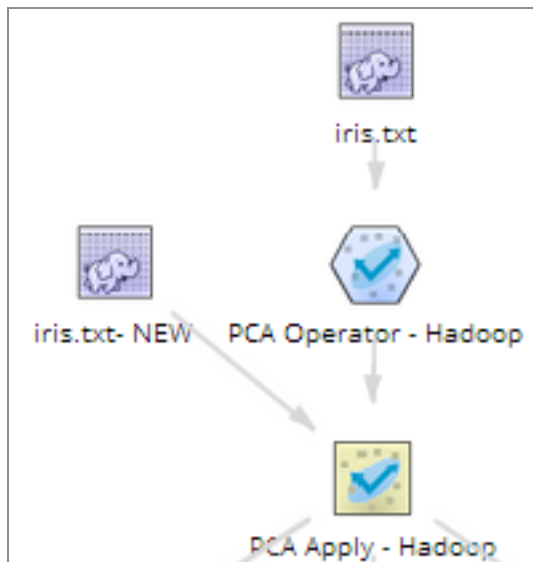
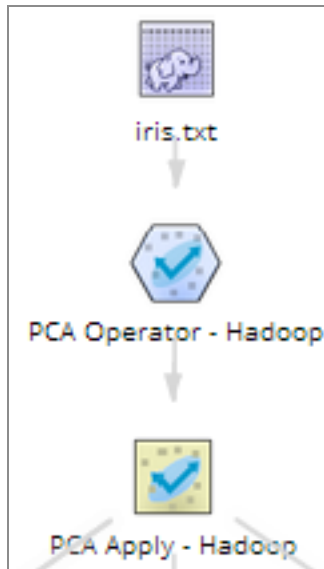
## Algorithm

The PCA Apply Operator applies the principal component matrix transformation algorithm defined by the PCA operator against the input data source.

## Input

If the matrix transformation is to be applied to the source data set, no other input is required. However, if the matrix transformation is to be applied against a new data source, the data source to be transformed must also be an input into the PCA Apply Operator.

The two possible flow combinations for input into the PCA Apply operator are shown below for an example data set source called Iris. The PCA Apply operator is applied against either the training iris.txt data set or the iris.txt-NEW data set.





## Restrictions

The PCA Apply operator can only be used with a PCA operator as input, applied against a Hadoop data source.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Target Number of Features</b>	<p>Dictates the number of principal components to define. This value must be less than or equal to the Maximum Rank for Distributed Mode parameter value set for the associated PCA operator. See <a href="#">Principal Component Analysis</a> for details.</p> <p><b>Note:</b> This value must be less than or equal to the number of columns in the source data set that was passed into the PCA operator.</p> <p>Default value: <b>5</b>.</p>
<b>Carryover Columns</b>	<p>You can choose to keep columns from the original input data (that was passed into the PCA operator) "untransformed" and included in the PCA Apply operator output.</p> <p>In this case, click <b>Carryover Columns</b> button to open the dialog for selecting the columns to retain in the result table.</p>
<b>Store Results?</b>	<p>Specifies whether to store the results.</p> <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	<p>The HDFS directory where the results of the operator are stored. This is the main directory, the sub-directory of which is specified in <b>Results Name</b>. Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.</p>

Parameter	Description
<b>Results Name</b>	The name of the file in which to store the results.
<b>Overwrite</b>	<p>Specifies whether to delete existing data at that path and file name.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options are the following.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options are the following.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>

## Output

### Visual Output

An overview of the new reduced principal components data set.

#### **alpine\_pcaattr[0-5]+**

Each of the newly derived principal component columns is provided, along with their values for the new transformed data set. In this case, the source Iris data set with hundreds of variables was reduced to only five principal component variables and saved in Hadoop file format.

#### **Carryover Columns**

Any carryover columns from the original data set that were specified in the PCA operator configuration, such as any necessary unique ID key or the dependent variable to predict in the following model, are displayed here.

In this example, the "class" column was carried over to be used in the following Alpine Forest model.

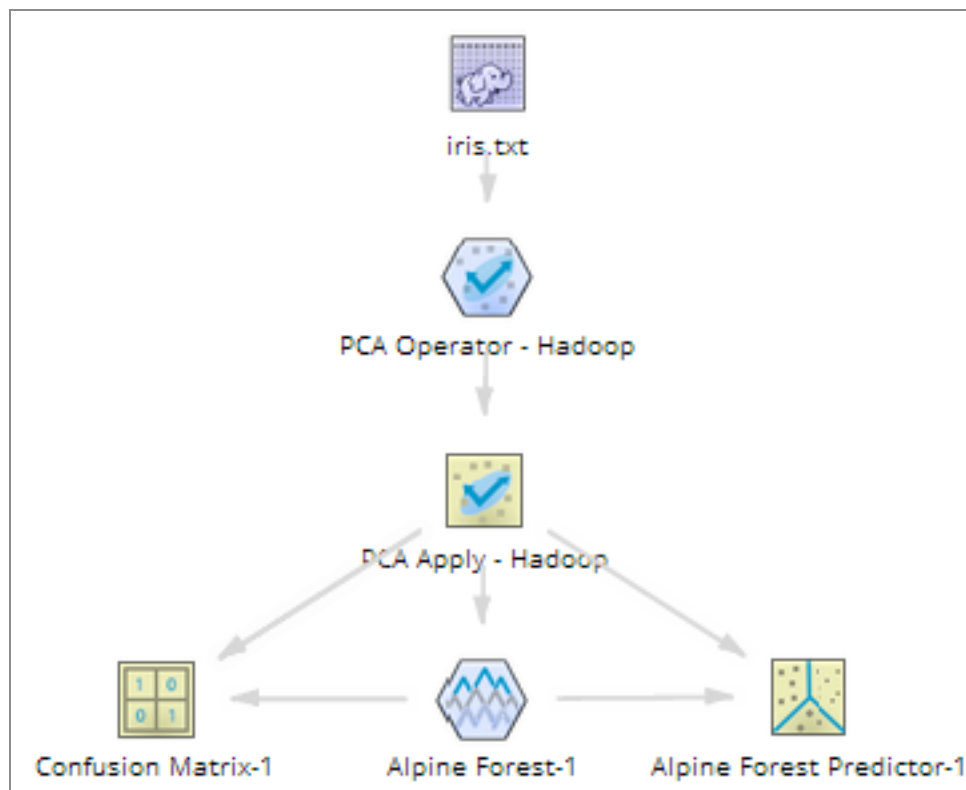
RESULTS - PCA Apply - Hadoop					
"class"	pc_1	pc_2	pc_3	pc_4	pc_5
Iris-setosa	-2.7265	-0.263	0.2206	-0.762	-0.2784
Iris-setosa	-2.7103	0.0642	-0.3321	0.5686	0.0645
Iris-setosa	-2.8887	0.192	-0.7329	0.3144	0.0135
Iris-setosa	-2.7457	0.2535	0.5085	-0.4583	0.4209
Iris-setosa	-2.6997	-0.3032	-0.6618	-0.1095	0.339
Iris-setosa	-2.2694	-0.8398	0.1633	0.1344	0.4466
Iris-setosa	-2.84	0.0311	-0.1591	-0.0096	-0.2973

## Data Output

The PCA Apply operator applies the matrix transformation algorithm received from the PCA operator against the input data set, outputting the transformed principal component data set. The PCA Apply operator can therefore be followed directly by any operator that accepts an input data set.

## Example

The following example shows the PCA and PCA Apply operators together within a Hadoop workflow, with their output being passed into an Alpine Forest operator.



## Predictor (DB)

Applies an input regression, classification, or clustering model to an input data set in order to predict a value (or the highest probability value)



### Information at a Glance

Parameter	Description
Category	Predict
Data source type	DB
Send output to other operators	Yes
Data processing tool	MapReduce

The input column names must match the column names in the data set selected for model training, except for the dependent columns.

The prediction operation outputs its prediction columns with the columns of the input data set into a user-specified prediction table.

The operator includes the following prediction columns in the user-specified output table.

- **PRED\_<model\_abbreviation>** - the predicted value or value with highest probability
- **CONF\_<model\_abbreviation>** - the confidence in the predicted value
- **INFO\_<model\_abbreviation>** - a dictionary of information about the results

Model Type	Model	Column Abbreviation
Classification	• Naive Bayes	• NB

Model Type	Model	Column Abbreviation
	<ul style="list-style-type: none"> <li>Logistic Regression</li> <li>SVM</li> <li>Alpine Forest Classification</li> <li>Decision Tree</li> </ul>	<ul style="list-style-type: none"> <li>LOR</li> <li>SVM</li> <li>AFC</li> <li>DT</li> </ul>
Regression	<ul style="list-style-type: none"> <li>Linear Regression</li> <li>Alpine Forest Regression</li> </ul>	<ul style="list-style-type: none"> <li>LIR</li> <li>AFR</li> </ul>
Clustering	K-Means	KM  K-means output columns look a bit different.  The columns are: <ul style="list-style-type: none"> <li>PRED_KM - predicted cluster</li> <li>DIST_KM - distance to the center of the cluster</li> <li>INFO_KM - a dictionary of information about the results</li> </ul>

## Algorithm

The Predictor operator is used to predict the value of the dependent variable based on the model(s) generated from the input model operator(s).

Input Model	What Predictor Calculates
Classification algorithms	Value with the highest probability
Numeric regression algorithms	Predicted value
Clustering algorithms	Predicted cluster

## Input

An input regression, classification, or clustering model, and an input data set against which the model is applied.

## Configuration

<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Storage Parameters</b>	<p>Advanced database settings for the operator output. Available only for <b>TABLE</b> output.</p> <p>See <a href="#">Storage Parameters dialog</a> for more information.</p>
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

The data rows of the output table/view displayed (up to 2000 rows of the data).

## Predictor (HD)

Applies an input regression, classification, or clustering model to an input dataset in order to predict a value (or the highest probability value).



## Information at a Glance

Parameter	Description
Category	Predict
Data source type	HD
Send output to other operators	Yes
Data processing tool	MapReduce

The input column names must match the column names in the data set selected for model training, except for the dependent columns.

The prediction operation outputs its prediction columns with the columns of the input dataset into a user-specified prediction table.

The operator includes the following prediction columns in the user-specified output table.

- **PRED\_<model\_abbreviation>** - the predicted value or value with highest probability
- **CONF\_<model\_abbreviation>** - the confidence in the predicted value
- **INFO\_<model\_abbreviation>** - a dictionary of information about the results

Model Type	Model	Column Abbreviation
Classification	• Naive Bayes	• NB
	• Logistic Regression	• LOR
	• SVM	• SVM
		• AFC
	• Alpine Forest Classification	• DT

Model Type	Model	Column Abbreviation
	<ul style="list-style-type: none"> <li>Decision Tree</li> </ul>	
Regression	<ul style="list-style-type: none"> <li>Linear Regression</li> <li>Alpine Forest Regression</li> </ul>	<ul style="list-style-type: none"> <li>LIR</li> <li>AFR</li> </ul>
Clustering	K-Means	KM <ul style="list-style-type: none"> <li>PRED_KM - predicted cluster</li> <li>DIST_KM - distance to the center of the cluster</li> <li>INFO_KM - a dictionary of information about the results</li> </ul>

## Algorithm

The Predictor operator is used to predict the value of dependent variable based on the model(s) generated from the input model operator(s).

Input Model	What Predictor Calculates
Classification algorithms	Value with the highest probability
Numeric regression algorithms	Predicted value
Clustering algorithms	Predicted cluster

## Input

An input regression, classification, or clustering model, and an input dataset against which the model is applied.



## Configuration

<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Store Results?</b>	Specifies whether to store the results. <ul style="list-style-type: none"> <li>• <b>true</b> - results are stored.</li> <li>• <b>false</b> - the data set is passed to the next operator without storing.</li> </ul>
<b>Results Location</b>	The HDFS directory where the results of the operator are stored. This is the main directory, the subdirectory of which is specified in <b>Results Name</b> . Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.
<b>Results Name</b>	The name of the file in which to store the results.
<b>Overwrite</b>	Specifies whether to delete existing data at that path and file name. <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
<b>Compression</b>	Select the type of compression for the output.  Available Parquet compression options are the following. <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> Available Avro compression options are the following. <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>

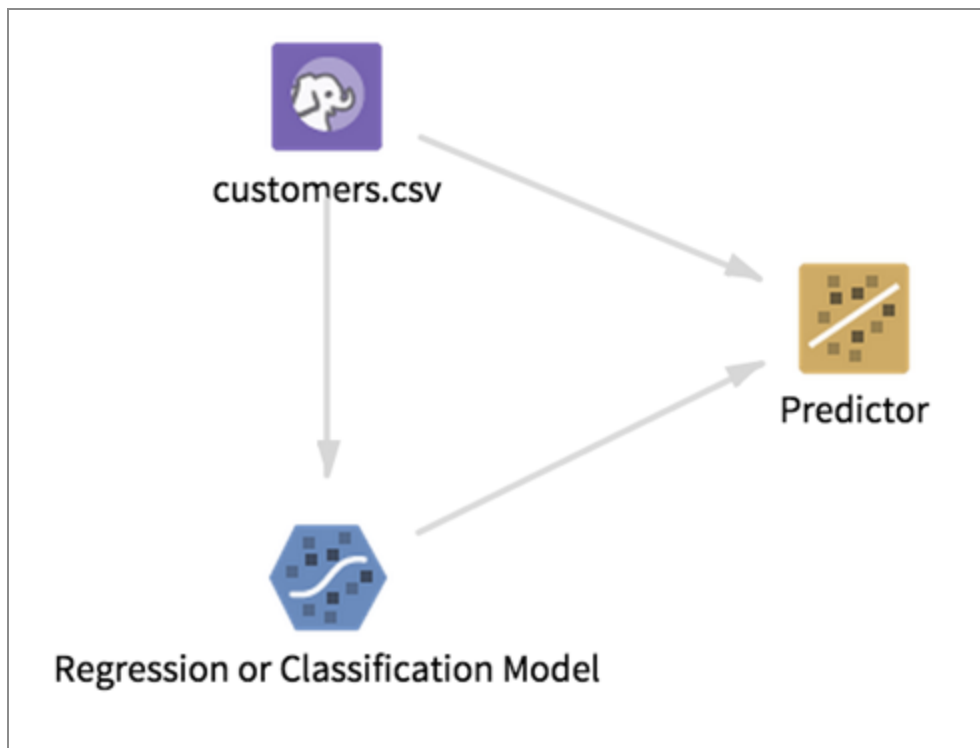
## Output

### Visual Output

The data rows of the output table or view displayed (up to 2000 rows of the data).

### Example

The following image shows an example input configuration.



## Model Validation Operators

The Model Validation operators (Scoring operators) allow the modeler to assess whether a model is statistically valid and how well it is performing.



**Note:** Find these operators in the **Predict** section of TIBCO Data Science - Team Studio.

## Alpine Forest Evaluator

Provides model accuracy data, a confusion matrix heat map that illustrates the classification model's accuracy for each possible predicted value, and an error convergence rate graph.



### Information at a Glance

Parameter	Description
Category	Model Validation
Data source type	HD
Send output to other operators	No
Data processing tool	MapReduce

The Alpine Forest Evaluator operator is an Alpine Forest model evaluation operator, similar to the Goodness of Fit evaluator, but more graphic in nature.



**Note:** The Alpine Forest Evaluator operator works for Alpine Forest Classification flows only.

### Algorithm

The Alpine Forest Evaluator operator is used to evaluate the accuracy of the predicted classifications of any Alpine Forest operator algorithm. Performance of the model is evaluated using the count of true positives, true negatives, false positives, and false

negatives in a matrix. The following table shows the confusion matrix for a two-class classifier.

		Predicted		<i>a</i> is the number of correct predictions that an instance is negative, <i>b</i> is the number of incorrect predictions that an instance is positive,
		Negative	Positive	
Actual	Negative	<i>a</i>	<i>b</i>	<i>c</i> is the number of incorrect predictions that an instance is negative, and <i>d</i> is the number of correct predictions that an instance is positive.
	Positive	<i>c</i>	<i>d</i>	

Like the Confusion Matrix operator, the Alpine Forest Evaluator operator also calculates several standard accuracy terms (See [Confusion Matrix](#) for more details).

- The accuracy (AC) is the proportion of the total number of predictions that were correct.
- The recall or true positive rate (TP) is the proportion of positive cases that were correctly identified.
- The false positive rate (FP) is the proportion of negative cases that were incorrectly classified as positive.
- The true negative rate (TN) is defined as the proportion of negative cases that were classified correctly.
- The false negative rate (FN) is the proportion of positive cases that were incorrectly classified as negative.
- Finally, precision (P) is the proportion of the predicted positive cases that were correct.

The Alpine Forest Evaluator also calculates an error rate in order to generate the error convergence graph. This indicates the misclassification rate of the model, which is simply 1 - accuracy (AC). This can also be computed as:

$$\text{Error Rate} = \frac{b+c}{a+b+c+d}$$

## Input

- The Alpine Forest Classification operator
- A data set from a preceding operator

## Restrictions

The Alpine Forest Evaluator operator must be connected to the Alpine Forest Classification operator and a data set from a preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.

## Output

### Visual Output

The Alpine Forest Evaluator operator produces classification accuracy data, a graphical heat map, and a graphical tracking of the number of decision trees vs. error rates.

### Classification Accuracy Data Output

The data output provides the classification accuracy counts for every Observed/Predicted combination for each class. In the following example, the intersection of the Observed (0) row and Predicted (0) column indicates that 47,714 predictions of value 0 were correct, while the Observed (0)/Predicted (1) cell indicates the model predicted 1 instead of 0 only 7 times. So for predicting the class of 0, the class recall was 99.98% correct.

However, the Observed (1)/Predicted (0) cell indicates 1,028 instances of the model incorrectly predicting 0 for actual values of 1 and the Observed (1)/Predicted (1) cell indicates the model predicted 1 correctly 1,251 times, giving an accuracy (class recall) for predicting 1 of 54.89%.

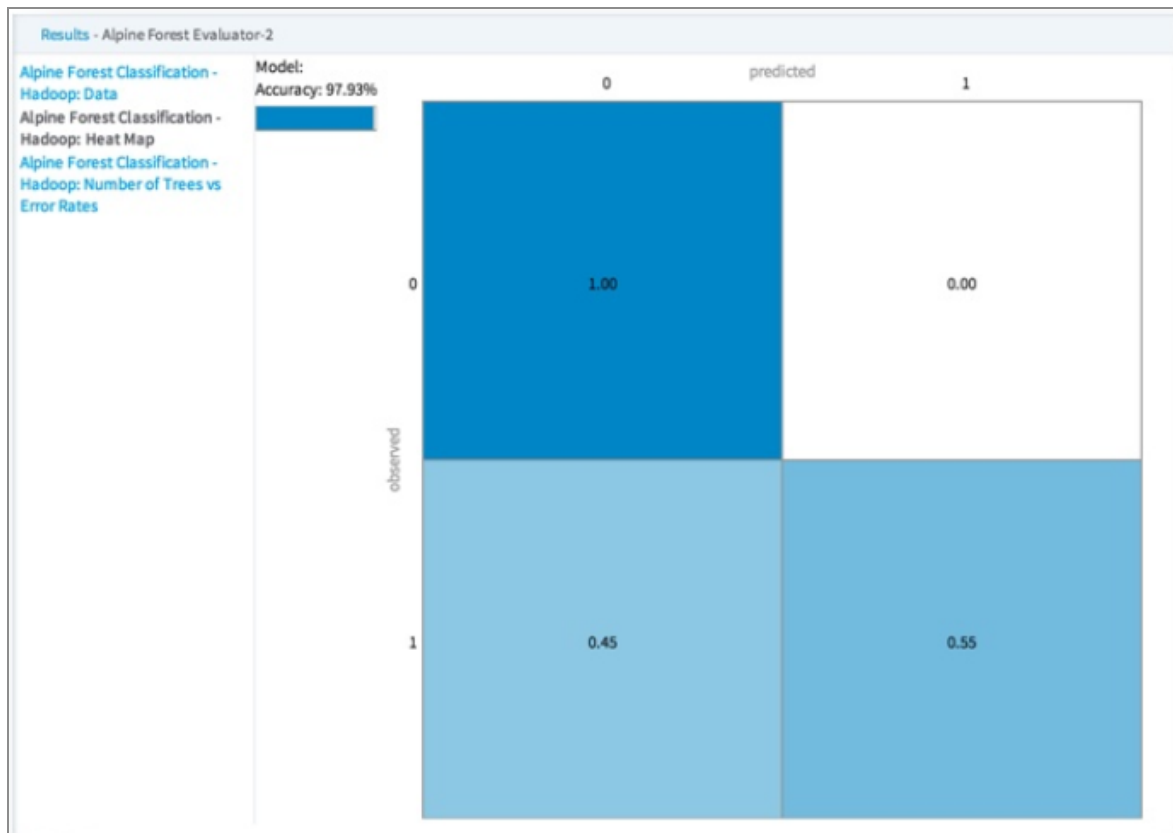
Overall, the Alpine Forest Classification model's accuracy, in this example, is calculated to be 97.93% accurate.

Results - Alpine Forest Evaluator-2			
Alpine Forest Classification - Hadoop: Data			
Alpine Forest Classification - Hadoop: Heat Map			
Alpine Forest Classification - Hadoop: Number of Trees vs Error Rates			
	Predicted (0)	Predicted (1)	Class Recall
Observed (0)	47714	7	0.9998533
Observed (1)	1028	1251	0.5489250
Class Precision	0.9789094	0.9944356	Accuracy: 0.9793000

## Heat Map

A Confusion Matrix heat map displays information about actual versus predicted counts of a classification model.

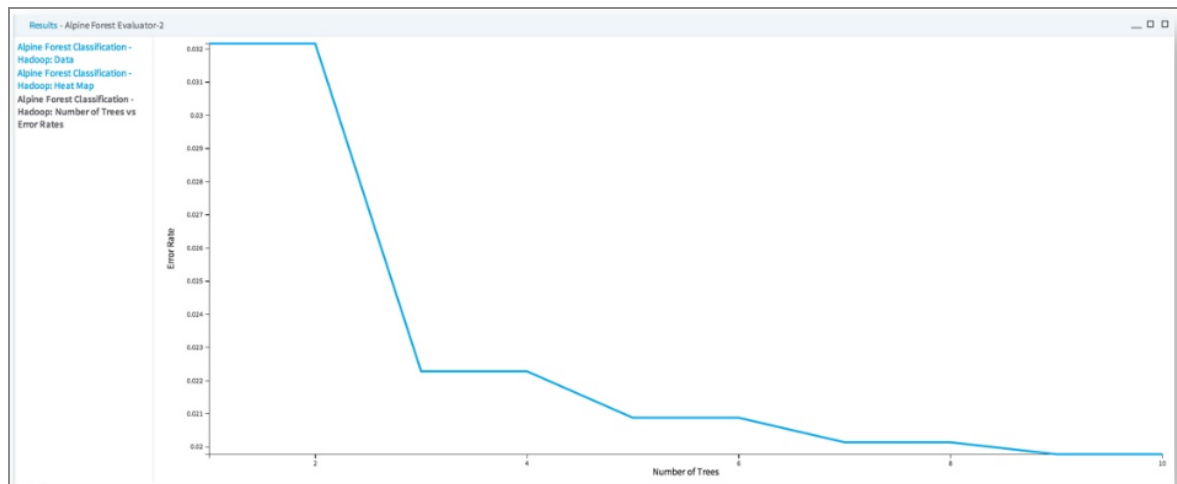
The following example shows a Confusion Matrix heat map for an Alpine Forest model. In this case, it is evident that the model performs best when predicting the value 0 with 100% accuracy. However, the accuracy drops for predicting the value 1, being correct only 49% of the time.



### Number of Trees vs. Error Rate Graph

Displays the error rate as the number of trees increases. This illustrates the Alpine Forest model's error convergence rate.

The following example illustrates how such a graph is useful for determining the number of decision trees needed in the Alpine Forest model in order for the error rate to be at its lowest. In this case, once there are nine decision trees, the error rate falls to 0.00%.



## Data Output

None. This is a terminal operator.

## Classification Threshold Metrics

Use to output (binary or multi-class) classification performance metrics for different confidence thresholds associated with a unique class that the user specifies.



## Information at a Glance

Parameter	Description
Category	Model Validation
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark



The Classification Threshold Metrics operator partially leverages Spark MLLib Classification Threshold Tuning in the [Evaluation Metrics package \(Spark version 1.5.1\)](#).

For more information about this operator and its available metrics, see [Prediction Threshold](#).

## Input

Classification Threshold Tuning must be preceded by either the Classifier operator or the Predictor operator. The outputs from those operators are required for the calculations in this operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	<p>Select the column that contains the dependent variable used to train the classification model (can be numeric or categorical).</p> <div> <p><b>Important:</b> If the distinct values in the <b>Dependent Column</b> selected are not a subset of the model classes (classes in sparse <b>Confidences</b> column), a <i>n</i> error occurs at runtime.</p> </div>
<b>Confidences Column</b>	<p>Select the column that contains the confidences levels associated with classes from the classification model (binary or multi-class). This column must have the sparse data type, and contains the dictionary (with string keys and double values) of all confidence levels associated with model classes.</p> <p>Example: {"red":0.52, "green":0.32, "blue":0.26} or {"0":0.52, "1":0.48}</p> <p>It is most likely to be the <b>INFO_model_name</b> column in the output of the Classifier or Predictor operators.</p>
<b>Class to Predict</b>	Enter one of the model classes to predict (quotes are not needed for both

Parameter	Description
	<p>numeric or string entries). This class is considered as the positive class in order to compute the classification metrics.</p> <p>Example: red or 1</p> <p><b>Note:</b> If the <b>Dependent</b> column is numeric and the value entered for <b>Class to Predict</b> cannot be cast to numeric, an error appears before closing the parameter dialog.</p> <p>If the value entered for <b>Class to Predict</b> is not a member of the model classes (classes in <b>Confidences</b> column), an error occurs at runtime.</p>
<b>Number of Bins (approx.)</b>	<p>Select the approximate number of confidence threshold bins (default is 20), corresponding to the approximate number of rows in the output.</p> <ul style="list-style-type: none"> <li>• If 0 is entered, the result contains a point for each distinct confidence threshold in the input, and this could be as large as the input itself.</li> <li>• Otherwise, the result is approximately made of <math>X</math> bins after down-sampling.</li> </ul> <p>Points are made of bins of equal number of consecutive points. The size of each bin is equal to <math>\text{floor}(\text{total\_rows}/\text{num\_bins})</math>, which means the resulting number of bins might not exactly match the value specified. The last bin in each partition might be smaller as a result, meaning there might be an extra sample at partition boundaries.</p>
<b>Beta Value for F-measure ( <math>\beta</math> )</b>	<p>Enter the <math>\beta</math> value to compute F-score (must be <math>\geq 0</math>, default = 1).</p> <p><b>Note:</b> The beta parameter determines the weight of precision in the combined score F-measure.</p> $F_{\beta} = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$ <p>As shown in the formula above, <math>\beta &lt; 1</math> lends more weight to precision, while <math>\beta &gt; 1</math> favors recall. If <math>\beta = 1</math>, the F1-measure is called the harmonic mean of recall and precision.</p>

Parameter	Description
<b>Write Rows Removed Due to Null Data To File</b>	<p>Rows with at least one null value in either the <b>Dependent</b> column or <b>Confidences</b> column are removed from the analysis. This parameter allows you to specify whether rows with null values are written to a file.</p> <p>The file is written to the same directory as the rest of the output. The file name is suffixed with <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.

Parameter	Description
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

confidence_threshold	cumulative_count	cumulative_count_percent	accuracy	recall	precision	f1_score	false_positive_rate	AUC
0.17	501	0.09	0.93	0.92	0.29	0.40	0.0	0.28
0.13	1003	0.1	0.9	0.94	0.24	0.37	0.09	0.46
0.092	1505	0.15	0.86	0.87	0.2	0.40	0.13	0.54
0.044	2006	0.2	0.82	0.74	0.17	0.79	0.27	0.57
0.034	2508	0.25	0.78	0.79	0.14	0.18	0.33	0.57
0.026	3006	0.3	0.73	0.63	0.13	0.77	0.35	0.56
0.024	3507	0.35	0.68	0.67	0.11	0.49	0.2	0.54

### Data Output

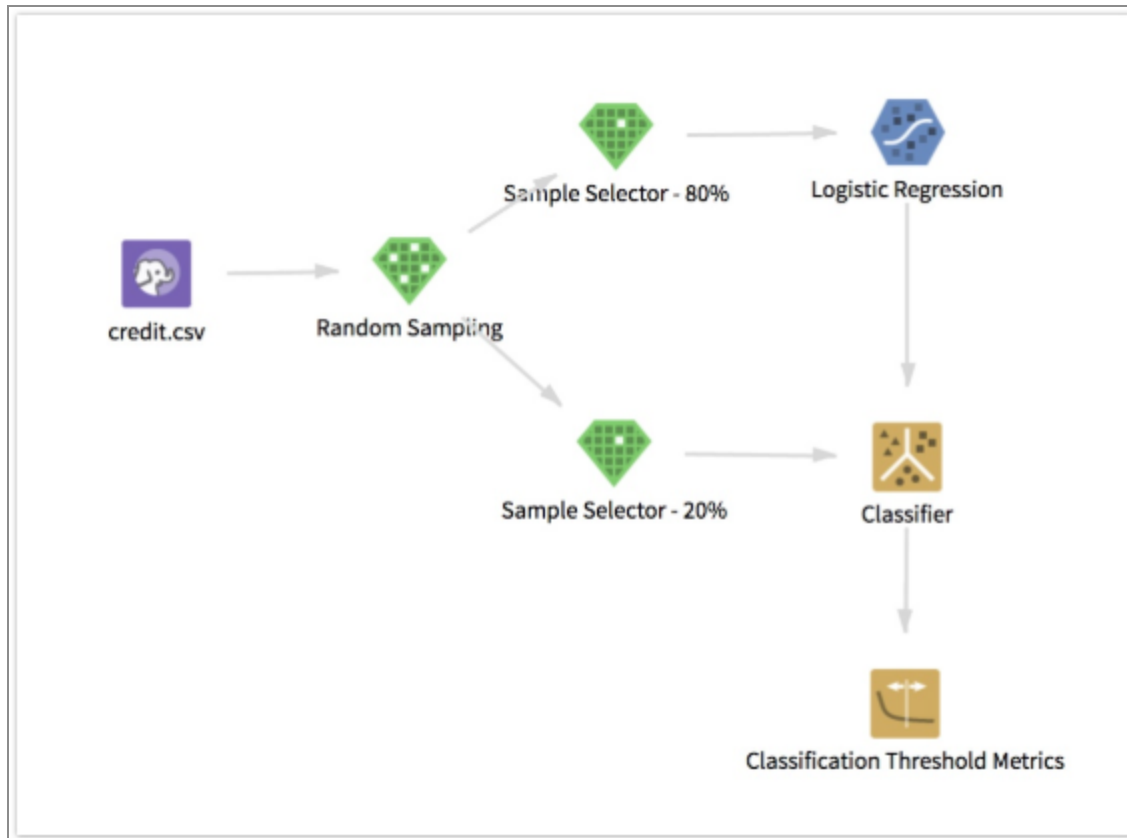
A data set that contains the associated confidence thresholds (in descending order) and performance metrics, which can be connected to subsequent operators.

The following metrics are available in the output (for the positive class selected).

- Recall
- Precision
- F-score (for beta value  $\beta$  chosen)
- False positive rate
- Cumulative Count

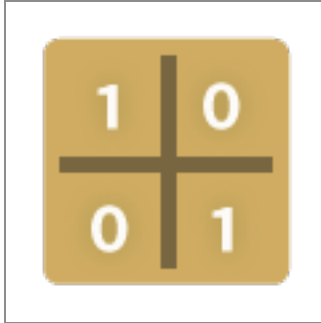
- Cumulative Count (percentage)
- Accuracy
- Lift
- KS

## Example



## Confusion Matrix

Displays information about actual versus predicted counts of a classification model and helps assess the model's accuracy for each of the possible class values.



## Information at a Glance

Parameter	Description
Category	Model Validation
Data source type	HD, DB
Send output to other operators	No
Data processing tool	MapReduce

The Confusion Matrix Operator is a classification model evaluation operator similar to the Goodness of Fit evaluator, but it is more graphic in nature.



**Note:** All Hadoop and database classification workflows can be evaluated using this operator.

## Algorithm

The Confusion Matrix operator is used to evaluate the accuracy of the predicted classifications of any TIBCO Data Science - Team Studio Classification Modeling algorithm, including the results of the Logistic Regression, Alpine Forest, Naive Bayes, Decision Tree, or SVM Operators.

The model performance is evaluated using the count of true positives, true negatives, false positives, and false negatives in a matrix. The following table shows the confusion matrix for a two-class classifier:

		<b>Predicted</b>		a is the number of correct predictions that an instance is negative, b is the number of incorrect predictions that an instance is positive,
		Negative	Positive	
<b>Actual</b>	Negative	a	b	c is the number of incorrect predictions that an instance is negative, and d is the number of correct predictions that an instance is positive.
	Positive	c	d	

In the case of a 2-class classification model (for example), the Confusion Matrix operator calculates several standard accuracy terms.

#	Accuracy term	Description	Equation
1	<b>Accuracy (AC)</b>	The accuracy (AC) is the proportion of the total number of predictions that were correct.	$AC = \frac{a+d}{a+b+c+d}$
2	<b>Recall or true positive rate (TP)</b>	The recall or true positive rate (TP) is the proportion of positive cases that were correctly identified.	$TP = \frac{d}{c+d}$
3	<b>False positive rate (FP)</b>	The false positive rate (FP) is the proportion of negative cases that were incorrectly classified as positive.	$FP = \frac{b}{a+b}$
4	<b>True negative rate (TN)</b>	The true negative rate (TN) is the proportion of negative cases that were classified correctly.	$TN = \frac{a}{a+b}$
5	<b>False negative rate (FN)</b>	The false negative rate (FN) is the proportion of positive cases that were incorrectly classified as negative.	$FN = \frac{c}{c+d}$
6	<b>Precision (P)</b>	Precision (P) is the proportion of the predicted positive cases that were correct.	$P = \frac{d}{b+d}$

The accuracy determined using equation 1 might not be an adequate performance measure when the number of negative cases is much greater than the number of positive cases (Kubat et al., 1998). Suppose there are 1000 cases, 995 of which are negative and 5 of

which are positive. If the system classifies them all as negative, the accuracy would be 99.5%, even though the classifier missed all positive cases.

Other performance measures account for this by including TP in a product: for example, geometric mean (g-mean) (Kubat et al., 1998), defined in equations 7 and 8, and F-Measure (Lewis and Gale, 1994), defined in equation 9.

#	Measure	Description	Equation
7	geometric mean (g-mean), 1	Geometric mean of True positive rate and Precision	$g-mean_1 = \sqrt{TP \times P}$
8	geometric mean (g-mean), 2	Geometric mean of True positive rate and True negative rate	$g-mean_2 = \sqrt{TP \times TN}$
9	F-Measure	The harmonic mean of Precision and False positive rate	$F = \frac{(\beta^2 + 1) \times P \times TP}{\beta^2 \times P + TP}$

In equation 9,  $\beta$  has a value from 0 to infinity and is used to control the weight assigned to TP and P. Any classifier evaluated using equations 7, 8, or 9 has a measure value of 0, if all positive cases are classified incorrectly.

## Input

- A data set from a preceding operator.
- Model(s) from preceding operators. When more than one model is received from its preceding operators, the result can be used for model comparison. This input is optional on database.

## Bad or Missing Values

Predictions are made only for rows that contain data. Rows with missing data are skipped.

## Configuration

Confusion Matrix offers two possible configurations.

- Connect a classification model operator and a data set. In this configuration, the model scores the samples in the data set, and the Confusion Matrix summarizes the



results. For Hadoop, this configuration is the only option, where the operator requires no configuration. For a database, for this configuration, you must set the `Use Model` parameter to **true**. With both a model and a data set connected, no parameters are necessary.

- The second configuration for database uses just an input table, with the prediction columns already present. In this case, set `Use Model` to **false**, and select the prediction columns to evaluate using the **Prediction Columns** input. With just a data set connect (on database) the following parameters apply.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	Define the column used as the dependent variable in the model.
<b>Use Model</b>	<p>Specify whether the evaluation uses a model from preceding operator(s) or the data in the prediction columns of the input data set.</p> <ul style="list-style-type: none"> <li>• If <b>true</b> (the default), at least one model operator must directly precede it.</li> <li>• If <b>false</b>, the prediction columns must be present in the input data set from its preceding operator.</li> </ul> <p><b>Note:</b> This parameter does not apply to Hadoop data sources.</p>
<b>Prediction Columns</b>	<p>Choose the list of columns in the input data set to compare to the dependent column.</p> <p><b>Note:</b> This parameter does not apply to Hadoop data sources.</p>

## Output

### Visual Output

The Confusion Matrix produces both classification accuracy data and a graphical heat map.

## Classification Accuracy Data Table

The data output provides the classification accuracy counts for every Observed / Predicted combination for each class.

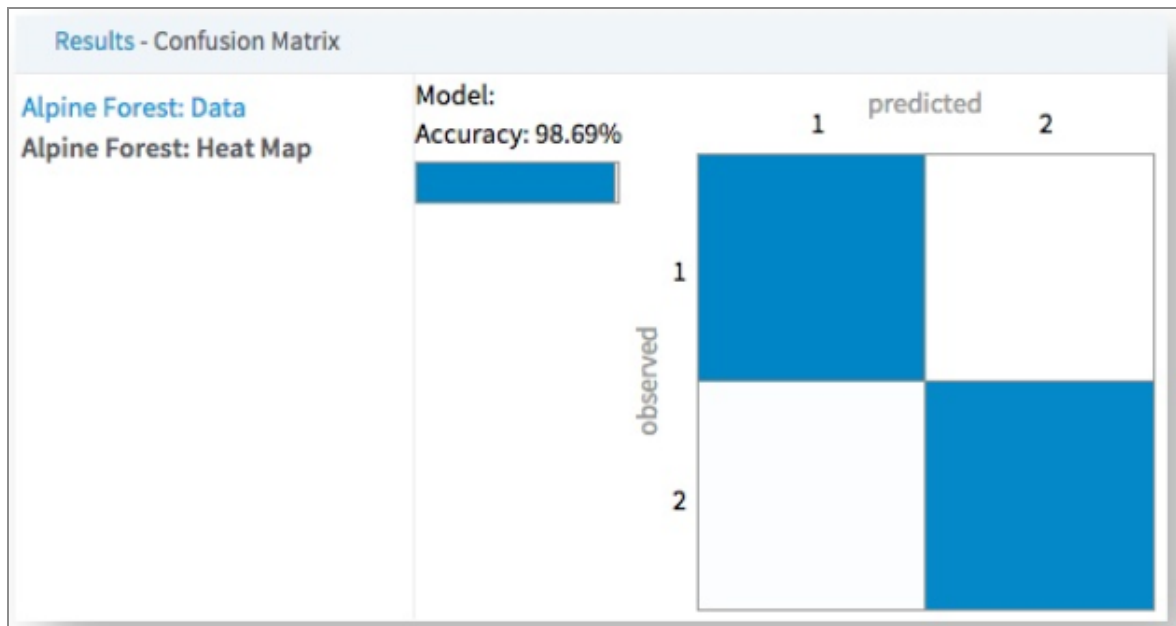
In the following example, the intersection of the Observed (1) row and Predicted(1) column indicates that 111,309 predictions of value 1 were correct, while the Observed (1)/ Predicted(2) cell indicates the model predicted 2 instead of 1 426 times. So for predicting the class of 1, the class recall was 99.62% correct. However, the Observed (2)/ Predicted(1) cell indicates 2,311 instances of the model incorrectly predicting 1 for actual values of 2 and the Observed(2)/ Predicted(2).

Results - Confusion Matrix				
Alpine Forest: Data Alpine Forest: Heat Map		Predicted (1)	Predicted (2)	Class Recall
	Observed (1)	111309	426	0.9962
	Observed (2)	2311	95647	0.9764
	Class Precision	0.9797	0.9956	Accuracy: 0.9869

## Heat Map

A Confusion Matrix Heat Map displays information about actual vs. predicted counts of a classification model.

The following example shows a Confusion Matrix Heat Map for a Logistic Regression model. In this case, it is evident that the model performs the best when predicting the value 0 with 99% accuracy. However, the accuracy drops for predicting the value 1, being correct only 10% of the time.

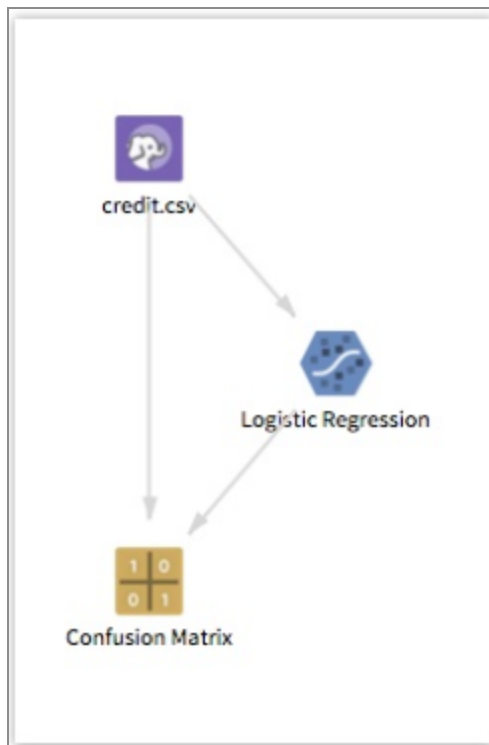


### Data Output

None. This is a terminal operator.

### Example

This example illustrates a Logistic Regression Operator and the associated data set, `credit.csv`, sending output to the Confusion Matrix.



## Goodness of Fit

Verifies a trained model.



## Information at a Glance

Parameter	Description
Category	Model Validation

Parameter	Description
Data source type	DB, HD
Send output to other operators	No
Data processing tool	MapReduce

This operator applies the trained model on the input data set, and then calculates the following for the model.

- Precision
- Recall
- F1
- Sensitivity
- Specificity
- Accuracy

It applies in general to classification models.

## Algorithm

When both columns being compared are categorical (or levels), the natural summary is a contingency table - a simple matrix of counts of how often each combination of categories or levels is seen. For example, a result could look like the following illustration.

		Facts		
		Positive	Negative	
Predic- tion	Positive	True Positive	False Positive (Type I error, P-value)	→ Positive predictive value
	Negative	False Negative (Type II error)	True Negative	→ Negative predictive value
		↓ Sensitivity	↓ Specificity	

From this table, all of the traditional Goodness of Fit statistics (Precision, Recall, F1, Sensitivity, Specificity, and Accuracy) can immediately be read off, as shown in the following table.

Score	Formula
Precision	$TP/(TP + FP)$
Recall	$TP/(TP + FN)$
F1	$2 (Precision \times Recall)/(Precision + Recall)$
Sensitivity	$TP/(TP + FN)$
Specificity	$TN/(FP + TN)$
Accuracy	$(TP + TN)/(TP + FP + TN + FN)$

## Input

1. A data set from a preceding operator.
2. Model(s) from preceding operators. If more than one model is received from its preceding operators, the result can be used for model comparison. This input is optional on a database data source.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	The column to use as the dependent variable in the model.
<b>Use Model</b>	(Not present on Hadoop) Specify whether the evaluation uses a model from a

Parameter	Description
	preceding operator(s) or the data in the prediction columns of the input data set.
	If <b>true</b> (the default), at least one model operator must directly precede it. If <b>false</b> , the prediction columns must be present in the input data set from its preceding operator.
<b>Prediction Columns</b>	(Not present on Hadoop) Choose the list of columns in the input data set to compare to the dependent column.

## Output

### Visual Output

A Goodness of Fit result table that shows the primary Goodness of Fit statistics of each model.

Node	Accuracy	Error	Stats	Recall	Precision	F1	Specificity	Sensitivity
Logistic Regression	0.8174	0.1826	1	0.8462	0.8333	0.8397	0.78	0.8462
Logistic Regression	0.8174	0.1826	-1	0.78	0.7959	0.7879	0.8462	0.78

### Data Output

None. This is a terminal operator.

## Lift (DB)

Visualizes the performance of a classification model. It applies in general to classification models (for example, CART, Decision Tree, Logistic Regression, Naive Bayes, Neural Network, and Alpine Forest).



## Information at a Glance

Parameter	Description
Category	Model Validation
Data source type	DB
Send output to other operators	No
Data processing tool	n/a

While a cumulative gains chart shows the total number of events captured by a model over a given number of samples, a lift curve shows the ratio of a model to a random guess. Lift charts show how a model performs compared to random guessing given  $x$  number of samples.

For example, suppose a population has an average response rate of 1%, but a certain model has identified a segment with a response rate of 10%. That segment has a "lift" of 10.0 (10%/1%).

By ranking quantiles of data by lift, you can see which areas have the most lift and thus which the model performs best on. For more information and examples about lift, see [here](#).

## Input

- A data set from the preceding operator.
- Optional - One or more model(s) from the preceding operator(s). The models must be a classification model. The following models are supported.
  - CART
  - Decision Tree
  - Logistic Regression
  - Naive Bayes
  - Neural Network
  - Alpine Forest



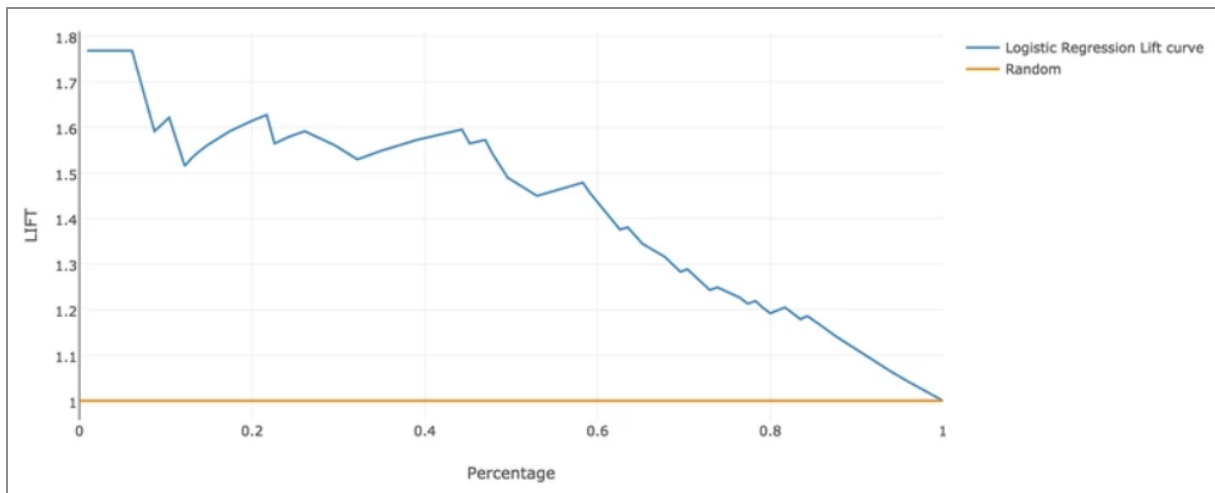
## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	Define the column to use as the class variable.
<b>Value to Predict</b>	<p>The value that represents the event to analyze.</p> <div> <p><b>Note:</b> The value of this column must match the data as it is stored in the database that matches how it appears in the data explorer. If the user defines a Boolean <b>Dependent Column</b> with 1s and 0s, the user must use 1 or 0 as the <b>Value to Predict</b>. If the column uses True and False, the user must use "True" or "False" as the <b>Value to Predict</b>.</p> </div>
<b>Use Model</b>	<p>Specify whether the evaluation uses a model from preceding operator(s) or the data in the prediction columns of the input data set. If <b>true</b>, at least one model operator must directly precede it. If <b>false</b>, the prediction columns must be present in the input data set from its preceding operator.</p> <p>Default value: <b>true</b>.</p>
<b>Confidence Columns</b>	Choose the list of columns in the input data set to compare to the dependent column.

## Output

### Visual Output

Lift diagram. Used for model comparison if more than one model is supplied by the preceding operators.



### Data Output

None. This is a terminal operator.

## Lift (HD)

Visualizes the performance of a classification model. It applies in general to classification models (for example, CART, Decision Tree, Logistic Regression, Naive Bayes, Neural Network, and Alpine Forest).



### Information at a Glance

Parameter	Description
Category	Model Validation

Parameter	Description
Data source type	HD
Send output to other operators	No
Data processing tool	MapReduce

While a cumulative gains chart shows the total number of events captured by a model over a given number of samples, a lift curve shows the ratio of a model to a random guess. Lift charts show how a model performs compared to random guessing given  $x$  number of samples.

For example, suppose a population has an average response rate of 1%, but a certain model has identified a segment with a response rate of 10%. That segment has a "lift" of 10.0 (10%/1%).

By ranking quantiles of data by lift, you can see which areas have the most lift and thus which the model performs best on. For more information and examples about lift, see [here](#).

## Input

- A data set from the preceding operator.
- One or more model(s) from the preceding operator(s). The models must be a classification model. The following models are supported.
  - CART
  - Decision Tree
  - Logistic Regression
  - Naive Bayes
  - Neural Network
  - Alpine Forest

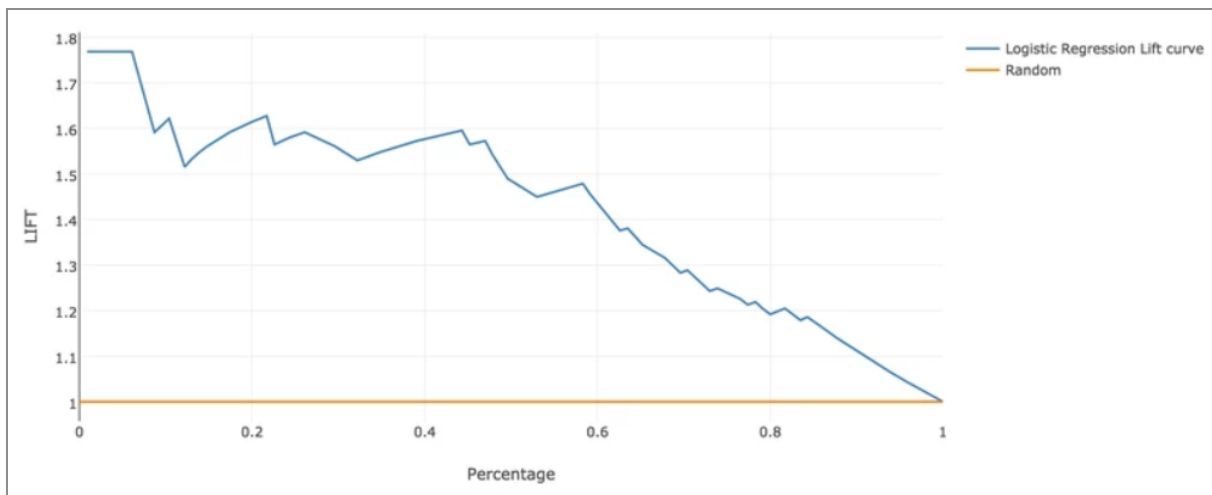
## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	Define the column to use as the class variable.

## Output

### Visual Output

Lift diagram. Used for model comparison if more than one model is supplied by the preceding operators.



### Data Output

None. This is a terminal operator.

## Regression Evaluator (DB)

Computes several commonly used statistical tests to determine the accuracy of several columns (Predicted Values). These represent predictions against one column (Actual Value), which is specified as the "ground truth."



## Information at a Glance

Parameter	Description
Category	Model Validation
Data source type	DB
Send output to other operators	No
Data processing tool	n/a

**i Note:** The Regression Evaluator (DB) operator is for database data only. For Hadoop data, use the [Regression Evaluator \(HD\)](#) operator.

For information about the metrics used in this operator, see [Computed Metrics and Use Case for the Regression Evaluator](#).

## Input

A tabular data set from a database that contains a numeric column of actual values (known truth) and numeric column(s) of predicted values.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Actual Value</b>	A numeric column that holds the dependent variable that the models were used to train on, or a column of known values for the dependent variable.
<b>Predicted Values (to Compare with Actual Value)</b>	A set of numeric column(s) whose results predict the model. For example, if you are using this to evaluate several different linear regressions, the predicted values for each of the regressions is selected here.
<b>Output Type</b>	<ul style="list-style-type: none"> <li>• <b>TABLE</b> outputs a database table. Specifying <b>TABLE</b> enables <b>Storage Parameters</b>.</li> <li>• <b>VIEW</b> outputs a database view.</li> </ul>
<b>Output Schema</b>	The schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

A table of metrics about each of the predicted columns.

Results - Regression Evaluator (Dataset)					
Output Bad Data Report	Model	Mean Absolute Error	Mean Squared Error	R Squared	Root Mean Squared Error
	PRED_LIR	8.0108	103.5005	-0.2211	10.1735
	PRED_LIR_1	7.9117	77.6638	0.0837	8.8127
	PRED_LIR_2	8.6298	84.7545	0	9.2062

Data Output

None. This is a terminal operator.

Regression Evaluator (HD)

Computes several commonly used statistical tests to determine the accuracy of several columns (Predicted Values). These represent predictions against one column (Actual Value) which is specified as the "ground truth."



Information at a Glance

Parameter	Description
Category	Model Validation
Data source type	HD
Send output to other operators	No
Data processing tool	Spark

**i Note:** The Regression Evaluator (HD) operator is for Hadoop data only. For database data, use the [Regression Evaluator \(DB\)](#) operator.

For information about the metrics used in this operator, see [Computed Metrics and Use Case for the Regression Evaluator](#).

## Input

A tabular data set from Hadoop that contains a numeric column of actual values (known truth) and numeric column(s) of predicted values.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Actual Value</b>	A numeric column that holds the dependent variable that the models were used to train on, or a column of known values for the dependent variable.
<b>Predicted Values (to Compare with Actual Value)</b>	A set of numeric column(s) whose results predict the model. For example, if you are using this to evaluate several different linear regressions, the predicted values for each of the regressions is selected here.
<b>Write Rows Removed Due To Null Data To File</b>	<p>Rows with null values are removed from the analysis. This allows you to specify that the data with null values be written to a file.</p> <p>The file is written to the same directory as the rest of the output. The filename is suffixed with <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI.</li> </ul>



Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>
<b>Advanced Spark Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>

## Output

### Visual Output

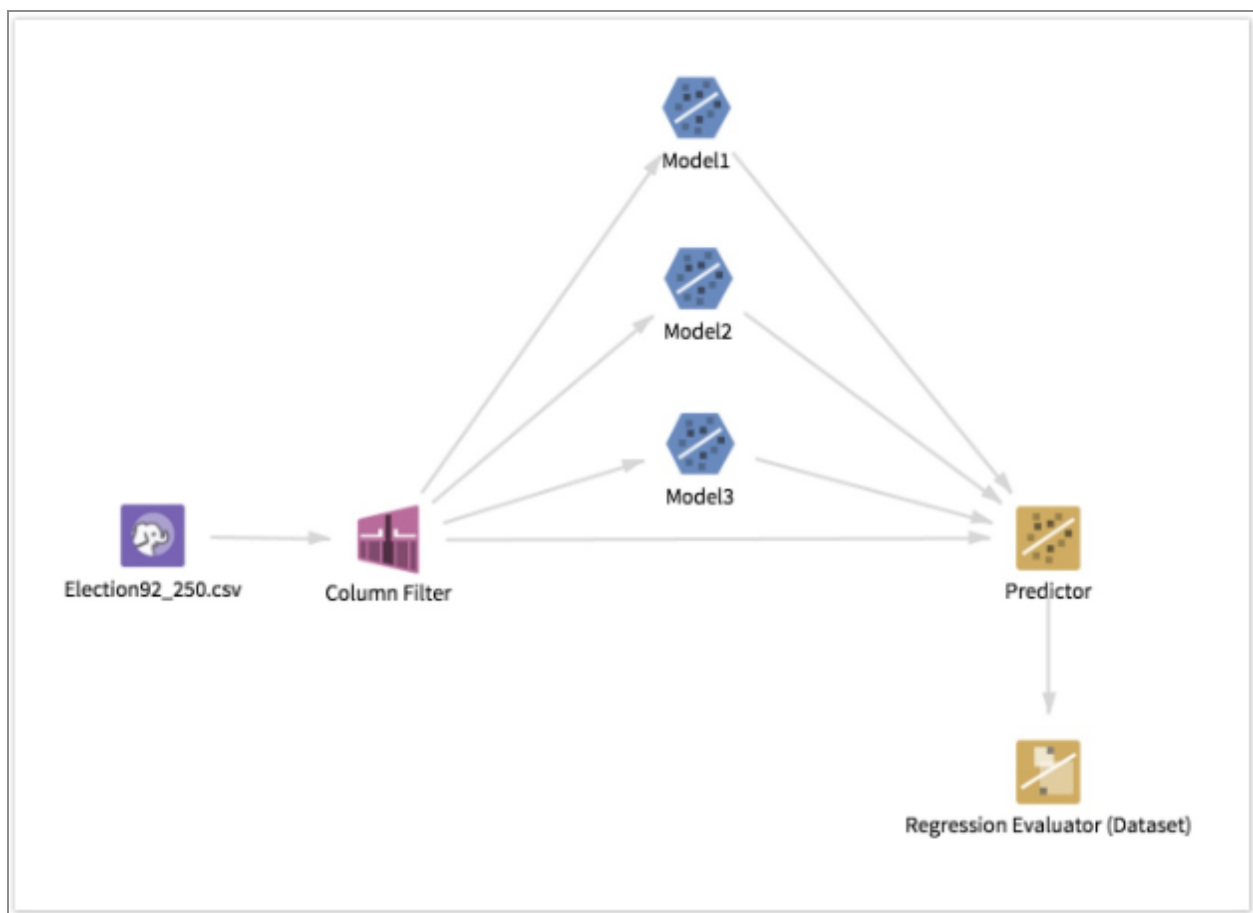
A table of metrics about each of the predicted columns.

Results - Regression Evaluator (Dataset)					
Output <a href="#">Bad Data Report</a>	Model	Mean Absolute Error	Mean Squared Error	R Squared	Root Mean Squared Error
	PRED_LIR	8.0108	103.5005	-0.2211	10.1735
	PRED_LIR_1	7.9117	77.6638	0.0837	8.8127
	PRED_LIR_2	8.6298	84.7545	0	9.2062

### Data Output to Succeeding Operators

None. This is a terminal operator.

## Example



## ROC

Generates a Receiver Operating Characteristic (ROC), or ROC curve.



### Information at a Glance

Parameter	Description
Category	Model Validation
Data source type	HD, DB
Send output to other operators	No
Data processing tool	MapReduce

The ROC curve is used to verify and compare the trained model(s) passed from a preceding model operator or operators by applying the algorithm on the data set passed from a preceding operator. The ROC-AUC method considers the coordinate pairing of the false positive rate (FP) and the true positive rate (TP). This set of coordinates forms the Receiver Operating Characteristic (ROC) curve.

The value of the ROC curve can be summarized by calculating the Area Under the ROC curve (AUC).

A random model typically has an ROC curve running along the diagonal. A better model curves to the upper left-hand side, thus having an AUC value approaching one.

This operator can be applied, in general, to any classification model (for example, CART, Decision Tree, Logistic Regression, Naive Bayes, Neural Network and Alpine Forest Classification).

## Input

- A data set from the preceding operator.
- One or more model(s) from the preceding operator(s). This input is optional on database.

## Configuration

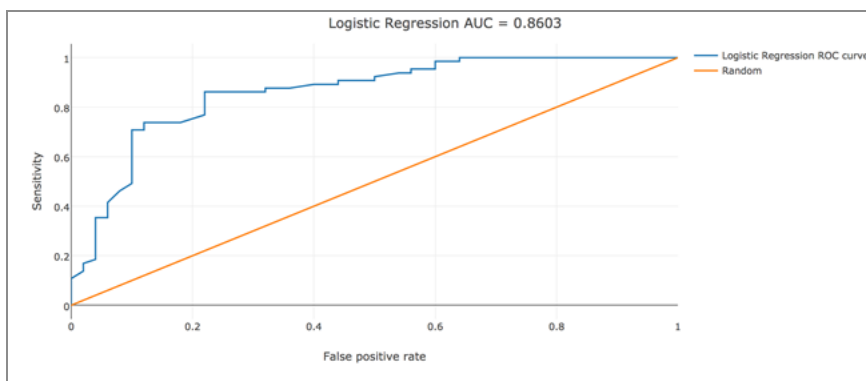
Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Column</b>	Define the column used as the class variable. (Not present in Hadoop)
<b>Value to Predict</b>	<p>The value represents the event to analyze.</p> <div> <p><b>Note:</b> The value of this column must match the data as it is stored in the database, which matches how it is displayed in the data explorer. For example, consider a column that contains Boolean values.</p> <ul style="list-style-type: none"> <li>• If the <b>Dependent Column</b> represents the Boolean values as 1 and 0, then for the <b>Value to Predict</b>, the user must also use 1 or 0.</li> <li>• If the <b>Dependent Column</b> represents the Boolean values as True and False, then for the <b>Value to Predict</b>, the user must also use True or False.</li> </ul> </div>
<b>Use Model</b>	<p>Specifies whether the evaluation should use a model from its preceding operator(s), or if it should use the data in the prediction columns of the input data set.</p> <ul style="list-style-type: none"> <li>• If <b>true</b> (the default), at least one model operator must directly precede it.</li> <li>• If <b>false</b>, the prediction columns must be present in the input data set from its preceding operator.</li> </ul> <p>(Not present in Hadoop)</p>

Parameter	Description
<b>Confidence Columns</b>	Specifies the list of columns in the input data set to compare to the <b>Dependent Column</b> . (Not present in Hadoop)

## Output

### Visual Output

A ROC-AUC diagram.



### Date Output

None. This is a terminal operator.

## T-Test - Independent Samples

Computes a test of statistical significance against a student's t-distribution for one measure across two different groups.



## Information at a Glance

Parameter	Description
Category	Model Validation
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

The independent samples t-test is used to test whether two groups are significantly different for the same measure.

For information about Student's t-distribution, see [https://en.wikipedia.org/wiki/Student%27s\\_t-distribution](https://en.wikipedia.org/wiki/Student%27s_t-distribution)

## Algorithm

The means and variances for all of the test statistics are computed using Spark's [MultivariateStatisticalSummary](#) object, but the t-tests themselves are computed from Java's [commons-math](#) library.

## Input

A tabular data set with numeric columns.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Values</b>	A numeric column that contains the sample values.

Parameter	Description
<b>Columns to Test</b>	Specify the columns to test.
<b>Column To Group By</b>	A categorical column used to separate the two samples.
<b>First Group Value (appears in group by column)</b>	<p>The name of the group for the first sample. This must be a value in the <b>Column to Group By</b> list.</p> <p><b>Note:</b> For example, to split out people who voted for a particular candidate versus those who did not, and assuming the values in the column were "yes" and "no", you could specify "yes" as the first group value and "no" as the second group value.</p>
<b>Second Group Value (appears in group by column)</b>	The name of the group for the second sample. This must be a value in the <b>Column to Group By</b> list.
<b>Sample Means Have Equal Variance (Homoscedastic T-Test)</b>	Specify whether to use a homoscedastic t-test ( <b>No</b> or <b>Yes</b> ).
<b>Write Rows Removed Due to Null Data To File</b>	<p>Rows with null values are removed from the analysis. This parameter allows you to specify that the data with null values be written to a file.</p> <p>The file is written to the same directory as the rest of the output. The filename is suffixed with <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI.</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Advanced Spark</b>	<ul style="list-style-type: none"> <li>• <b>Yes</b> specifies using the default Spark optimization settings.</li> </ul>

Parameter	Description
<b>Settings Automatic Optimization</b>	<ul style="list-style-type: none"> <li>• <b>No</b> enables providing customized Spark optimization. Click <b>Edit Settings</b> to customize Spark optimization. See <a href="#">Advanced Settings dialog</a> for more information.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>

## Output

### Visual Output



See [Independent Samples T-Test Use Case](#) for example data on a puppy training program that illustrates use of the independent samples t-test. The results contain one row for each column selected in the **Sample Columns** parameter. In this case we see that the puppies that Jenny trained do not perform significantly differently than those that Rachel trained either before or after training, because none of the p-values are close to zero.

Results - T-Test, Independent Samples-1						
<b>Output</b> <a href="#">Bad Data Report</a> <a href="#">Sample Summary Stats</a>	TTest Type	Column Tested	T Statistic	Two Tailed PValue	Lower One Tailed PValue	Upper One Tailed PValue
	Independent Samples T-Test	Score_Before_Training	-0.8966	0.3754	0.1877	0.8123
	Independent Samples T-Test	Score_After_Training	-0.0909	0.928	0.464	0.536

## Data Output

- **T Statistic** - A value computed based on the average and variance. The higher the magnitude of the t-statistic, the higher the difference between the means.
- **Two Tailed PValue** - The sum of the area under the Student's t-distribution above the absolute value of the t-statistic and below the inverse of the t-statistic. A higher value indicates a greater absolute difference in the sample compared. The null hypothesis is usually rejected if  $p < 0.05$ .
- **Lower One Tailed PValue** - The area under the Student's t-distribution between negative infinity and the t statistic. A lower p-value indicates that sample a is less than sample b. The null hypothesis is usually rejected if  $p < 0.05$ .
- **Upper One Tailed PValue** - The area under the Student's t-distribution between positive infinity and the t statistic. A lower p-value indicates that sample a is greater than sample b. The null hypothesis is usually rejected if  $p < 0.05$ .

## T-Test - Paired Samples

Computes a test of statistical significance for two measures of the same data points. This is the same as computing a single sample t-test against the difference between the two columns and a known mean of zero.



## Information at a Glance

Parameter	Description
Category	Model Validation
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

The paired samples t-test is used to test whether two responses measured on the same statistical unit are significantly different. Mathematically, it is the same as running a single sample t-test on the delta of the two samples for each row against an assumed mean of 0.0.

For information about Student's t-distribution, see [https://en.wikipedia.org/wiki/Student%27s\\_t-distribution](https://en.wikipedia.org/wiki/Student%27s_t-distribution)

## Algorithm

The means and variances for all of the test statistics are computed using Spark's [MultivariateStatisticalSummary](#) object, but the t-tests themselves are computed from Java's [commons-math](#) library.

## Input

A tabular data set with numeric columns.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>First Column</b>	The first measure to compute the t-test on. This must be a numeric column.
<b>Second Column</b>	The second measure to compute the t- test on. This must be a numeric column.
<b>Write Rows Removed Due to Null Data To File</b>	<p>Rows with null values are removed from the analysis. This parameter allows you to specify that the data with null values be written to a file. The file is written to the same directory as the rest of the output. The filename is suffixed with <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows (Fastest)</b> - remove null value data but do not count and display in the result UI.</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all removed rows to an external file.</li> </ul>
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li><b>Deflate</b></li> <li><b>Snappy</b></li> <li>no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li><b>Yes</b> - if the path exists, delete that file and save the results.</li> <li><b>No</b> - fail if the path already exists.</li> </ul>

## Output

### Visual Output

See [Paired Samples T-Test Use Case](#) for example data on a puppy training program that illustrates use of the paired samples t-test. In this case, we see that on average, puppies are not statistically better at the skills test after the training program, since none of the p-values are close to zero.

Results - T-Test, Paired Samples-1					
<b>Output</b> <a href="#">Bad Data Report</a> <a href="#">Sample Summary Stats</a>	TTest Type	T Statistic	Two Tailed PValue	Lower One Tailed PValue	Upper One Tailed PValue
	Paired T-Test	-0.3798	0.7047	0.3523	0.6477

### Data Output

- T Statistic** - A value computed based on the average and variance. The higher the magnitude of the t-statistic, the higher the difference between the means.
- Two Tailed PValue** - The sum of the area under the Students t-distribution above the absolute value of the t-statistic and below the inverse of the t-statistic. A higher value indicates a greater absolute difference in the sample compared. We

usually reject the null hypothesis if  $p < 0.05$ .

- Lower One Tailed PValue - The area under the Student's t-distribution between negative infinity and the t statistic. A lower p-value indicates that sample a is less than sample b. We usually reject the null hypothesis if  $p < 0.05$ .
- Upper One Tailed PValue - The area under the Student's t-distribution between positive infinity and the t statistic. A lower p-value indicates that sample a is greater than sample b. We usually reject the null hypothesis if  $p < 0.05$ .

## T-Test - Single Sample

Tests for statistical significance between a set of numeric values (from one column) and a known mean. This operator allows one to compute the test across several different sample columns with one operator.



### Information at a Glance

Parameter	Description
Category	Model Validation
Data source type	HD
Send output to other operators	Yes
Data processing tool	Spark

The single sample t-test is used to test whether a sample population has a significantly different mean from the known population mean.

For information about Student's t-distribution, see [https://en.wikipedia.org/wiki/Student%27s\\_t-distribution](https://en.wikipedia.org/wiki/Student%27s_t-distribution).

## Algorithm

The means and variances for all of the test statistics are computed using Spark's [MultivariateStatisticalSummary](#) object, but the t-tests themselves are computed from Java's [commons-math](#) library.

## Input

A tabular data set with numeric columns.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Sample Columns</b>	Select the column(s) of numeric values on which to compute the t-test.
<b>Assumed Mean</b>	Enter a numeric value (the population mean) against which to compute the t-test.
<b>Write Bad Data To File</b>	<p>Rows with null values are removed from the analysis. This parameter allows you to specify that the data with null values be written to a file.</p> <p>The file is written to the same directory as the rest of the output. The file name is suffixed with <code>_baddata</code>.</p> <ul style="list-style-type: none"> <li>• <b>Do Not Write Null Rows to File</b> - remove null value data and display in the result UI, but do not write to an external file.</li> <li>• <b>Do Not Write or Count Null Rows</b> (Fastest) - remove null value data but do not count and display in the result UI.</li> <li>• <b>Write All Null Rows to File</b> - remove null value data and write all</li> </ul>

Parameter	Description
	removed rows to an external file.
<b>Storage Format</b>	<p>Select the format in which to store the results. The storage format is determined by your type of operator.</p> <p>Typical formats are <b>Avro</b>, <b>CSV</b>, <b>TSV</b>, or <b>Parquet</b>.</p>
<b>Compression</b>	<p>Select the type of compression for the output.</p> <p>Available Parquet compression options.</p> <ul style="list-style-type: none"> <li>• <b>GZIP</b></li> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul> <p>Available Avro compression options.</p> <ul style="list-style-type: none"> <li>• <b>Deflate</b></li> <li>• <b>Snappy</b></li> <li>• no compression</li> </ul>
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>

## Output

### Visual Output

Each row represents a column selected in the **Sample Columns** parameter.

See [Single Sample T-Test Use Case](#) for example data on a puppy training program that illustrates use of the single sample t-test. In this case, we see that we have above average puppies since the **Upper One Tailed PValue** for the **Score\_Before\_Training** column is very close to zero, and that after training the puppies, they are still above average since the **Upper One Tailed PValue** for the **Score\_After\_Training** column is also close to zero.

Results - T-Test, Single Sample-1						
<b>Output</b> <a href="#">Bad Data Report</a> <a href="#">Sample Summary Stats</a>	TTest Type	Column Tested	T Statistic	Two Tailed PValue	Lower One Tailed PValue	Upper One Tailed PValue
	Single Sample T-Test	Score_Before_Training	27.7099	5.1059e-58	1	2.553e-58
	Single Sample T-Test	Score_After_Training	27.1118	6.4222e-57	1	3.2111e-57

## Data Output

- **T Statistic** - A value computed based on the average and variance. The higher the magnitude of the t-statistic, the higher the difference between the means.
- **Two Tailed PValue** - The sum of the area under the Student's t-distribution above the absolute value of the t-statistic and below the inverse of the t-statistic. A higher value indicates a greater absolute difference in the sample compared. We usually reject the null hypothesis if  $p < 0.05$ .
- **Lower One Tailed PValue** - The area under the Student's t-distribution between negative infinity and the t statistic. A lower p-value indicates that sample a is less than sample b. We usually reject the null hypothesis if  $p < 0.05$ .
- **Upper One Tailed PValue** - The area under the Student's t-distribution between positive infinity and the t statistic. A lower p-value indicates that sample a is greater than sample b. We usually reject the null hypothesis if  $p < 0.05$ .

## Tool Operators

The Tools Operators are a collection of functions that are helpful with the TIBCO Data Science - Team Studio modeling process and with extending the functionality of TIBCO Data Science - Team Studio flows.

## Convert

Provides a method for converting a Hadoop CSV file into either Avro or Parquet format.





## Information at a Glance

Parameter	Description
Category	Tools
Data source type	HD
Send output to other operators	Yes
Data processing tool	n/a

## Input

One CSV data set from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Storage Format</b>	Select the format in which to store the results. The storage format is determined by your type of operator.  Typical formats are <b>Avro</b> , <b>CSV</b> , <b>TSV</b> , or <b>Parquet</b> .
<b>Compression</b>	Select the type of compression for the output.

---

Available Parquet compression options.

- **GZIP**
- **Deflate**
- **Snappy**
- no compression

Available Avro compression options.

- **Deflate**
  - **Snappy**
  - no compression
- 

<b>Results Location</b>	The HDFS directory where the results of the operator are stored. This is the main directory, the sub-directory of which is specified in <b>Results Name</b> . Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.
-------------------------	---

---

<b>Results Name</b>	The name of the file in which to store the results.
---------------------	---

---

<b>Overwrite</b>	<p>Specifies whether to delete existing data at that path and file name.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - Fail if the path already exists.</li> </ul>
------------------	--

---

## Output

### Visual Output

A preview of the data.

### Data Output

A data set of the chosen format and compression.

## Export

Saves a trained model and stores it as a work file in the workspace. Models can also be stored in TIBCO® ModelOps, if it is available and configured.



### Information at a Glance

Parameter	Description
Category	Tools
Data source type	HD, DB
Send output to other operators	No
Data processing tool	n/a

For detailed information about the types of model operators that you can export to the supported file types, see [Model Export Formats](#)

## Input

A trained model operator to export.

## Restrictions

Currently, the following model configurations are not supported for PMML (Predictive Model Markup Language) export.

- SVM Classification: Only Hadoop SVM is supported for linear kernel type models (not Gaussian or polynomial).
- Alpine Forest: Only Hadoop Alpine Forest model export is currently supported.

To enable the export of models to TIBCO® ModelOps, perform the following steps:

1. The system administrator should add the following configuration to the `alpine.conf` file.

```
modelops
{
  enabled = true
  hostname = "modelops_hostname"
  https = true
}
```

**Note:**

Replace the hostname parameter with the hostname of the TIBCO ModelOps instance. For example, *hostname = "modelops-1.devmodelops120.streamingaz.tibcocloud.com"*

2. Restart the Chorus container.

## Configuration

If the TIBCO® ModelOps is not configured, users have the following parameters.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Export Format</b>	Format of the exported model. <ul style="list-style-type: none"> <li>• <b>Analytics Model Format</b> (the default)</li> <li>• <b>PFA</b></li> <li>• <b>PMML</b></li> </ul>
<b>File Name</b>	The file name for the exported model.  Default value: <b>@flow_name_model_0</b>  For more information about the variables used for the default name, see <a href="#">Workflow Variables</a> .

Parameter	Description
<b>Ancestry Depth</b>	A value between <b>1</b> (the default) and <b>3</b> , representing how many (supported) operators before the model should be exported when using Analytics Model or PFA formats.

If the TIBCO® ModelOps is available and configured, the following parameters are also available, in addition to those described above.

<b>Export Location</b>	Options: <ul style="list-style-type: none"> <li>• <b>Workspace</b> (the default)</li> <li>• <b>TIBCO ModelOps</b></li> </ul>
<b>Username</b>	The username for your TIBCO® ModelOps account. Default value: empty, but it can be configured by an administrator.
<b>Password</b>	The password for your TIBCO® ModelOps account. Default value: empty, but can be configured by an administrator.
<b>Project</b>	Select a TIBCO® ModelOps project to store the exported model. The dropdown list includes all projects that are available to the user specified in the credentials above.
<b>Overwrite?</b>	Choose whether to overwrite an existing model in the specified TIBCO® ModelOps project. <ul style="list-style-type: none"> <li>• If <b>Yes</b> (the default) and the TIBCO® ModelOps Project has an existing model with the same name as specified in <b>File Name</b>, then AMS checks out the existing model and commits a new version.</li> <li>• If <b>No</b>, the operator fails with a warning that a model already exists in the project.</li> </ul>

## Output

### Work file Output

The exported PMML (Predictive Model Markup Language), PFA (Portable Format for Analytics), or Analytics Model file is exported and saved as a work file within the current workspace, or in the specified project on the Artifact Management Server.

Click **Download File** to download the PMML file to the desktop.



**Important:** When you run the download file from within Safari browser, the generated PMML file is downloaded in XML format. Google Chrome downloads the correct PMML format.

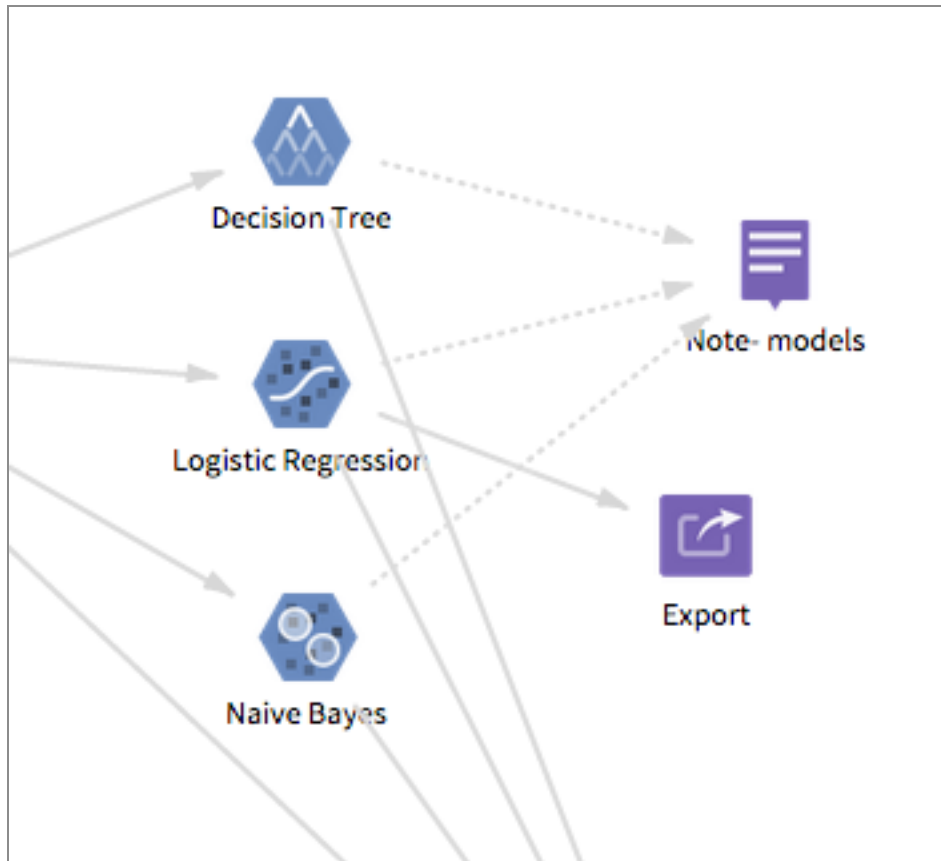
### Visual Output

None.

### Data Output

None. This is a terminal operator.

## Example



## Export to Excel (DB)

Exports multiple inputs as separate tabs to an Excel Workbook stored in the current workspace.



## Information at a Glance

Parameter	Description
Category	Tools
Data source type	DB
Send output to other operators	No
Data processing tool	n/a



**Note:** The Export to Excel (DB) operator is for database data only. For Hadoop data, use the [Export to Excel \(HD\)](#) operator.

Export to Excel (DB) supports database tables as inputs. The workbook sheet names are the (cleaned) input operator names, sorted alphabetically.

## Input

Tabular data sets from preceding operators.

### Bad or Missing Values

Null and empty values in the inputs are converted to blank cells in the workbook.

## Restrictions

If the workbook size is above the TIBCO Data Science - Team Studio work file size limit set for your instance, the operator fails and reports an error with the estimated file size and instance limitation value.

This setting is configurable in the TIBCO Data Science - Team Studio configuration properties in the file `alpine.conf`.

- `custom_operator.excel_export.db_table_max_cells` for Export to Excel (DB), maximum 50000000, or 50 million cells (1M rows \* 50 columns).
- `custom_operator.excel_reader.max_mb_input_stream` for Excel Reader (DB and HD), maximum 30MB.



## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Output Workbook Name</b>	Specify the name of the workbook stored in the TIBCO Data Science - Team Studio Workspace.
<b>Create New Version if Exists</b>	<p><b>true</b> (the default) - A new version of the workbook is created in the workspace, and previous versions are still accessible through the work file version-control button.</p> <p><b>false</b> - If a work file of the same name already exists, the operator fails with an output error message that notifies the user of the existing work file.</p>

## Output

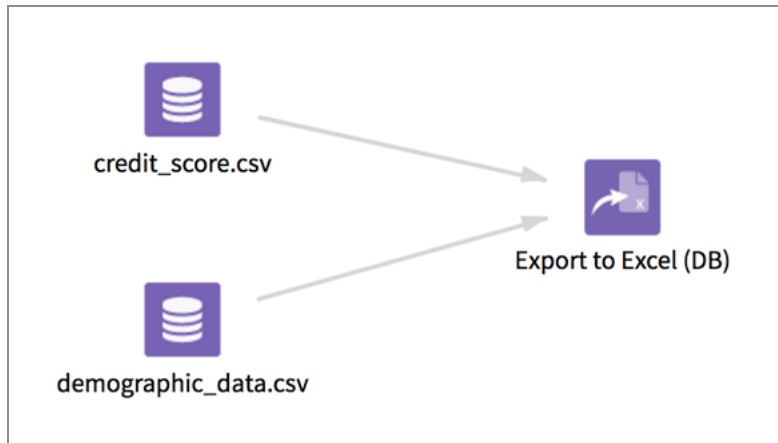
### Visual Output - Summary

Results - Export to Excel (DB)	
Summary	Parameters selected
	Number of Sheets: 2
	Sheet 1: credit
	Sheet 2: demographics
Output	
The Excel workbook is stored in the current workspace as my_workbook_DB.xlsx	

### Data Output

None. This is a terminal operator.

## Example



## Export to Excel (HD)

Exports multiple inputs as separate tabs to an Excel Workbook stored in the current workspace.



## Information at a Glance

Parameter	Description
Category	Tools
Data source type	HD
Send output to other operators	No
Data processing tool	n/a

**i Note:** The Export to Excel (HD) operator is for Hadoop data only. For database data, use the [Export to Excel \(DB\)](#) operator.

Export to Excel (HD) supports tabular HDFS inputs. The workbook sheet names are the (cleaned) input operator names, sorted alphabetically.

## Input

Tabular data sets from preceding operators. Avro and Parquet inputs are supported.

### Bad or Missing Values

Null and empty values in the inputs are converted to blank cells in the workbook.

## Restrictions

If the workbook size is above the TIBCO Data Science - Team Studio work file size limit set for your instance, the operator fails and reports an error with the estimated file size and instance limitation value.

This setting is configurable in the TIBCO Data Science - Team Studio configuration properties in the file `alpine.conf`.

- `custom_operator.excel_export.max_mb_input_size_alpine_server` for Export to Excel (HD), maximum 30MB.
- `custom_operator.excel_reader.max_mb_input_stream` for Excel Reader (DB and HD), maximum 30MB.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Output Workbook</b>	Specify the name of the workbook stored in the TIBCO Data Science - Team Studio Workspace.

Parameter	Description
<b>Name</b>	
<b>Create New Version if Exists</b>	<p><b>true</b> - A new version of the workbook is created in the workspace, and previous versions are still accessible through the work file version-control button.</p> <p><b>false</b> - If a work file of the same name already exists, the operator fails with an output error message that notifies the user of the existing work file.</p>

## Output

### Visual Output - Summary

Results - Export to Excel (DB)	
Summary	<p><b>Parameters selected</b></p> <p>Number of Sheets: 2</p> <p>Sheet 1: credit</p> <p>Sheet 2: demographics</p> <p><b>Output</b></p> <p>The Excel workbook is stored in the current workspace as my_workbook_DB.xlsx</p>

### Data Output

None. This is a terminal operator.

## Export to FTP

Exports a single database table to an FTP or SFTP server. Supports password authentication.



## Information at a Glance

Parameter	Description
Category	Tools
Data source type	DB
Send output to other operators	No
Data processing tool	Database

## Input

A database table from a preceding operator.

## Bad or Missing Values

Null values in the input table are converted to the string `null`. Empty values are kept as blank values in the FTP file exported.

## Restrictions

The default maximum number of values allowed for the input table is 150 million (for example, 3M rows \* 50 columns). If the input is larger, the operator fails with a meaningful error message. This limit is set to prevent causing memory issues on the TIBCO Data Science - Team Studio server; however, this value is configurable in the configuration file `alpine.conf`. Set the following parameter in the configuration file.

```
custom_operator.ftp_export.db_table_max_cells = 150000000
```

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.

Parameter	Description
<b>Server Protocol</b>	Specify the FTP server protocol, either <b>FTP</b> (the default) or <b>SFTP</b> .
<b>Server Hostname</b>	Enter the FTP server hostname (without the protocol).
<b>Server Port</b>	Enter the FTP server port.
<b>Username</b>	Enter the username to authenticate on the FTP server.
<b>Password</b>	Enter the password for the specified username to authenticate on the FTP server.
<b>Account (FTP Only)</b>	Optional - For FTP only. Specify the account password if one is required to authenticate on the FTP server.
<b>Delimiter</b>	Specify one of the following delimiters for the output file. <ul style="list-style-type: none"> <li>• <b>Comma</b> (the default)</li> <li>• <b>Tab</b></li> <li>• <b>Pipe</b></li> <li>• <b>Tilde</b></li> </ul>
<b>Text Qualifier</b>	Specify whether the operator should handle escaping special characters and enclosing values in double quotes, if required.  Can be either <b>None</b> (the default) or <b>Double Quotes</b> .  If this option is set to <b>Double Quotes</b> , the following rules apply. <ul style="list-style-type: none"> <li>• If the input value contains a comma, newline, or double quote, the output value is enclosed in double quotes.</li> <li>• Any double-quotes character in the input value is escaped with another double quote.</li> <li>• If the value does not contain a comma, newline, or double quote, it is</li> </ul>

Parameter	Description
	returned unchanged.
<b>Output Directory</b>	The location to store the output files.
<b>Output Name</b>	The name to contain the results.
<b>Overwrite File if Exists</b>	<p>If set to <b>true</b> (the default), the existing file on the FTP server is overwritten.</p> <p><b>Note:</b> If the output path corresponds to an existing directory, it is not overwritten. An error is displayed with the message to delete this directory or to choose a different combination of <b>Output Directory</b> and <b>Output Name</b>.</p>
<b>Fetch Size</b>	<p>Define the number of entries to read from database at once.</p> <p>Default value: <b>20000</b>.</p>

## Output

### Visual Output

#### Visual Output - Summary

Summary	<p><b>Parameters selected</b></p> <p>FTP Server Hostname: 10.0.0.42</p> <p>FTP Server Port: 21</p> <p>Input Table: "auto_test_data"."Pred_Output_Small"</p> <p>Delimiter: Comma</p> <p>Text Qualifier: None</p> <p>Output Path: /test/output</p> <p>Overwrite File: true</p> <p><b>Output</b></p> <p>The DB Table was exported to FTP at location: /test/output.</p>
---------	--

### Data Output

None. This is a terminal operator.

## Additional Notes

### Permissions on Output File

Permissions on the output file are set to 640, and if the directory was created during the export process (that is, it did not exist already), directory permissions are set to 750.

**i Note:** Some FTP servers do not support the chmod command. In this case, permissions are not changed on the output file and output directory. A warning in the user interface indicates that the permissions could not be changed.

## Export to SBDF (DB)

Converts a database table to the Spotfire binary data frame (SBDF) format. The SBDF files are stored in the same workspace as the workflow and can be downloaded for use in TIBCO Spotfire.



### Information at a Glance

Parameter	Description
Category	Tools
Data source type	DB
Send output to other operators	No
Data processing tool	n/a

**i Note:** The Export to SBDF (DB) operator is for database data only. For Hadoop data, use the [Export to SBDF \(HD\)](#) operator.



## Input


A single database table.

### Bad or Missing Values

Null and empty values in the inputs are converted to blank cells in the SBDF file.

## Restrictions

By default, the custom operator can export only up to 10GB of data. You can modify this limit in the `alpine.conf` file by setting the following configuration.

 **Note:** Raising this limit might overload the memory already reserved for TIBCO Data Science - Team Studio.

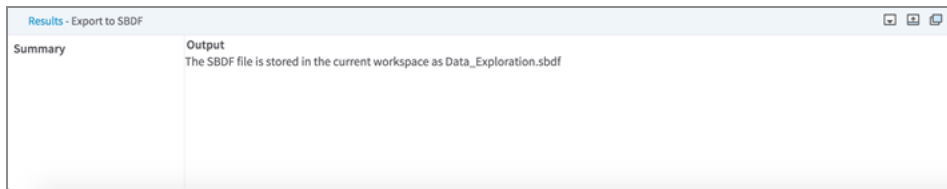
```
custom_operator.sbdf_export.max_mb_input_size_alpine_server=10240
```

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>File Output Name</b> *required	Specify the name of the SBDF file stored in the TIBCO Data Science - Team Studio workspace. Note that the <code>.sbdf</code> file extension should be omitted, because it is automatically appended to the file name.
<b>Overwrite if Exists</b> *required	<p>Select <b>true</b> or <b>false</b>. If the SBDF file already exists, and if this option is set to <b>true</b>, the file is overwritten. A new version of the file is created in the workspace, and previous versions are still accessible through the work file version-control button.</p> <p><b>false:</b> If a work file of the same name already exists, the operator fails with an output error message that notifies the user of the existing work file.</p>

## Output

### Visual Output



### Data Output

None. This is a terminal operator.

## Export to SBDF (HD)

Converts an HDFS tabular data set to the Spotfire binary data frame (SBDF) format. The SBDF files are stored in the same workspace as the workflow and can be downloaded for use in TIBCO Spotfire.



### Information at a Glance

Parameter	Description
Category	Tools
Data source type	HD
Send output to other operators	No
Data processing tool	n/a



**Note:** The Export to SBDF (HD) operator is for Hadoop data only. For database data, use the [Export to SBDF \(DB\)](#) operator.

## Input


A single HDFS tabular data set. Avro and Parquet inputs are not supported.

### Bad or Missing Values

Null and empty values in the inputs are converted to blank cells in the SBDF file.

## Restrictions

By default, the custom operator can export only up to 10GB of data. You can modify this limit in the `alpine.conf` file by setting the following configuration.

 **Note:** Raising this limit might overload the memory already reserved for TIBCO Data Science - Team Studio.

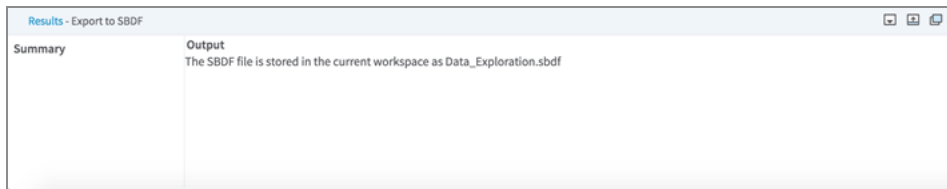
```
custom_operator.sbdf_export.max_mb_input_size_alpine_server=10240
```

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>File Output Name</b> *required	Specify the name of the SBDF file stored in the TIBCO Data Science - Team Studio workspace. Note that the <code>.sbdf</code> file extension should be omitted, because it is automatically appended to the file name.
<b>Overwrite if Exists</b> *required	<p>Select <b>true</b> or <b>false</b>. If the SBDF file already exists, and if this option is set to true, the file is overwritten. A new version of the SBDF file is created in the workspace, and previous versions are still accessible through the work file version-control button.</p> <p><b>false:</b> If a work file of the same name already exists, the operator fails with an output error message that notifies the user of the existing work file.</p>

## Output

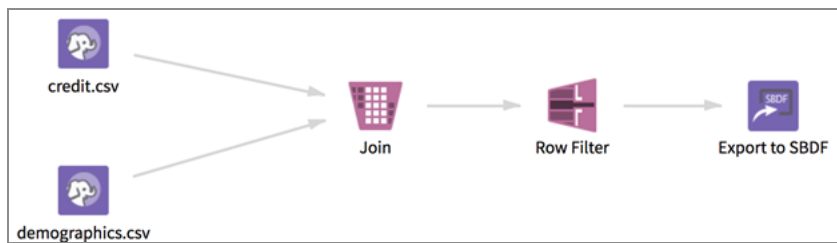
### Visual Output



### Data Output

None. This is a terminal operator.

## Example



## Flow Control

Provides a method for continuing or stopping flows depending on customized test conditions.



## Information at a Glance

Parameter	Description
Category	Tools
Data source type	DB, HD
Send output to other operators	Yes
Data processing tool	Pig

A typical use case for the Flow Control operator might be a test condition that checks for new data, with the flow being processed if new data is detected, and halted otherwise.

## Input

One or two data sets from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Test Dataset</b>	Used to determine the test condition for processing or not processing the flow. This can either be the same data set as the passthrough data set, or a different test condition data source (such as the results from a Row Filter).
<b>Test Condition</b>	<p>If the <b>Test Dataset</b> matches the <b>Test Condition</b>, the flow continues. Otherwise, the <b>If Test Condition Fails</b> parameter determines the resulting flow behavior.</p> <ul style="list-style-type: none"> <li>• <b>is empty</b></li> <li>• <b>has rows</b> (the default)</li> </ul>

Parameter	Description
<b>If Test Condition Fails</b>	<p>If the <b>Test Condition</b> is not met, this branch of the flow ends and, optionally, produces an error.</p> <ul style="list-style-type: none"> <li>• <b>Stop flow branch</b> (the default)</li> <li>• <b>Stop flow branch with error</b></li> </ul>
<b>Passthrough Dataset</b>	<p>If the <b>Test Condition</b> passes, the passthrough data set is passed along to the rest of the workflow.</p> <p>If only one data set is connected to this operator, the test data set and the passthrough data set are the same.</p>

## Output

### Visual Output

A message that indicates whether the test condition was met.

If Stop flow branch with error is set, the error appears in the log as follows:

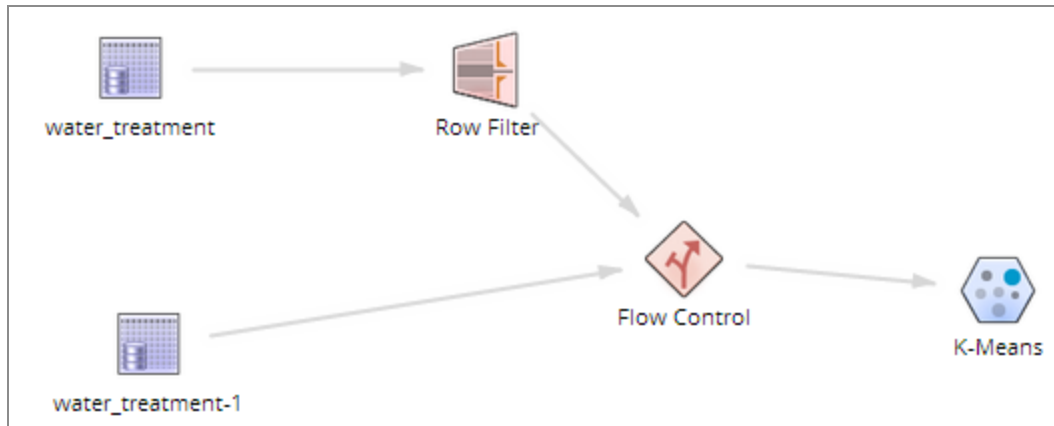
```
[13:16:42] process flow running error .....
           Execution of 'Flow Control' failed. Error details: Condition failed - ending flow branch with error
[13:16:42] Analytic Flow finished
```

### Data Output

If the Flow Condition is passed, the designated **Passthrough Dataset** is passed to the next operator in the flow.

## Example

The following example flow employs the Row Filter operator to first test the data source for negative values. If the prescribed Test Condition exists, the Flow Control operator stops the flow from processing and produces an error in the log.



## HQL Execute

Executes a user-defined HiveQL clause.



### Information at a Glance

Parameter	Description
Category	Tools
Data source type	HD
Send output to other operators	Yes
Data processing tool	n/a

The HQL clause executed inside this operator does not have any dependency on its preceding or succeeding operator. The connection only ensures its execution sequence within the analytic workflow. The HQL Execute code can invoke any well-formed HQL statement based on Spark SQL, including UDFs. Only queries supported by Spark SQL are supported at this time.

## Input

No input is required. Alternatively, a Hive table, [Load To Hive](#) operator, or other HQL Execute operators (with **Pass Output File** = true) can be used as input to HQL Execute.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Data Source Name</b>	Choose the Hive data source that this runs on.
<b>SQL Clause</b>	<p>Define the HQL statement to execute. Only HQL accepted by the Spark SQL implementation is accepted.</p> <p>For more information, see <a href="#">Define SQL Statement dialog</a>.</p> <div> <p><b>Note:</b> Do not change the <i>alpine_sql_output</i> variable while editing a clause. Doing so disrupts the operator from running.</p> </div> <p>If you compute an aggregated column (for example, SELECT COUNT(*)), you must give it an alias for the HQL Execute operator to run successfully (for example, SELECT COUNT(*) AS total_count).</p>
<b>Hive Result Database Name</b>	Choose the database in which to place the results, if any.
<b>Hive Result Table Name</b>	Choose the table in which to place the results, if any.
<b>Drop If Exists</b>	<ul style="list-style-type: none"> <li>If <b>Yes</b> (the default), drop the existing table of the same name and create a new one.</li> <li>If <b>No</b>, stop the flow and alert the user that an error has occurred.</li> </ul>



Parameter	Description
<b>Pass Output File</b>	Pass the output file.
<b>Results File Structure</b>	Define the structure of the results file.

## Output

### Visual Output

None.

### Data Output

The data set from the preceding operator.

## Load Model

Loads a TIBCO Data Science - Team Studio Analytics Model from the current workspace to use with predictor and evaluator operators.



## Information at a Glance

Parameter	Description
Category	Tools
Data source type	DB, HD
Send output to other operators	Yes
Data processing tool	n/a

Analytics Models are models that have been trained within TIBCO Data Science - Team Studio and exported (using the Export operator) to the workspace. When the model is exported, you can use the Load Model operator to use your exported models in other workflows. The following list shows supported model operators.

- [Linear Regression \(HD\)](#)
- [Logistic Regression \(HD\)](#)
- [K-Means \(HD\)](#)
- [Naive Bayes \(DB\)](#)
- [PCA \(HD\)](#)
- [Gradient Boosting Classification](#)
- [Gradient Boosting Regression](#)

## Input

Load Model is a source operator - no inputs are required.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Model</b>	The model to load. The list includes all Analytics Model (.am) files in the current workspace.

## Output

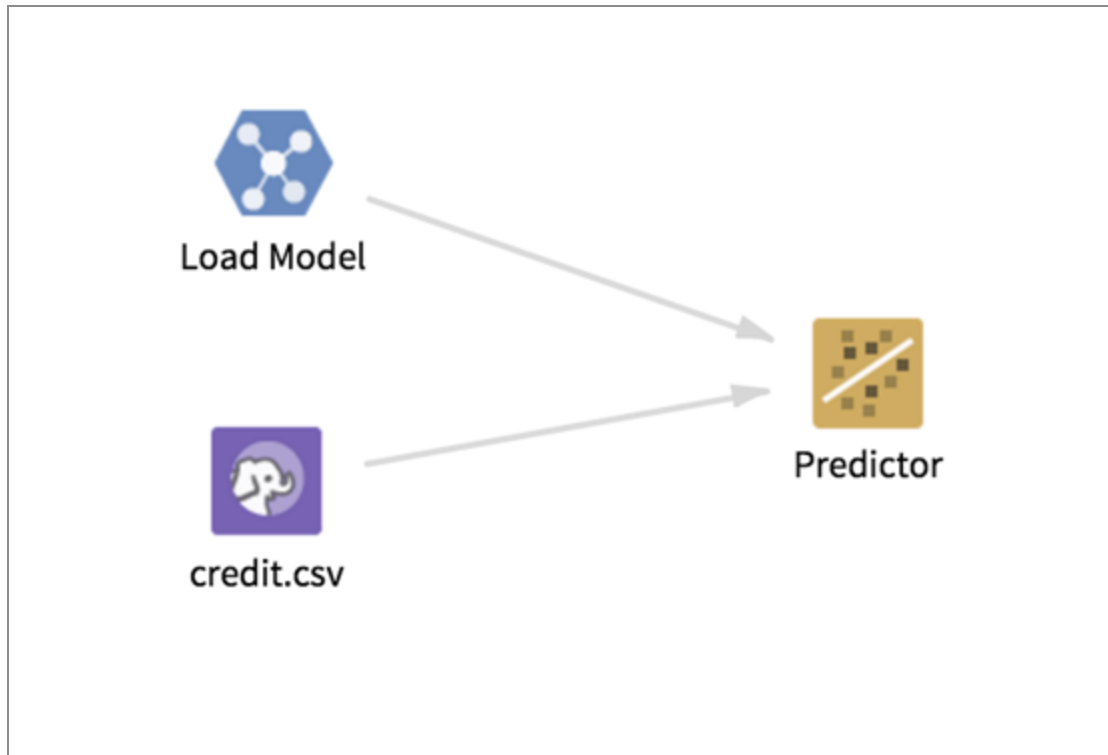
### Visual Output

None.

### Data Output

A model that corresponds to the selected Analytics Model type.

## Example



## Note

Embeds explanatory information within a workflow.



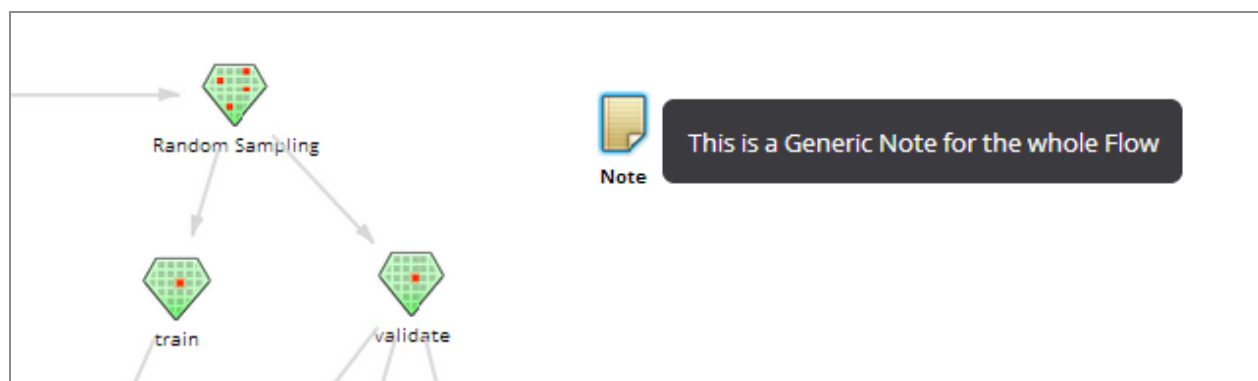
## Information at a Glance

Parameter	Description
Category	Tools
Data source type	DB, HD

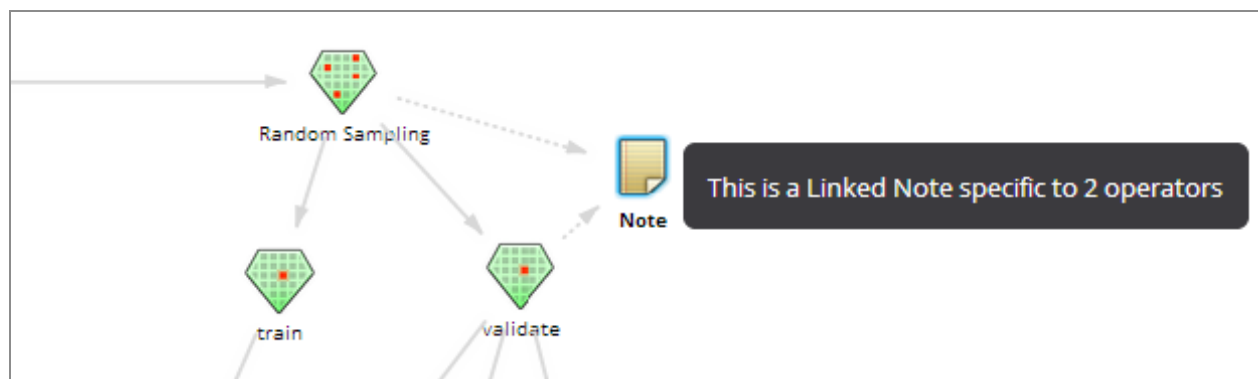
Parameter	Description
Send output to other operators	n/a
Data processing tool	n/a

The information can be left floating in the workflow as a generic note, or it can be linked specifically to one or more operators as a more specific note. The note content is displayed as a tool tip.

The following image shows a floating note.



The following image shows a note linked to two operators.



## Input

None. Optionally, you can link a note to one or more operators.

## Configuration

Parameter	Description
Note	Explanatory text about the workflow.

## Output

None.

## Pig Execute

Executes a user-defined Pig script (for parsing and sorting Hadoop data sources). The Pig Execute operator can also reference Pig UDFs (user-defined functions) that are supplied to the TIBCO Data Science - Team Studio server.



## Information at a Glance

Parameter	Description
Category	Tools
Data source type	HD
Send output to other operators	Yes
Data processing tool	Pig

The Pig script executed inside this operator is passed the results from its preceding operators, and it is expected to pass along its results to the succeeding operator.

In version 5.7 and later, the resulting file structure of the output is detected automatically. Otherwise, the user must define the output structure.

## Input

The Pig Execute operator can accept one or more inputs; however, input is not required.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Pig Script</b>	<p>The Pig script to execute.</p> <p>Click <b>Edit Pig Script</b> to open the Pig Script editor dialog. For more information, see <a href="#">Define Pig Script dialog</a>.</p>
<b>Pass Output File</b>	Specify whether to pass the output to the next operator.
<b>Results Location</b>	The HDFS directory where the results of the operator are stored. This is the main directory, the subdirectory of which is specified in <b>Results Name</b> . Click <b>Choose File</b> to open the Hadoop File Explorer dialog and browse to the storage location. Do not edit the text directly.
<b>Results Name</b>	The name of the file in which to store the results.
<b>Overwrite</b>	<p>Specifies whether to delete existing data at that path and file name.</p> <ul style="list-style-type: none"><li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li><li>• <b>No</b> - Fail if the path already exists.</li></ul>

## Output

### Visual Output

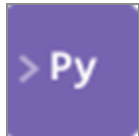
A preview of the output of the Pig script.

## Data Output

The data created in the operator.

## Python Execute (DB)

Runs a Jupyter notebook stored in your current workspace from a workflow in TIBCO Data Science - Team Studio.



### Information at a Glance

Parameter	Description
Category	Tools
Data source type	DB
Send output to other operators	Yes
Data processing tool	Python

**Note:** The Python Execute (DB) operator is for database data only. For Hadoop data, use the [Python Execute \(HD\)](#) operator.

**Notebook setup:** For a notebook to be usable with the Python Execute operator, it must have the automatically generated tag Ready For Python Execute visible in your workspace. This attribute is set if the following conditions are met.

- At least one input or output is specified in the notebook with argument `use_input_substitution = True` or `use_output_substitution = True`.
- Notebook input(s) argument `execution_label` are distinct and exclusively one of the following strings: 1, 2 or 3.
- For example, your Notebook code might look something like this:

```
df_account=cc.read_input_table(table_name='account', schema_name='demo',
database_name='miner_demo',use_input_substitution=True, execution_
label="1")
```

- Inputs/output defined with `use_input_substitution = True` must all be DB inputs (in this case, the notebook is usable with the Python Execute (DB) operator).

## Input

From zero to three inputs to use as substitute inputs for the notebook selected, depending on the number of inputs for substitution the notebook configuration allows.

You can substitute up to three inputs, or use the inputs defined in the notebook if you prefer not to specify substitutions. To run Python Execute, each substituted input must contain a superset of columns in the corresponding notebook input with compatible data types. One data set can be output, or zero outputs if this is a terminal operator in your workflow.

Depending on the notebook configuration for input and output, the operator can be a source operator (if no inputs are selected for substitution in the notebook), or a terminal operator (if no output is specified in the notebook). If a single output is specified, the operator transmits this output to subsequent operators.

## Bad or Missing Values

Missing values are not removed if present in the input(s). They should be handled directly in the notebook or in preceding steps of the workflow.

## Restrictions

If the notebook selected has no tabular output defined with argument `use_output_substitution = True`, the Python Execute operator does not transmit any data to subsequent operators and is considered a terminal operator. Although following operators cannot run, the user can still draw a connection with subsequent operators.

The Python Execute (DB) operator can be used only in a flow connected to a single database data source.

If the notebook selected is set up to transmit an output that contains variables with datetime format, the Python Execute operator transmits those as string variables to the next operator (the user can then convert those to the correct format in a Variables operator).



## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator..
<b>Notebook</b>	<p>Select the Python notebook to run in your current workspace. To appear in this list, notebooks must be set up for use with Python Execute.</p> <p><b>Note:</b> Clicking <b>Open Notebook Selected</b> opens the notebook in a new browser tab.</p>
<b>Substitute Input 1</b>	<p>Optional. Select the connected input to use as a substitute for notebook input with argument <code>execution_label = 1</code>.</p> <p>If the notebook contains such input and you do not select a substitute in your workflow, it runs with the input defined in the notebook.</p>
<b>Substitute Input 2</b>	<p>Optional. Select the connected input to use as a substitute for notebook input with the argument <code>execution_label = 2</code>.</p> <p>If the notebook contains such input and you do not select a substitute in your workflow, it runs with the input defined in the notebook.</p>
<b>Substitute Input 3</b>	<p>Optional. Select the connected input to use as a substitute for notebook input with the argument <code>execution_label = 3</code>.</p> <p>If the notebook contains such input and you do not select a substitute in your workflow, it runs with the input defined in the notebook.</p>
<b>Data Source (DB)</b>	<p>Select the data source in which to store the output from notebook execution (if defined).</p> <p>If inputs are connected to the Python Execute operator, <b>Data Source (DB)</b> must match the data source of your inputs.</p>
<b>Output Schema</b>	The schema for the output table or view.

Parameter	Description
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

The operator result panel displays one or two tabs, depending on whether it is terminal.

- **Output** (only available if the operator is not terminal):

Output Summary		ID	AGE_IN_YEARS	LEVEL_OF_EDUCATION	YEARS_WITH_CURRENT_EMPLOYER	YEARS_AT_CURRENT_ADDRESS
		1	41	High school degree	8	21
		2	41	Some college	18	6
		2	41	Some college	18	6
		3	41	Some college	7	6
		3	41	Some college	7	6
		4	41	High school degree	2	22
		4	41	High school degree	2	22
		4	41	High school degree	2	22

- **Summary** of the parameters selected and notebook execution results:

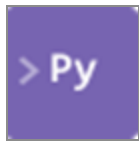
Summary	<p><b>Parameters Selected</b></p> <p>Notebook: 3.DB InputOutputNotebook.ipynb</p> <p>Number of Notebook Inputs: 2</p> <p>Number of Inputs Substituted: 2</p> <p><b>Output</b></p> <p>Execution Time: 15 seconds</p> <p>The execution of the notebook selected did not generate a table (as it was not specified in the notebook):</p> <p>The next operator won't receive any data.</p> <p><b>Text Output:</b></p> <p>/home/chorus/ChorusCommander/.pyenv/versions/project_env/lib/python2.7/commander/lib/ChorusCommander.py:875: FutureWarning: convert_objects is deprecated. Use the data-type specific converters pd.to_datetime, pd.to_timedelta and pd.to_numeric. return df.convert_objects(convert_numeric=True)</p> <p>WARNING: "miner_demo"."demo"."account_ntbk" already exists and drop_if_exists set to False, table will not be created WARNING: nothing write into "miner_demo"."demo"."account_ntbk", the append_if_exists is set to False</p>
---------	--

## Data Output

If the notebook contains an output with argument `use_output_substitution = True`, the operator transmits a tabular data set to subsequent operators. If no output is defined in the notebook, this operator is terminal.

## Python Execute (HD)

Runs a Jupyter notebook stored in your current workspace from a workflow in TIBCO Data Science - Team Studio.



## Information at a Glance

Parameter	Description
Category	Tools
Data source type	HD
Send output to other operators	Yes
Data processing tool	PySpark

**i Note:** The Python Execute (HD) operator is for Hadoop data only. For database data, use the [Python Execute \(DB\)](#) operator.

**Notebook setup:** For a notebook to be usable with the Python Execute operator, it must have the automatically generated tag `Ready For Python Execute` visible in your workspace. This attribute is set if the following conditions are met.

- At least one input or output is specified in the notebook with argument `use_input_substitution = True` or `use_output_substitution = True`.
- Notebook input(s) argument `execution_label` are distinct and exclusively one of the

following strings: 1, 2 or 3.

- For example, your Notebook code might look something like this:

```
df_account=cc.read_input_table(table_name='account', schema_name='demo',
database_name='miner_demo',use_input_substitution=True, execution_
label="1")
```

- Inputs/output defined with `use_input_substitution = True` must all be Hadoop inputs (in this case, the notebook is usable with the Python Execute (HD) operator).

## Input

From zero to three inputs to use as substitute inputs for the notebook selected, depending on the number of inputs for substitution the notebook configuration allows.

You can substitute up to three inputs, or use the inputs defined in the notebook if you prefer not to specify substitutions. To run Python Execute, each substituted input must contain a superset of columns in the corresponding notebook input with compatible data types. One data set can be output, or zero outputs if this is a terminal operator in your workflow.

Depending on the notebook configuration for inputs and output, the operator can be a source operator (if no inputs are selected for substitution in the notebook), or a terminal operator (if no output is specified in the notebook). If a single output is specified, the operator transmits this output to subsequent operators.

## Bad or Missing Values

Missing values are not removed if present in the input(s). They should be handled directly in the notebook or in preceding steps of the workflow.

## Restrictions

If the notebook selected has no tabular output defined with argument `use_output_substitution = True`, the Python Execute operator transmits no data to subsequent operators and is considered a terminal operator. Although following operators cannot run, the user can still draw a connection with subsequent operators.

Parquet and Avro inputs are supported only with PySpark notebooks (that is, the `cc.read_input_file` method in the notebook should have the `sqlContext` argument specified).

If the notebook selected is set up to transmit an output that contains variables with datetime format, the Python Execute operator transmits those as string variables to the

next operator. (The user can then convert those to the correct format in a Variables operator.)

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Notebook</b>	<p>Select the Python/PySpark notebook to run in your current workspace. To appear in this list, notebooks must be set up for use with Python Execute.</p> <p><b>Note:</b> Clicking <b>Open Notebook Selected</b> opens the notebook in a new browser tab.</p>
<b>Substitute Input 1</b>	Optional. Select the connected input to use as a substitute for notebook input with the argument <code>execution_label = 1</code> . If the notebook contains such input and you do not select a substitute in your workflow, it runs with the input defined in the notebook.
<b>Substitute Input 2</b>	<p>Optional. Select the connected input to use as a substitute for notebook input with the argument <code>execution_label = 2</code>.</p> <p>If the notebook contains such input and you do not select a substitute in your workflow, it runs with the input defined in the notebook.</p>
<b>Substitute Input 3</b>	Optional. Select the connected input to use as a substitute for notebook input with the argument <code>execution_label = 3</code> . If the notebook contains such input and you do not select a substitute in your workflow, it runs with the input defined in the notebook.
<b>Data Source (HD)</b>	<p>Select the Hadoop data source in which to store output from notebook execution (if defined).</p> <p>If inputs are connected to the Python Execute operator, <b>Data Source (HD)</b> must match the data source of your inputs.</p>
<b>Output</b>	The location to store the output files.

Parameter	Description
<b>Directory</b>	
<b>Output Name</b>	The name to contain the results.
<b>Overwrite Output</b>	<p>Specifies whether to delete existing data at that path.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - if the path exists, delete that file and save the results.</li> <li>• <b>No</b> - fail if the path already exists.</li> </ul>

## Output

### Visual Output

The operator result panel displays one or two tabs, depending on whether it is terminal.

- **Output** (only available if the operator is not terminal):

Output Summary	ID	AGE_IN_YEARS	LEVEL_OF_EDUCATION	YEARS_WITH_CURRENT_EMPLOYER	YEARS_AT_CURRENT_ADDRESS
	1	41	High school degree	8	21
	2	41	Some college	18	6
	2	41	Some college	18	6
	3	41	Some college	7	6
	3	41	Some college	7	6
	4	41	High school degree	2	22
	4	41	High school degree	2	22
	4	41	High school degree	2	22

- **Summary** of the parameters selected and notebook execution results:

Summary	Parameters Selected
	Notebook: 3.DB InputOutputNotebook.ipynb
	Number of Notebook Inputs: 2
	Number of Inputs Substituted: 2
	Output
	Execution Time: 15 seconds
	The execution of the notebook selected did not generate a table (as it was not specified in the notebook):
	The next operator won't receive any data.
	Text Output:
	/home/chorus/ChorusCommander/.pyenv/versions/project_env/lib/python2.7/commander_lib/ChorusCommander.py:875: FutureWarning: convert_objects is deprecated. Use the data-type specific converters pd.to_datetime, pd.to_timedelta and pd.to_numeric. return df.convert_objects(convert_numeric=True)
	WARNING: "miner_demo":"demo":"account_ntbk" already exists and drop_if_exists set to False, table will not be created WARNING: nothing write into "miner_demo":"demo":"account_ntbk", the append_if_exists is set to False

### Data Output

If the notebook contains an output with argument `use_output_substitution = True`, the operator transmits a tabular data set to subsequent operators.

If no output is defined in the notebook, this operator is terminal.

## Example



## R Execute (DB)

To configure R Execute, connect a valid data source to the R Execute operator. An intermediate operator also constitutes a data source for R Execute.



## Information at a Glance

Parameter	Description
Category	Tools
Data source type	DB
Send output to other operators	No
Data processing tool	R engine

R Execute (DB) is for database data only. For Hadoop data, use the [R Execute \(HD\)](#) operator.

For information about configuring and using this operator, see [R Execute](#).

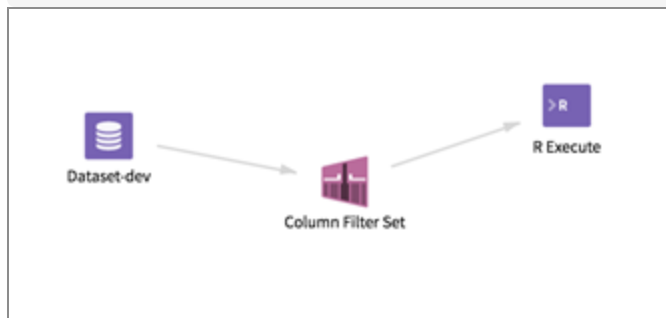
## Input

Specify that you require the input dataset by referring to an R object called *alpine\_input* in the script. This is a data frame object.

You might choose not to use the input dataset (by not referring to *alpine\_input* in the script), in which case the data is not read in to R.

- If the input is a preceding operator, the preceding operator runs, but the data is not transferred to R if you do not use *alpine\_input* in the script.
- If the preceding operator is a data source (a database table), the database query does not run, saving execution time.

**i Note:** Some kind of a data source must be specified, even if you do not use the input dataset in the R code. This is because you might still generate an output data frame, and the data frame must be stored in the same data store type as the one selected for the input (that is, database output for a database input). Also, it must be the same specific data store (the same database).



## Restrictions

See the topic "R Execute Prerequisites" in *TIBCO® Data Science - Team Studio System Requirements* for information about package, system, and server requirements.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the



Parameter	Description
	operator.
<b>R Script</b>	<p>The R Script to execute. Select <b>Define Clause</b> to specify the R script.</p> <p>See: <a href="#">Define R Script dialog</a>.</p>
<b>Output Schema</b>	Specifies the database schema to contain the table representation of the R Execute output data frame.
<b>Output Table</b>	Specifies the name of the database table, within the schema defined above, where the table representation of the R Execute output data frame is stored.
<b>Drop if Exists</b>	<ul style="list-style-type: none"> <li>• If <b>Yes</b> (the default), drop the existing table of the same name and create a new one.</li> <li>• If <b>No</b>, stop the flow and alert the user that an error has occurred.</li> </ul>
<b>Pass Output Table</b>	<p>Required only if R Execute is not a terminal operator.</p> <p>Specify whether to pass the R Execute output to the next operator.</p> <p>This option is important if you want to R Execute the operator's output to the next operator.</p> <p>If <b>Pass Output Table</b> is set to <b>Yes</b>, then the <b>Result Table Structure</b> details must be configured.</p> <p>Default value: <b>No</b></p>
<b>Result Table Structure</b>	<p>Specify the table structure of the operator output to pass to the next operator.</p> <p>This option dynamically pulls the values from the <i>alpine_output</i> variable in the R script. The values do not populate until you have run the R script once.</p> <p>Enabled only if <b>Pass Output File</b> is set to <b>Yes</b>.</p> <p>For more information, see <a href="#">Edit Table Columns dialog</a>.</p> <p>The following code provides an example of how to use <i>alpine_output</i>:</p>

Parameter	Description
	<pre>print("hello") foo = c("setosa", "virginica", "versicolor") alpine_output = data.frame(class = foo, predicted_class = foo)</pre>

## Output

### Visual Output

The table is stored whether **Pass Output Table** is set to **Yes**, and whether the **Results Table Structure** is provided. The **Pass Output Table** and **Results Table Structure** combination is used only to check the integrity of the flow in case the R Execute operator is followed by another operator.

If the *alpine\_output* object does not exist in your R code, then the output is not generated. If you set **Pass Output Table** to **Yes**, and if the R script does not contain a reference to the *alpine\_output* object, then the flow fails at runtime, and an error message is displayed for this inconsistency.

### Data

<b>Data</b> <a href="#">R-Console Output</a> <a href="#">R-Script</a>	play	wind	humidity	outlook	temperature
	no	false	85	sunny	85
	yes	false	96	rain	70

If you create an *alpine\_output* object in the R code, then the R Execute operator output displays the output data frame (persisted in the database) in the results console in the **Data** tab.

### R-Console Output

<b>Data</b> <a href="#">R-Console Output</a> <a href="#">R-Script</a>	outlook	temperature	humidity	wind	play
	overcast:4	Min. :64.00	Min. :65.00	false:8	no :5
	rain :5	1st Qu.:69.25	1st Qu.:71.25	true :6	yes:9
	sunny :5	Median :72.00	Median :80.00		
		Mean :73.57	Mean :80.29		
		3rd Qu.:78.75	3rd Qu.:88.75		
		Max. :85.00	Max. :96.00		

If your R code included functions that printed output to the R console, then the output is displayed in the results console in the **R-Console Output** tab.

**Note:** The R Execute operator's console printing behavior is a bit different from the R console or RStudio behavior. Normally, if you have a statement in the R code that reads `summary(alpine_input)`, it is shown in the R console or RStudio console. However, because R Execute is capturing the console output into an object (which is executed in R using R's `capture.output` function), you must wrap such calls using the `print()` function. For example, instead of `summary(alpine_input)`, specify `print(summary(alpine_input))`. This is a limitation of how R's `capture.output` function works.

### R-Script

<p>Data</p> <p>R-Console Output</p> <p>R-Script</p>	<pre>print(summary(alpine_input)) alpine_output = alpine_input</pre>
---	--

Your R code is shown in the results console in the **R-Script** tab.

### Output to Succeeding Operators

Describe any data output to the database/cluster or that which is passed on to the next operator.

## R Execute (HD)

To configure R Execute, connect a valid data source to the R Execute operator. An intermediate operator also constitutes a data source for R Execute.



## Information at a Glance

Parameter	Description
Category	Tools
Data source type	HD
Send output to other operators	No
Data processing tool	R engine

R Execute (HD) is for Hadoop data only. For database data, use the [R Execute \(DB\)](#) operator.

For information about configuring and using this operator, see [R Execute](#).

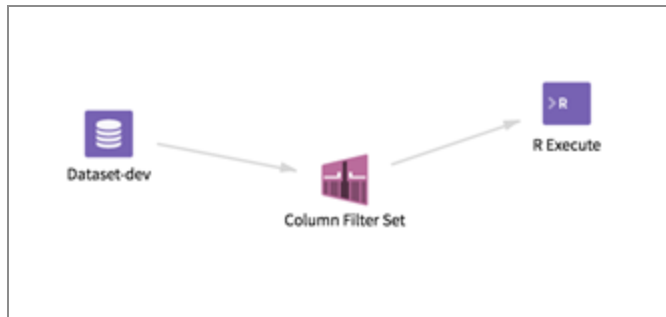
## Input

You specify that you require the input dataset by referring to an R object called *alpine\_input* in the script. This is a data frame object.

You might choose not to use the input dataset (by not referring to *alpine\_input* in the script), in which case the data is not read in to R.

- If the input is a preceding operator, the preceding operator runs, but the data is not transferred to R if you do not use *alpine\_input* in the script.
- If the preceding operator is a data source (a Hadoop file or the database table), the Hadoop data transfer or the database query does not run, saving execution time.

**Note:** Some kind of a data source must be specified, even if you do not use the input dataset in the R code. This is because you might still generate an output data frame, and the data frame must be stored in the same data store type as the one selected for the input (that is, Hadoop output for a Hadoop input, database output for a database input). Also, it must be the same specific data store (same Hadoop cluster or same database).



## Restrictions

See the topic "R Execute Prerequisites" in *TIBCO® Data Science - Team Studio System Requirements* for information about package, system, and server requirements.

## Configuration

<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>R Script</b>	The R script to execute. Select <b>Define Clause</b> to specify the R-Script.
<b>Results Location</b>	<p>Specifies the HDFS directory where the results of the R Execute operator are stored. This is the main directory, the subdirectory of which is specified in the <b>Results Name</b> option (see below).</p> <p>Click <b>Choose File</b> to specify the location. Do not edit the text directly.</p>
<b>Results Name</b>	Select the name of the Hadoop file where the results of the R Execute operator are stored.
<b>Overwrite</b>	<p>Determines whether the operator should overwrite an existing file if a file with the same name exists.</p> <p>Default value: <b>Yes</b></p>
<b>Pass Output File</b>	<p>Specifies whether to pass the R Execute output to the next operator.</p> <ul style="list-style-type: none"> <li>If set to <b>Yes</b>, then the <b>Results File Structure</b> details must be configured.</li> </ul>

- Because **Pass Output File** is not required to be set to **Yes** if R Execute is a terminal operator, **Results File Structure** would not have to be provided, in case you set **Pass Output File** to **No**.

Default value: **No**.

**Results File Structure** Specifies the file structure of the operator's output to pass to the next operator (if **Pass Output File** is set to **Yes**).

For more information, see [Results File Structure dialog](#).

## Output

### Visual Output

The table is stored whether **Pass Output File** is set to **Yes**, and whether the **Results File Structure** is provided. The **Pass Output File** and **Results File Structure** combination is used only to check the integrity of the flow in case the R Execute operator is followed by another operator.

If the *alpine\_output* object does not exist in your R code, then the output is not generated. If you set **Pass Output File** to **Yes**, and if the R script does not contain a reference to the *alpine\_output* object, then the flow fails at runtime, and an error message is displayed for this inconsistency.

### Data Output

<b>Data</b> <a href="#">R-Console Output</a> <a href="#">R-Script</a>	play	wind	humidity	outlook	temperature
	no	false	85	sunny	85
	yes	false	96	rain	70

If you create an *alpine\_output* object in the R code, then the R Execute operator output displays the output data frame (persisted in the HDFS/MapR file structure) in the results console in the **Data** tab.

### R-Console Output

<b>Data</b> <a href="#">R-Console Output</a> <a href="#">R-Script</a>	outlook	temperature	humidity	wind	play
	overcast:4	Min. :64.00	Min. :65.00	false:8	no :5
	rain :5	1st Qu.:69.25	1st Qu.:71.25	true :6	yes:9
	sunny :5	Median :72.00	Median :80.00		
		Mean :73.57	Mean :80.29		
		3rd Qu.:78.75	3rd Qu.:88.75		
		Max. :85.00	Max. :96.00		

If your R code included functions that printed output to the R console, then the output is displayed in the results console in the **R-Console Output** tab.

**i Note:** The R Execute operator's console printing behavior is a bit different from the R console or RStudio behavior. Normally, if you have a statement in the R code that reads `summary(alpine_input)`, it is shown in the R console or RStudio console. However, because R Execute is capturing the console output into an object (which is executed in R using R's `capture.output` function), you must wrap such calls using the `print()` function. For example, instead of `summary(alpine_input)`, specify `print(summary(alpine_input))`. This is a limitation of how R's `capture.output` function works.

### R-Script

<p>Data</p> <p>R-Console Output</p> <p>R-Script</p>	<pre>print(summary(alpine_input)) alpine_output = alpine_input</pre>
---	--

Your R code is shown in the results console in the **R-Script** tab.

## SQL Execute

Executes a user-defined SQL clause.



### Information at a Glance

**i Note:** You can also use this operator in a workflow that uses TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Tools
Data source type	DB
Send output to other operators	Yes
Data processing tool	n/a

The SQL statement executed inside this operator does not have any dependency on its preceding or succeeding operator.

The connection only ensures its execution sequence within the analytic workflow.

The SQL Execute code can invoke any well-formed SQL statement, including stored procedures and in-database languages (for example, PL/R in PostgreSQL).

When used with a Google Big Query data source, the SQL Execute operator treats any line beginning with "-" or "#" to be a comment line and does not submit them for execution.

You can run temporary functions in Google Big Query using SQL Execute. For example:

```
Next line is expected to be the first query for udf definition to work.
CREATE TEMP FUNCTION multiplyInputs(x FLOAT64, y FLOAT64)
RETURNS FLOAT64
LANGUAGE js AS """
return x*y;
""";

WITH `datascience.credit` AS
(SELECT 1 AS x, 3 as y
UNION ALL
SELECT 4 AS x, 5 as y)
SELECT x, y, multiplyInputs(x, y) as product
FROM `datascience.credit`;
```

## Input

The SQL Execute operator does not require any input, but can be connected to a data set output.



## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Data Source</b>	<p>Select from the list of the data sources connected to the operator (optional).</p> <p>If not specified, the SQL Execute operator gets a database connection from the preceding operator. You can select a different connection for the operator, but this selection cannot be passed to a succeeding operator.</p>
<b>SQL Clause</b>	Click <b>Define Clause</b> to open the <a href="#">Define SQL Statement dialog</a> dialog and define the SQL to run.
<b>Pass Output Table</b>	<p>Specify whether to pass the SQL Execute output to the next operator.</p> <p>If <b>Yes</b>, the <b>Result Table Structure</b> details must be configured.</p> <p>Set this option to <b>Yes</b> to send the SQL Execute operator output to the next operator. If SQL Execute is a terminal operator, this setting is not required.</p> <p>Default value: <b>No</b>.</p>
<b>Result Table Structure</b>	<p>Displays the Edit Table Columns dialog, where you can specify the table structure of the operator's output to pass to the next operator.</p> <p>Specifying the table structure dynamically pulls the values from the <i>alpine_sql_output</i> variable in the SQL script. The controls that contain the values are not populated until you run the SQL script once.</p> <p>The button to specify the result table structure is enabled only if <b>Pass Output File</b> is set to <b>Yes</b>.</p> <p>See: <a href="#">Edit Table Columns dialog</a>.</p>
<b>Output Schema</b>	The schema for the output table or view.

Parameter	Description
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Drop If Exists</b>	<p>Specifies whether to overwrite an existing table.</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - If a table with the name exists, it is dropped before storing the results.</li> <li>• <b>No</b> - If a table with the name exists, the results window shows an error message.</li> </ul>

## Output

### Visual Output

None.

### Data Output

The results of the SQL statements.

## Sub-Flow

Allows you to incorporate another workflow within a parent workflow.



## Information at a Glance

Parameter	Description
Category	Tools

Parameter	Description
Data source type	HD
Send output to other operators	Yes
Data processing tool	Depends on customer-specific implementation

You can use a workflow as a set of operators that are executed as a sub-flow within another parent workflow. Doing this provides significant leverage of TIBCO Data Science - Team Studio for creating many related workflows with minimal additional effort.

**i Note:** As of version 6.x, there is a known issue with nested sub-flows for database data sources. A nested sub-flow occurs when a sub-flow operator in a workflow calls another workflow that contains a sub-flow operator. Nesting sub-flows causes the output names of the variables to collide, and your results might be incorrect or the workflow might fail.

## Input

Data sets from the preceding operator.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Sub-Flow</b>	<p>The workflow to use as a sub-flow. The options for <b>Sub-Flow</b> are the workflows in the same workspace as the current workflow, and they appear in a dropdown menu.</p> <p>Any workflow in the current folder can be used as a sub-flow except the recursive referenced one. If the recursive occurs, the system produces an error.</p>

Parameter	Description
<b>Input Table Mapping</b>	<p>The user can now attach the sub-flow to other operators.</p> <p>See: <a href="#">Input Table Mapping dialog</a>.</p>
<b>Exit Operator</b>	<p>The operator in the sub-flow to use as the point of attachment for any subsequent operators in the parent workflow. The options for <b>Exit Operator</b> are the operator's sub-flow workflow, and they appear in a dropdown menu.</p> <p>The exit operator should have the correct structure (for example, columnnames) for the subsequent operators. If the subsequent operator does not need an input, the exit operator can be empty.</p> <div> <p><b>Note:</b> For Hadoop flows, only operators with stored results can be used as exit operators.</p> </div>
<b>Sub-Flow Variable</b>	<p>Define the values of variables in the sub-flow workflow. The values you provide are used when running the sub-flow workflow in the context of the current (parent) workflow.</p> <p>See: <a href="#">Sub-Flow Variable dialog</a>.</p>

## Output

### Visual Output

All of the outputs of the operators inside the sub-flow with a node name that is added the sub-flow operator's name as a prefix.

## Apache Spark Specific Operators

The operators described in the following sections are compatible with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Introduced in TIBCO® Data Science - Team Studio version 7.1.0, these operators offer optimized Machine Learning modeling using the Spark engine.

## Data Operators

The Data Extraction (Load Data) operators are used to create connections to data sources for bringing data into the workflow.

### Dataset

This is a source operator that connects a database table or view. With this operator, you can incorporate data into the workflow.



### Information at a Glance

**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Load Data
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

The database table can subsequently be used in the following projects:

- A data-mining algorithm
- A prediction algorithm
- A statistical analysis

## Input

A Dataset operator does not take any input, since it is a source operator.

## Configuration

The following table provides the configuration details for the Dataset operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Data Source</b>	Specify the database connection for access to the database where the data set (Table or View) resides. The data source list is populated with the database data sources associated with the current workflow.
<b>Schema Name</b>	Specify the schema name of the data set (Table or View). The list of schemas is populated with the schemes found in the selected <b>Data Source</b> .
<b>Table Name</b>	Specify the name of the data set (Table or View). The list of tables is populated with the tables found in the selected <b>Schema Name</b> .

## Output

### Visual Output

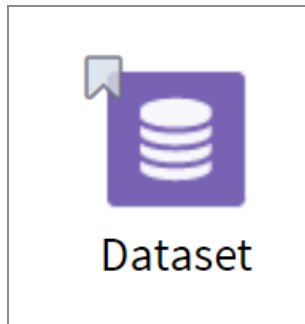
None.

### Data Output

A data set that corresponds to the specified database table or view.

## Example

The following example illustrates the Dataset operator that connects a database table from the TIBCO Data Virtualization data source. This data source is incorporated into the workflow that the downstream operators use.



## Data

None.

## Parameter Setting

The parameter settings for the **Dataset** operator are as follows:

- **Data Source:** TDV
- **Schema Name:** Datasets\_S3
- **Table Name:** golfnew.csv

## Output

The following figure displays the output for the parameter settings for the Dataset operator.

Operation created output with 5 columns and 14 rows.

outlook	temperature	humidity	wind	play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	78	false	yes
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
overcast	72	90	true	yes
rain	70	96	false	yes

## Import Excel

This operator imports an Excel workbook sheet (or a specified portion of the sheet) from the Chorus workspace as a database table. The operator uses TIBCO® Data Virtualization to infer the schema of the imported database table.



### Information at a Glance

**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Load Data
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

The Excel file from the workspace is read into memory. The data specified by the **Sheet Number**, **Top Left Corner Cell**, **Right Cut-Off Column Letter**, and **Bottom Cut-Off Row Number** parameters are extracted. If the **Has Header Row** parameter is set to **No**, then the operator prepends a row of column names with the **Col** prefix to the extracted data set. Then, the data is uploaded to TIBCO Data Virtualization in batches of 1,000 records. The TIBCO Data Virtualization infers the schema of a result database table based on the data values of each column. The resulting schema is then passed on to the downstream operators.

**Note:** You must **Step Run** or **Run** this operator before running a downstream operator.



## Input

The Import Excel operator is a source operator. Hence, there is no input data port.

### Bad or Missing Values

The blank cells or empty cells are converted to null values.

## Restrictions

The Excel files are read on the TIBCO Data Science - Team Studio server. Depending on the memory available on your instance, loading very large Excel files on this server might require a large amount of memory and can cause out-of-memory issues. For more information, see the [Apache POI limitations](#).

## Configuration

The following table provides the configuration details for the Import Excel operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Excel Workbook</b>	Select the Excel workbook stored in the current workspace. Only workbooks with the <b>.xls</b> , <b>.xlsx</b> , and <b>.xlsm</b> extensions can be selected. For more information on uploading an Excel file to the workspace, see <a href="#">Creating a Work File</a> .
<b>Sheet Number</b>	Specify the sheet number from the Excel workbook to extract (the first sheet is 1).  Default: <b>1</b>
<b>Top Left Corner Cell</b>	Specify the cell address that defines the top-left cell of the data portion to extract in the selected sheet (for example, <b>B10</b> ).
<b>Right Cut-Off Column</b>	Specify the column letter where the portion of data to extract should be cut at the right. If not specified, the data extraction is cut at the last defined cell

Parameter	Description
<b>Letter</b>	of the first row selected (the row number in the <b>Top Left Corner Cell</b> ). This parameter is optional.
<b>Bottom Cut-Off Row Number (0=not specified)</b>	<p>Specify the row number where the portion of data to extract should be cut at the bottom. If this is set to <b>0</b>, the data extraction is cut at the last defined row in the sheet. This parameter is optional.</p> <p><b>Note:</b> It is the row number indicated in the Excel worksheet and not the actual number of rows to be extracted.</p> <p>For example, if the <b>Top Left Corner Cell</b> is <b>A1</b> and <b>Has Header Row</b> is <b>Yes</b>, then the number of rows to be extracted is the <b>Bottom Cut-Off Row Number</b> - 1.</p>
<b>Has Header Row</b>	Specify whether the first row is treated as a header and each cell value of the row is used as the column name of the result table. If this is set to <b>No</b> , column names are set to Col1, Col2, Col3, and so on.
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

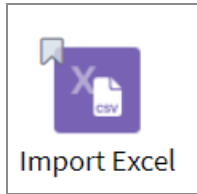
Displays the data extracted from the specified Excel worksheet.

### Output to successive operators

A single tabular data set extracted from the Excel sheet that can be used by a downstream operator. The column schema is generated after running this operator.

## Example

The following example demonstrates the Import Excel operator.



### Data

None.

### Parameter Setting

The parameter setting for the **Import Excel** operator is as follows:

- **Excel Workbook:** raw\_materials.xlsx

**Note:** The *raw\_materials.xlsx* file is available in the current workspace.

- **Sheet Number:** 1
- **Top Left Corner Cell:** A1
- **Bottom Cut-Off Row Number:** 0
- **Has Header Row:** Yes
- **Store Results:** Yes

### Results

The following figure displays the output for the parameter settings for the Import Excel operator.

### Output

Operation created output with 62 columns and 33 rows. Visualization of rows limited to 25. Please refer to output table for full results.

Campaign	Purpose	Hardness KP	Strengthmg	ModelSet	TargetTabletHardness	InitialTargetBCHMainCompression	InitialTargetFillDepth	InitialTargetMainCompressionForce
CM07517	PLS Test	17.2	100	Calibrat	17	5.05	9.38	17.1
CM10617	PLS Test	16	100	Test	17	4.8	9.5	24.65
CM10517	PLS Test	15.8	100	Test	17	4.85	9.4	22.233
CM04118	PLS Test	16.7	100	Calibrat	17	4.85	9.9	24.103
CM05318	Hybrid T	13.8	100	Calibrat	17	4.92	9.2	21.064
CM05418	PLS Test	15.8	100	Calibrat	17	4.9	9.52	23.506
CM05418	PLS Test	15.8	100	Test	17	4.9	9.52	23.506
CM06618	PLS Test	13.7	100	Calibrat	14	4.87	9.3	23.014
CM06718	Hybrid T	14.1	100	Calibrat	14	4.9	9.28	23.545
CM06818	Hybrid T	12.9	100	Calibrat	14	4.85	8.8	25.962
CM06918	Hybrid T	11.3	100	Calibrat	14	4.72	8.33	29.436
19-PN-00	PLS Test	10.8	100	Test	11	5.3	8.69	11.387
19-PN-00	PLS Test	11.2	100	Test	11	5.3	8.45	11.318
19-PN-00	PLS Test	11.1	100	Test	11	5.3	8.27	11.385
19-PN-00	Hybrid T	15.8	200	Calibrat	16	4.85	8.28	17.755

## Exploration Operators

The Exploration operators provide various methods to explore and visualize your data.

For successful modeling, it is crucial to understand the type of visualization that is suitable for your data. For more information about selecting the best visualization type, see [Visualizing data with charts and graphs](#).

## Correlation

This operator is used to specify two or more numeric type attributes (columns) in a data set for relative analysis against each other by calculating the correlation between each pair of selected columns.



## Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Explore
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The covariance between two variables ( $X$  and  $Y$ ) is calculated as given in the following formula:

$$\text{Cov}(X, Y) = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y}),$$

where  $\bar{X}$  and  $\bar{Y}$  are the mean values for  $X$  and  $Y$ , respectively.

The correlation is calculated by normalizing the covariance, as given in the following formula:

$$\text{Corr}(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Cov}(X, X)\text{Cov}(Y, Y)}} = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}}.$$

For information about correlation and covariance, and the algorithms that describe them, see [Correlation and Covariance](#).

## Input

An input is a single tabular data set.

**i Note:** The Correlation operator generates a column **Attribute** in its output. Hence, the input data set should not contain a column with the name **Attribute** or else it results in an error.

## Missing or Null Values

The selected columns should not have any null values.

## Restrictions

The algorithm is relevant only for numeric data.

## Configuration

The following table provides the configuration details for the Correlation operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	<p>Specify the numeric columns for which the correlation or covariance should be calculated. Click <b>Select Columns</b> to select the required columns.</p> <p><b>Note:</b> The input data set should not contain a column with the name <b>Attribute</b> or else it results in an error.</p>
<b>Group by</b>	<p>When you select one or more Group-by columns, the operator calculates a separate correlation (or covariance) matrix for every combination of values in the Group-by columns. You can select one or more columns. Click <b>Select Columns</b> to select the required columns.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• The <b>Group by</b> column selection cannot overlap with the column selected in <b>Columns</b> parameters.</li> <li>• There must be at least a minimum of two unique data points for a given combination of the <b>Group by</b> column and the <b>Attribute</b> column to generate a Correlation or Covariance value, or else it results in a NaN value.</li> </ul>
<b>Calculate</b>	<p>Specify whether to calculate the <b>Correlation</b> or the <b>Covariance</b>. Correlation is normalized covariance, scaled so that the correlation between any variable and a positive multiple of itself is always 1.</p> <p>Default: <b>Correlation</b></p>
<b>Output</b>	Specify the schema for the output table or view.

Parameter	Description
<b>Schema</b>	
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

Display correlation (or covariance) matrices for each combination of specified group-by values stacked into one output

**i Note:** If a **Group-by** parameter is not specified, only one matrix calculated from the full data is output.

### Data Output

The visual output is passed as its output to the downstream operator.

## Example

The following example illustrates the Correlation operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

## Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Columns:** temperature, humidity
- **Group By:** outlook
- **Calculate:** Covariance
- **Store Results:** Yes

## Results

The following figure displays the results for the parameter settings for the **golf** data set.

outlook	Attribute	temperature	humidity
overcast	temperature	76.6667	29.6667
overcast	humidity	29.6667	106
sunny	temperature	40.7	25.75
sunny	humidity	25.75	132.5
rain	temperature	13.7	12.8
rain	humidity	12.8	87.2

## Summary Statistics

The Summary Statistics operator loads a data set and calculates basic statistics for each of the selected columns, either over the entire data set or grouped by another set of columns.





## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Explore
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The Summary Statistics operator takes an input data set and performs basic statistical calculations.

For each selected column, this operator computes **Count**, **Distinct**, **Min**, **Max**, **Mean**, **number of positive values**, **number of negative values**, **number of zeroes**, **number of empty values**, **number of null values**, **Lower Quartile**, **Upper Quartile**, **Median**, **Standard Deviation**, **Coefficient of Variation**, and the **n** (where **n** is specified in input) **Most Common Value** and their counts.

When **Group By** columns are selected, these statistics are calculated for each unique value in each **Group By** column, and corresponding `Group_By_<col>` columns are added to the output data set.

## Input

An input is a single tabular data set.

### Missing or Null Values

Skips missing or null values while performing this operation or calculating the number of distinct values.

## Configuration

The following table provides the configuration details for the Summary Statistics operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	Specify the columns for which the summary statistics should be displayed. Click <b>Select Columns</b> to select the available columns from the input data set for analysis. The selected columns cannot be used in the <b>Group By</b> parameter box.
<b>Group By</b>	Specify the columns in the input data set to determine grouped results. Click <b>Select Columns</b> to select the available columns from the input data set for analysis.
<b>Calculate the Number of Distinct Values (slower)</b>	<p>Specify whether to calculate the number of distinct values for selected columns.</p> <p>Default: <b>Yes</b></p> <div> <b>Note:</b> Calculating distinct values can add significant processing time. </div>
<b>Number of Most Common Values to Display</b>	Specify the maximum number of the most common values to output for each column and the corresponding counts to the output.
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

- **Output:** A table that displays the analysis results of the selected fields, limited by the maximum display of rows and columns.

The default contents of a table are as follows:

- Group By column name
  - Column name
  - Data type
  - Count
  - Distinct value
  - Min value
  - Max value
  - Mean value
  - Positive value count
  - Negative value count
  - Zero value count
  - Null value count
  - Empty value count
  - Lower Quartile
  - Upper Quartile
  - Median (approx.) - Approximate median value for numerical columns.
  - Standard Deviation
  - Coefficient of Variation
  - Most Common Value - The most common value for the column.
  - Most Common Count - The most common count for the column.
- **Parameter Summary Info:** Displays information about the input parameters. A list of

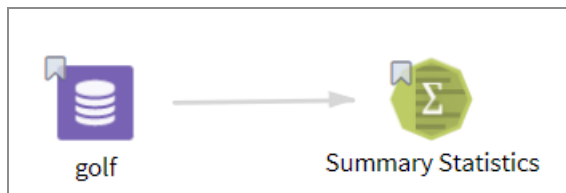
the input parameters and their current settings.

### Output to successive operators

A tabular data set containing a row per selected column and a combination of values of **Group By** columns (if selected). The columns represent the statistical measures calculated for each input column.

## Example

The following example demonstrates the Summary Statistics operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Columns:** outlook, temperature, humidity, play
- **Group By:** wind
- **Calculate the Number of Distinct Values (slower):** Yes
- **Number of Most Common Values to Display:** 3
- **Store Results:** Yes

### Results

The following figure displays the results for the parameter settings for the **golf** data set.

### Output

Group_By_wind	Column	Data Type	Count	Distinct	Min	Max	Mean	Positive	Negative	Zero	Nulls	Empty
true	outlook	String	6	3	N/A	N/A	N/A	N/A	N/A	N/A	0	0
true	temperature	Integer	6	6	64.000	80.000	71.167	6	0	0	0	0
true	humidity	Integer	6	4	65.000	90.000	77.500	6	0	0	0	0
true	play	String	6	2	N/A	N/A	N/A	N/A	N/A	N/A	0	0
false	outlook	String	8	3	N/A	N/A	N/A	N/A	N/A	N/A	0	0
false	temperature	Integer	8	8	68.000	85.000	75.375	8	0	0	0	0
false	humidity	Integer	8	7	70.000	96.000	82.375	8	0	0	0	0
false	play	String	8	2	N/A	N/A	N/A	N/A	N/A	N/A	0	0

## Parameter Summary Info

Name	Value
Columns	outlook, temperature, humidity, play
Group By	wind
Number of Most Common Values to Display	3
Calculate Number of Distinct Values (slower)	true

## Modeling Operators

The Modeling Algorithm (Model) operators define the modeling method or mathematical calculations for applying to an input data set.

TIBCO Data Science - Team Studio algorithms, or modeling approaches, use historical data from the input data to produce a predictive model, known as model training. Each Modeling operator is associated with a Predictor/Classifier operator and, together, they can be applied to other data sets within an analytic workflow in order to predict future results for that data set.

## Elastic-Net Linear Regression

The Elastic-Net Linear Regression operator applies the elastic-net linear regression algorithm to the input data set. This operator supports the open source implementation of the elastic-net regularized linear regression algorithm.



## Information at a Glance

**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The Elastic-Net Linear Regression operator fits a trend line to an observed data set where one of the data values (dependent variable) is linearly dependent on the value of the other causal data values or variables (independent variables). This operator implements the elastic-net linear regression in [Spark MLlib](#).

A Penalizing parameter (lambda) and the Elastic parameter (alpha) are applied to prevent the chances of overfitting the model. You can use this operator to optimize the best combination of alpha and lambda with a cross-validation method. This operator is limited by the cluster resources and Spark data frame size. One-hot encoding of high-cardinality categorical columns is limited by Spark cluster size.

## Input

An input is a single tabular data set.

### Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

The following table provides the configuration details for the Elastic-Net Linear Regression operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Variable</b>	Specify the categorical data columns as the dependent column. It must be numerical and the value cannot be a label or class.
<b>Use all available columns as Predictors</b>	When set to <b>Yes</b> , the operator uses all the available columns as predictors and ignores the Continuous Predictors and Categorical Predictors parameters. When set to <b>No</b> , the user must select at least one of the Continuous or Categorical Predictors.
<b>Continuous Predictors</b>	Specify the numerical data columns as independent columns. It must be numerical column. Click <b>Select Columns</b> to select the required columns.  <b>Note:</b> The columns selected in the <b>Categorical Predictors</b> parameter are not available.
<b>Categorical Predictors</b>	Specify the categorical data columns as independent columns.

Parameter	Description
	<p><b>Note:</b> The columns selected in the <b>Continuous Predictors</b> parameter are not available.</p>
<b>Normalize Numerical Features</b>	<p>Specify whether to normalize numerical features using Z-Transformation.</p> <p>Default: <b>Yes</b></p>
<b>Evaluation Metric</b>	<p>Specify the metric for evaluating the regression models such as <b>MAE</b>, <b>MSE</b>, <b>R2</b>, and <b>RMSE</b>.</p> <p>Default: <b>RMSE</b></p>
<b>Iterations</b>	<p>Specify the maximum number of iterations for each grid of parameters.</p> <p>Default: <b>100</b></p>
<b>Tolerance</b>	<p>Specify the convergence tolerance.</p> <p>Default: <b>0.01</b></p>
<b>Penalizing Parameter (<math>\lambda</math>)</b>	<p>The <math>\lambda</math> parameter grid for Lasso Regression. For more information, see <a href="#">Classification and Regression</a> in the Apache Spark documentation.</p> <p>The following values are valid:</p> <ul style="list-style-type: none"> <li>• A single value.</li> <li>• A comma-separated sequence of values (such as <math>V1</math>, <math>V2</math>, <math>V3</math>).</li> <li>• An interval notation following the pattern <code>start: end: count(n)</code>.</li> </ul> <p>Values of <math>\lambda</math> should span different orders of magnitude. In the case of <code>start: end: count(n)</code>, Team Studio creates an exponential grid of <math>n</math> <math>\lambda</math> values from <code>start</code> to <code>end</code>.</p>



Parameter	Description
	<ul style="list-style-type: none"> <li>• If <code>start &gt; end</code>, then "Not valid, start value of <math>\lambda</math> is greater than the end value" is returned.</li> <li>• If <code>count</code> is not an integer, then "Not valid; count should be an integer" is returned.</li> <li>• If <code>count &lt; 2</code>, then "Not valid; count should be at least 2" is returned.</li> </ul> <p>Default: <b>0.0, 0.5, 2</b></p>
<b>Elastic Parameter (<math>\alpha</math>)</b>	<p>The parameter to control the ElasticNet parameter.</p> <ul style="list-style-type: none"> <li>• When <math>\alpha = 0</math>, then the penalty is an L2 penalty.</li> <li>• When <math>\alpha = 1</math>, then the penalty is an L1 penalty.</li> </ul> <p>For more information, see <a href="#">Linear Methods - RDD-based API</a>.</p> <p>The following are valid values:</p> <ul style="list-style-type: none"> <li>• A single value.</li> <li>• A comma-separated sequence of values, such as <i>V1</i>, <i>V2</i>, and <i>V3</i>.</li> <li>• An interval notation following the pattern <code>start: end: step</code>.</li> </ul> <p>If <code>start &gt; end</code>, then "Not valid; start value of alpha is greater than the end value" is returned.</p> <p>If <code>step &gt; (end - start)</code>, then "Not valid; check the step value" is returned.</p> <p>Default: <b>0.0, 0.5, 0.1</b></p>
<b>Number of Cross Validation Folds</b>	<p>Specify the number of cross-validation samples.</p> <p>Default: <b>3</b></p>

Parameter	Description
<b>Random Seed</b>	Specify the seed used for the pseudo-random row extraction.  Default: <b>1</b>

## Output

### Visual Output

- **Parameter Summary Info:** Displays a list of the input parameters and their current settings.
- **Coefficients:** Displays the coefficient of the model.
- **Training Summary:** Displays a table with a row for each tested combination of hyper-parameters. For each hyper-parameter, the chosen metric is displayed and the Best Model is marked.
- **Objective History:** Displays a table showing the evolution of the objective function value during optimization. The objective function is the elastic-net objective function of linear regression with selected alpha and lambda:

$$L_{enet}(\hat{\beta}) = \frac{\sum_{i=1}^n (y_i - x_i^T \hat{\beta})^2}{2n} + \lambda \left( \frac{1 - \alpha}{2} \sum_{j=1}^m \hat{\beta}_j^2 + \alpha \sum_{j=1}^m |\hat{\beta}_j| \right)$$

For Spark ML elastic-net linear regression, the default optimization method is L-BFGS, a numerical optimization algorithm. The training process stops once the difference between two consecutive iterations is smaller than the user-specified tolerance.

In the case the exact solution of the objective function is obtained, the optimization method is the normal equation method. For example, when the value of lambda is 0, regularization is not applied to the objective function. In this case, the elastic-net linear regression is equivalent to the ordinary linear regression. An exact solution is obtained by the normal equation method. In another case, when alpha is 0, the exact solution is derivable. In this scenario, the elastic-net loss is equivalent to the ridge loss. The ridge loss is a convex function where a unique solution exists. In this case, the normal equation method is applied.

For more information, see [Optimization of linear methods](#).

### Output to successive operators

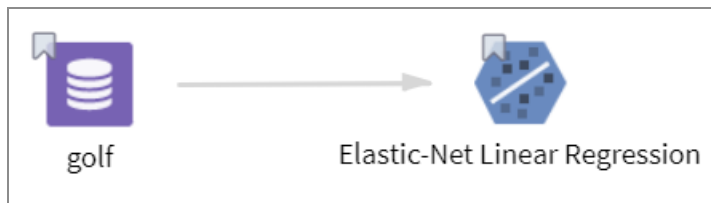
A regression model object that can be used with a [Predictor](#) operator. One column is produced in the [Predictor](#) operator.

- **PRED\_LR**: The value predicted by the regression model.

A regression model object that can also be used with a [Regression Evaluator](#) operator.

### Example

The following example demonstrates the Elastic-Net Linear Regression operator.



### Data

**golf**: This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Dependent Variable**: temperature
- **Use all available columns as Predictors**: No
- **Continuous Predictors**: humidity
- **Categorical Predictors**: outlook, play
- **Normalize Numerical Features**: Yes
- **Evaluation Metric**: RMSE
- **Iterations**: 100

- **Tolerance:** 0.01
- **Penalizing Parameter ( $\lambda$ ):** 0.0, 0.5, 2
- **Elastic Parameter ( $\alpha$ ):** 0.0, 0.5, 0.1
- **Number of Cross Validation Folds:** 3
- **Random Seed:** 1

## Results

The following figures display the results for the parameter settings for the **golf** data set.

### Parameter Summary Info

Name	Value
Dependent Variable	temperature
Continuous Predictors	humidity
Categorical Predictors	outlook,play
Normalize Numerical Features	Yes
Evaluation Metric	RMSE
Iterations	100
Tolerance	0.01
Penalizing Parameter( $\lambda$ )	0.0,0.5,2.0
Elastic Parameter( $\alpha$ )	0.0,0.5,0.1
Number of Cross Validation Folds	3
Random Seed	1

## Coefficients

Feature Name	Coefficients
Intercept	74.7558
humidity	0.7625
outlook_rain	-3.3163
outlook_sunny	0
play_yes	0

## Training Summary

Elastic Parameter( $\alpha$ )	Penalizing Parameter( $\lambda$ )	RMSE	Best Model
0	0	8.4619	
0	0.5	7.6429	
0	2	6.8308	
0.5	0	8.4619	
0.5	0.5	7.4126	
0.5	2	6.5584	
0.1	0	8.4619	
0.1	0.5	7.6018	
0.1	2	6.7362	

## Objective History

Iteration	Objective History
1	0.5
2	0.4911
3	0.4591
4	0.4577
5	0.457
6	0.457

## Elastic Net Logistic Regression

The Elastic-Net Logistic Regression operator applies the elastic-net logistic regression algorithm to the input data set. This operator supports the open source implementation of the elastic-net regularized logistic regression algorithm.



### Information at a Glance

**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

### Algorithm

This Elastic-Net Logistic Regression operator fits an s-curve logistic or logit function to a data set to calculate the probability of the occurrence of a specific categorical event, based on the values of a set of independent variables. This operator implements the logistic regression in Apache Spark 3.2 or later.

The logistic regression analysis predicts the odd outcomes of a categorical variable based on one or more predictor variables. This logistic regression operator implements the [Spark MLlib](#) open-source regularized logistic regression algorithm, optimized with L-BFGS for classification problems. This operator is used to optimize the hyper-parameters of logistic

regression with a cross-validation method. The output is the Spark Logistic Regression Classification model with the best validation performance.

## Input

An input is a single tabular data set.

### Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

The following table provides the configuration details for the Elastic-Net Logistic Regression operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Variable</b>	Specify the categorical data column as a dependent column. It must be numerical and the value cannot be a label or class.
<b>Use all available columns as Predictors</b>	When set to <b>Yes</b> , the operator uses all the available columns as predictors and ignores the Continuous Predictors and Categorical Predictors parameters. When set to <b>No</b> , the user must select at least one of the Continuous or Categorical Predictors.
<b>Continuous Predictors</b>	Specify the numerical data columns as independent columns. It must be numerical column. Click <b>Select Columns</b> to select the required columns.  <b>Note:</b> The columns selected in the <b>Categorical Predictors</b> parameter are not available.
<b>Categorical</b>	Specify the categorical data columns as independent

Parameter	Description
<b>Predictors</b>	<p>columns.</p> <p><b>Note:</b> The columns selected in the <b>Continuous Predictors</b> parameter are not available.</p>
<b>Normalize Numerical Features</b>	<p>Specify whether to normalize numerical features using Z-Transformation.</p> <p>Default: <b>Yes</b></p>
<b>Evaluation Metric</b>	<p>The metric for evaluating model performance during cross-validation training. For more information, see the <a href="#">Spark documentation on multinomial logistic regression</a>.</p> <p>The following values are:</p> <ul style="list-style-type: none"> <li>• <b>Auto</b></li> <li>• <b>FMeasure</b></li> <li>• <b>Accuracy</b></li> </ul> <p>If you select <b>Auto</b>, the operator uses <b>Accuracy</b> for binary classification and <b>FMeasure</b> for multiclass classification.</p> <p><b>Note:</b> The value of the beta parameter for <b>FMeasure</b> is set to 1.</p> <p>Default: <b>Auto</b></p>
<b>Iterations</b>	<p>Specify the maximum number of iterations for each grid of parameters.</p> <p>Default: <b>100</b>.</p>
<b>Tolerance</b>	<p>Specify the convergence tolerance.</p> <p>Default: <b>0.01</b></p>
<b>Penalizing Parameter (<math>\lambda</math>)</b>	<p>The <math>\lambda</math> parameter grid for Lasso Logistics Regression. For more information, see <a href="#">Multinomial logistic regression</a> in</p>



Parameter	Description
	<p>the Apache Spark documentation.</p> <p>The valid value is a comma-separated sequence of values, such as <i>V1</i>, <i>V2</i>, and <i>V3</i> representing <i>start</i>, <i>end</i>, and <i>count</i>. It is recommended that the values of <i>lambda</i> span different orders of magnitude. In the case of <i>start: end: count</i>, create an exponential grid of <i>n</i> <i>lambda</i> values from <i>start</i> to <i>end</i>.</p> <ul style="list-style-type: none"> <li>• If <i>start</i> &gt; <i>end</i>, then "Not valid, <i>start</i> value of <math>\lambda</math> is greater than the <i>end</i> value" is returned.</li> <li>• If <i>count</i> is not an integer, then "Not valid; <i>count</i> should be an integer" is returned.</li> <li>• If <i>count</i> &lt; 2, then "Not valid; <i>count</i> should be at least 2" is returned.</li> </ul> <p>Default: <b>0.0, 0.5, 1.0</b></p>
<b>Elastic Parameter (<math>\alpha</math>)</b>	<p>The parameter to control the ElasticNet parameter.</p> <ul style="list-style-type: none"> <li>• When <math>\alpha = 0</math>, then the penalty is an L2 penalty.</li> <li>• When <math>\alpha = 1</math>, then the penalty is an L1 penalty.</li> </ul> <p>For more information, see <a href="#">Linear Methods - RDD-based API</a>.</p> <p>The valid value is a comma-separated sequence of values, such as <i>V1</i>, <i>V2</i>, and <i>V3</i>, representing <i>start</i>, <i>end</i>, and <i>step</i>.</p> <p>If <i>start</i> &gt; <i>end</i>, then "Not valid; <i>start</i> value of <i>alpha</i> is greater than the <i>end</i> value" is returned.</p> <p>If <i>step</i> &gt; (<i>end</i> - <i>start</i>), then "Not valid; check the <i>step</i> value" is returned.</p> <p>Default: <b>0.0, 0.5, 1.0</b></p>
<b>Number of Cross</b>	Specify the number of cross-validation samples.

Parameter	Description
<b>Validation Folds</b>	Default: <b>3</b>
<b>Random Seed</b>	Specify the seed used for the pseudo-random row extraction.  Default: <b>1</b>

## Output

### Visual Output

- **Parameter Summary Info:** Displays a list of the input parameters and their current settings.
- **Coefficients:** For multiclass target, displays the coefficients for each value to predict and the reference class. For a binary classification task, displays the coefficients for value to predict (non-reference class.)
- **Training Summary:** Displays a table with a row for each tested combination of hyper-parameters. For each hyper-parameter, the chosen metric is displayed and the Best Model is marked.
- **Additional Model Info:** Displays the information of the levels within the dependent column and the reference categories of the logistic regression model.
- **Objective History:** Displays the objective function history during training. In our implementation, the objective function is Log Loss (negative Log Likelihood). For more information, see [Multinomial logistic regression](#) in the Apache Spark documentation.

### Output to successive operators

A classification model object that can be used with a [Predictor](#) operator. Three columns are produced in the [Predictor](#) operator.

- **PRED\_LOR:** The value predicted by the classification model.
- **CONF\_LOR:** The probability of the predicted classification.
- **INFO\_LOR:** The probability of each class prediction.

A classification model object that can also be used with a [Confusion Matrix](#) and [Goodness of Fit operator](#).

## Example

The following example demonstrates the Elastic-Net Logistic Regression operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Dependent Variable:** play
- **Use all available columns as Predictors:** No
- **Continuous Predictors:** temperature, humidity
- **Categorical Predictors:** wind
- **Normalize Numerical Features:** Yes
- **Evaluation Metric:** Auto
- **Iterations:** 100
- **Tolerance:** 0.01
- **Penalizing Parameter ( $\lambda$ ):** 0.0, 0.5, 0.2
- **Elastic Parameter ( $\alpha$ ):** 0.0, 0.5, 0.1
- **Number of Cross Validation Folds:** 3
- **Random Seed:** 1

### Results

These figures display the results for the parameter settings for the **golf** data set.

### Parameter Summary Info

Name	Value
Random Seed	1
Evaluation Metric	FMeasure
Normalize Numerical Features	true
Categorical Predictors	temperature, humidity
Continuous Predictors	wind
Tolerance	0.01
Penalizing Parameter( $\lambda$ )	0.0, 0.5, 1.0
Elastic Parameter( $\alpha$ )	0.0, 0.5, 1.0
Dependent Variable	play
Iterations	100
Number of Cross Validation Folds	3

### Coefficients

Feature Name	Model(no) Coefficient
wind_false	-2.7176
temperature	-0.0929
humidity	0.8748
Intercept	0.596

### Training Summary

Elastic Parameter( $\alpha$ )	Penalizing Parameter( $\lambda$ )	Accuracy	Best Model
2	0	0.5852	
0	0.1	0.7228	
0.5	0	0.5481	
0	0	0.7228	✓
0	0.5	0.7228	
0.5	0.5	0.5852	
0.5	0.1	0.5481	
2	0.1	0.5852	
2	0.5	0.5852	

### Additional Model Info

Categories: {yes,no}

Reference category: yes

### Objective History

Iteration	Objective History
3	0.4676
4	0.4645
5	0.4642
2	0.4897
1	0.6365

## Gradient-Boosted Tree Classification

This operator implements the Gradient-Boosted Tree Classification algorithm from [Spark ML](#).



## Information at a Glance

**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The Gradient-Boosted Tree algorithm is a predictive method, by which a series of shallow decision trees incrementally reduces prediction errors of previous trees. This operator implements the Gradient-Boosted Tree Classification algorithm from [Spark MLlib](#).

**Note:** Currently, this operator supports only binary classification.

## Input

An input is a single tabular data set.

### Bad or Missing Values

- Null values are not allowed and result in an error.
- The **Max Bins** parameter should be increased to the maximum cardinality of categorical features. However, depending on the available resources, the system might not be able to handle very high values result in an error.
- If the number of levels in the dependent column is not equal to 2, an error is reported.

## Configuration

Users can fine-tune the hyper-parameters of interest with the cross-validation training method and use the specified metric to evaluate the performance. The following table includes the configuration details for the Gradient-Boosted Tree Classification operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Variable</b>	Specify the categorical data column as a dependent column.
<b>Use all available columns as Predictors</b>	When set to <b>Yes</b> , the operator uses all the available columns as predictors and ignores the Continuous Predictors and Categorical Predictors parameters. When set to <b>No</b> , the user must select at least one of the Continuous or Categorical Predictors.
<b>Continuous Predictors</b>	Specify the numerical data columns as independent columns. It must be numerical column. Click <b>Select Columns</b> to select the required columns.  <b>Note:</b> The columns selected in the <b>Categorical Predictors</b> parameter are not available.
<b>Categorical Predictors</b>	Specify the categorical data columns as independent columns.  <b>Note:</b> The columns selected in the <b>Continuous Predictors</b> parameter are not available.

Parameter	Description
<b>Evaluation Metric</b>	<p>The metric for evaluating model performance during cross-validation training. For more information, see the <a href="#">Spark documentation on multinomial logistic regression</a>.</p> <p>The following values are available:</p> <ul style="list-style-type: none"> <li>• <b>Auto</b></li> <li>• <b>FMeasure</b></li> <li>• <b>Accuracy</b></li> </ul> <p>If <b>Auto</b>, then the operator uses <b>Accuracy</b>.</p> <p><b>Note:</b> The value of the beta parameter for <b>FMeasure</b> is set to 1.</p> <p>Default: <b>Auto</b></p>
<b>Number of Trees</b>	<p>A string specifying the number of trees. The input for this parameter should be a comma-separated sequence of integer values (for example, 10, 100.)</p> <p>Default: <b>100</b>.</p>
<b>Number of Feature Functions</b>	<p>A function to determine the number of features for building each decision tree. The following values are available:</p> <ul style="list-style-type: none"> <li>• <b>All</b></li> <li>• <math>\frac{1}{3}</math></li> <li>• <b>Square-Root</b></li> <li>• <b>Log2</b></li> <li>• <b>User Defined</b></li> </ul> <p>Default: <b>Square-Root</b></p>
<b>Feature Sampling Ratio</b>	<p>The fraction of the number of features per node to use when the <b>Number of Feature Functions</b> is set to <b>User Defined</b>. The input for this parameter should be a comma-separated sequence of double values in (0,1).</p> <p>Default: <b>0.5, 0.7</b>.</p>



Parameter	Description
<b>Max Depth</b>	<p>The maximum depth of each tree. The input for this parameter should be a comma-separated sequence of integer values.</p> <p>Default: <b>3, 5</b>.</p>
<b>Row Sampling Ratio</b>	<p>The fraction of training data for building each decision tree. The input for this parameter should be a comma-separated sequence of double values in (0,1).</p> <p>Default: <b>1</b></p>
<b>Min Leaf Size</b>	<p>The smallest number of data instances that exist within a terminal leaf node of a decision tree. The input for this parameter should be a comma-separated sequence of integer values (for example, 1,2).</p> <p>Default: <b>1</b></p>
<b>Max Bins</b>	<p>The maximum number of bins used for discretizing and splitting continuous features. The input for this parameter should be a comma-separated sequence of integer values (for example, 256). The number of <b>Max Bins</b> should be larger than the number of unique levels of any selected categorical columns.</p> <p>Default: <b>32</b></p>
<b>Learning Rate</b>	<p>The shrinkage parameter to control the contribution of each estimator. The input for this parameter should be a comma-separated sequence of double values in the interval (0,1).</p> <p>Default: <b>0.1</b></p>
<b>Number of Cross Validation Folds</b>	<p>The number of cross-validation samples.</p> <p>Default: <b>3</b></p>
<b>Random Seed</b>	<p>The seed used for the pseudo-random row extraction.</p> <p>Default: <b>1</b></p>

## Output

### Visual Output

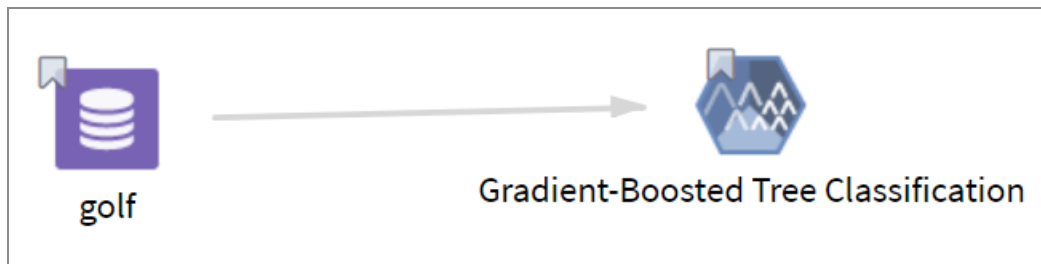
- **Parameter Summary Info:** Displays information about the input parameters and their current settings.
- **Variable Importance:** Displays the importance of predictors as evaluated in the training process. For each predictor, the importance of the model is shown in the second column.
- **Training Summary:** Displays a table with a row for each tested combination of hyper-parameters. For each hyper-parameter, the chosen metric is displayed and the Best Model is marked.

### Output to successive operators

A classification model object that can be used with a [Predictor](#), [Goodness of Fit](#), and [Confusion Matrix](#) operator.

## Example

The following example illustrates the Gradient-Boosted Classification operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Dependent Variable:** play
- **Use all available columns as Predictors:** Yes
- **Evaluation Metric:** Auto
- **Number of Trees:** 100
- **Number of Features Function:** Square Root
- **Feature Sampling Ratio:** 0.5, 0.7
- **Max Depth:** 3, 5
- **Row Sampling Ratio:** 1
- **Min Leaf Size:** 1
- **Max Bins:** 32
- **Learning Rate:** 0.1
- **Number of Cross Validation Folds:** 3
- **Random Seed:** 1

## Results

These figures display the results for the parameter settings for the **golf** data set.

## Parameter Summary Info

Name	Value
Dependent Variable	play
Continuous Predictors	temperature,humidity
Categorical Predictors	outlook,wind
Evaluation Metric	Accuracy
Number of Trees	100
Number of Features Functions	Square Root
Max Depth	3,5
Row Sampling Ratio	1.0
Min Leaf Size	1
Max Bins	32
Learning Rate	0.1
Number of Cross Validation Folds	3
Random Seed	1

## Variable Importance

Column Name	Variable Importance
outlook	0.3420
humidity	0.2439
temperature	0.2178
wind	0.1963

## Training Summary

Number of Features Functions	Max Bins	Max Depth	Number of Trees	Min Leaf Size	Learning Rate	Row Sampling Ratio	Accuracy	Best Model
Square Root	32	5	100	1	0.1000	1.0000	0.3500	
Square Root	32	3	100	1	0.1000	1.0000	0.5167	✓

## Gradient-Boosted Tree Regression

This operator implements the Gradient-Boosted Tree Regression algorithm from [Spark ML](#).



## Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The Gradient-Boosted Tree algorithm is a predictive method, by which a series of shallow regression trees incrementally reduces prediction errors of previous trees. This operator implements the open-source Gradient-Boosted Tree Regression (GBTR) algorithm from [Spark MLlib](#).

## Input

An input is a single tabular data set.

### Bad or Missing Values

- Null values are not allowed and result in an error.

- The **Max Bins** parameter should be increased to the maximum cardinality of categorical features. However, depending on the available resources, the system might not be able to handle very high values results in an error.
- If the number of levels in the dependent column is not equal to 2, an error is reported.

## Configuration

Users can fine-tune the hyper-parameters of interest with the cross-validation training method and utilize the chosen metric to evaluate the performance. The following table includes the configuration details for the Gradient-Boosted Tree Regression operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Variable</b>	Specify the categorical data column as a dependent column.
<b>Use all available columns as Predictors</b>	When set to <b>Yes</b> , the operator uses all the available columns as predictors and ignores the Continuous Predictors and Categorical Predictors parameters. When set to <b>No</b> , the user must select at least one of the Continuous or Categorical Predictors.
<b>Continuous Predictors</b>	Specify the numerical data columns as independent columns. It must be numerical column. Click <b>Select Columns</b> to select the required columns.  <b>Note:</b> The columns selected in the <b>Categorical Predictors</b> parameter are not available.
<b>Categorical Predictors</b>	Specify the categorical data columns as independent columns.  <b>Note:</b> The columns selected in the <b>Continuous Predictors</b> parameter are not available.
<b>Evaluation</b>	The metric for evaluating regression models. The following values are

Parameter	Description
<b>Metric</b>	<p>available:</p> <ul style="list-style-type: none"> <li>• <b>MAE</b></li> <li>• <b>MSE</b></li> <li>• <b>R2</b></li> <li>• <b>RMSE</b></li> </ul> <p>Default: <b>RMSE</b></p>
<b>Loss Function</b>	<p>The loss function to minimize. The following values are available:</p> <ul style="list-style-type: none"> <li>• <b>L1</b> (Absolute-value loss function)</li> <li>• <b>L2</b> (Squared-value loss function)</li> </ul> <p>Default: <b>L1</b></p>
<b>Number of Trees</b>	<p>A string specifying the number of trees. The input for this parameter should be a comma-separated sequence of integer values (for example, 10, 100.)</p> <p>Default: <b>100</b></p>
<b>Number of Feature Functions</b>	<p>A function to determine the number of features for building each decision tree. The following values are available:</p> <ul style="list-style-type: none"> <li>• <b>All</b></li> <li>• <math>\frac{1}{3}</math></li> <li>• <b>Square Root</b></li> <li>• <b>Log2</b></li> <li>• <b>User Defined</b></li> </ul> <p>Default: <b>Square Root</b></p>
<b>Feature Sampling Ratio</b>	<p>The fraction of the number of features per node to use when the <b>Number of Feature Functions</b> is set to <b>User Defined</b>. The input for this parameter should be a comma-separated sequence of double values in (0,1).</p> <p>Default: <b>0.5, 0.7</b></p>

Parameter	Description
<b>Max Depth</b>	<p>The maximum depth of each tree. The input for this parameter should be a comma-separated sequence of integer values.</p> <p>Default: <b>3, 5</b></p>
<b>Row Sampling Ratio</b>	<p>The fraction of training data for building each decision tree. The input for this parameter should be a comma-separated sequence of double values in (0,1).</p> <p>Default: <b>1</b></p>
<b>Min Leaf Size</b>	<p>The smallest number of data instances that can exist within a terminal leaf node of a decision tree. The input for this parameter should be a comma-separated sequence of integer values (for example, 1,2).</p> <p>Default: <b>1</b></p>
<b>Max Bins</b>	<p>The maximum number of bins used for discretizing and splitting continuous features. The input for this parameter should be a comma-separated sequence of integer values (for example, 256). The number of <b>Max Bins</b> should be larger than the number of unique levels of any selected categorical columns.</p> <p>Default: <b>32</b></p>
<b>Learning Rate</b>	<p>The shrinkage parameter to control the contribution of each estimator. The input for this parameter should be a comma-separated sequence of double values in the interval (0,1).</p> <p>Default: <b>0.1</b></p>
<b>Number of Cross Validation Folds</b>	<p>The number of cross-validation samples.</p> <p>Default: <b>3</b></p>
<b>Random Seed</b>	<p>The seed used for the pseudo-random row extraction.</p> <p>Default: <b>1</b></p>



## Output

### Visual Output

- **Parameter Summary Info** Displays information about the input parameters and their current settings.
- **Variable Importance** Displays the importance of predictors as evaluated in the training process. For each predictor, the importance of the model is shown in the second column.
- **Training Summary** Displays a table with a row for each tested combination of hyper-parameters. For each hyper-parameter, the chosen metric is displayed and the Best Model is marked.

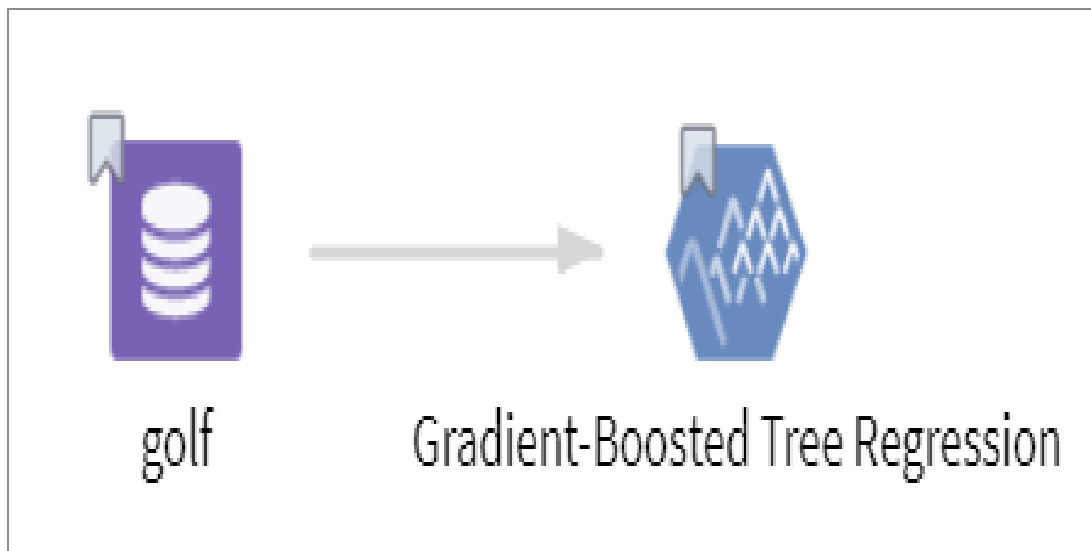
### Output to successive operators

The output of the operator is the model object with the best validation performance. You can use this regression model object with a compatible [Predictor](#) operator. One additional column, PRED\_GBTR (the predictive value of the regression model), is produced in the Predictor operator.

A regression model object that can also be used with a [Regression Evaluator](#) operator.

## Example

The following example illustrates a Gradient-Boosted Regression operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Dependent Variable:** humidity
- **Use all available columns as Predictors:** Yes
- **Evaluation Metric:** MAE
- **Loss Function:** L1
- **Number of Trees:** 100
- **Number of Features Function:** Square Root
- **Feature Sampling Ratio:** 0.5, 0.7
- **Max Depth:** 3, 5
- **Row Sampling Ratio:** 1
- **Min Leaf Size:** 1
- **Max Bins:** 32
- **Learning Rate:** 0.1
- **Number of Cross Validation Fold:** 3
- **Random Seed:** 1

### Results

These figures display the results for the parameter settings for the **golf** data set.

### Parameter Summary Info

Name	Value
Dependent Variable	humidity
Continuous Predictors	temperature
Categorical Predictors	outlook,wind,play
Evaluation Metric	MAE
Loss Function	L1
Number of Trees	100
Number of Features Functions	Square Root
Max Depth	3,5
Row Sampling Ratio	1.0
Min Leaf Size	1
Max Bins	32
Learning Rate	0.1
Number of Cross Validation Folds	3
Random Seed	1

## Variance Importance

Column Name	Variable Importance
temperature	0.5962
outlook	0.2239
wind	0.1065
play	0.0734

## Training Summary

Number of Features Functions	Loss Function	Max Bins	Max Depth	Number of Trees	Min Leaf Size	Learning Rate	Row Sampling Ratio	MAE	Best Model
Square Root	L1	32	3	100	1	0.1000	1.0000	11.5167	
Square Root	L1	32	5	100	1	0.1000	1.0000	10.6167	✓

## Isolation Forest

This operator applies the Isolation Forest unsupervised outlier detection algorithm to the input data set. The implementation of the Isolation Forest algorithm is provided by the open-source library from [LinkedIn](#).



### Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

### Algorithm

An Isolation Forest is an unsupervised learning algorithm for anomaly detection that isolates the potential anomalies in a data set. Isolation Forest is built on an ensemble of decision trees. The algorithm builds each tree with randomly selected features and samples. In principle, the most different observations are partitioned with fewer splits and are closer to the root. Thus, the path length is defined as the measure of normality, and the anomaly score returned by the algorithm is calculated with the function of inverse average path length over a forest of decision trees.

The columns specified are used to train the isolation anomaly detection model and the selected categorical columns are featured by a one-hot encoding algorithm.

## Input

An input is a single tabular data set.

### Columns containing Dates or Times

The input variables that contain date or date/time values should not be entered as string variables. The variables must be converted into numerical else they are ignored when using the Isolation Forest operator. The following methods can be used for converting the dates into numbers:

1. Generating an integer timestamp (For example, the number of seconds since 1 January 1970)
2. Extracting for example, day of month, month, year wherever appropriate

### Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

The following table provides the configuration details for the Isolation Forest operator.



**Note:** A column that contains unique value in each row should not be used as a Predictor.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Use all available columns as Predictors</b>	When set to <b>Yes</b> , the operator uses all the available columns as predictors and ignores the Continuous Predictors and Categorical Predictors parameters. When set to <b>No</b> , the user must select at least one of the Continuous or Categorical Predictors.

Parameter	Description
<b>Continuous Predictors</b>	<p>Specify the numerical data columns as independent columns. It must be a numerical column. Click <b>Select Columns</b> to select the required columns.</p> <p><b>Note:</b> The columns selected in the <b>Categorical Predictors</b> parameter are not available.</p>
<b>Categorical Predictors</b>	<p>Specify the categorical data columns as independent columns. Click <b>Select Columns</b> to select the required columns.</p> <p><b>Note:</b> The columns selected in the <b>Continuous Predictors</b> parameter are not available.</p>
<b>Number of Estimators</b>	<p>Specify the number of trees or estimators.</p> <p>Default: <b>100</b></p>
<b>Apply Bootstrap</b>	<p>Specify whether to sample each tree with replacement. If <b>Yes</b>, draw a sample for each tree with replacement. If <b>No</b>, do not sample with replacement.</p> <p>Default: <b>No</b></p>
<b>Fraction/ Number of Samples</b>	<p>Specify the number of samples used to train each tree. If the value is between <b>0.0</b> and <b>1.0</b>, it is treated as a fraction. If the value is more than <b>1.0</b>, it is treated as a count.</p> <p>Default: <b>1.0</b></p>
<b>Fraction/ Number of Features</b>	<p>Specify the number of features used to train each tree. If the value is between <b>0.0</b> and <b>1.0</b>, it is treated as a fraction. If the value is more than <b>1.0</b>, it is treated as a count.</p> <p>Default: <b>1.0</b></p>
<b>Contamination</b>	<p>Specify the fraction of outliers in the training data set. If the value is set to <b>0.0</b>, it speeds up the training and all predicted labels are false. The model and outlier scores are otherwise unaffected by this parameter.</p>

Parameter	Description
	Default: <b>0.1</b>
<b>Contamination Error (Advanced)</b>	The acceptable error when calculating the threshold required to achieve the specified contamination fraction. When the value is <b>0.0</b> , it forces an exact calculation of the threshold. The exact calculation is slow and can fail for large data sets. If there are issues with the exact calculation, a good choice for this parameter is often 1% of the specified contamination value.  Default: <b>1.0E-4</b>
<b>Random Seed</b>	Specify the seed used for the pseudo-random row extraction.  Default: <b>1</b>

## Output

### Visual Output

- **Parameters Summary Info:** Displays information about the input parameters and their current settings.
- **Training Summary:** Displays a table containing the data for normality count, anomaly count, and cutoff value.

The cutoff value is the outlier score threshold or the minimum score of the anomalous observations. The observations with outlier scores greater than the cutoff value are considered anomalous.

### Output to successive operators

A model object that can only be used with a [Predictor](#) operator. To perform the transformation against a data set, the Isolation Forest operator must be succeeded by a Predictor operator. The following additional columns are produced in the Predictor operator.

- **PRED\_ISF:** Specifies whether an observation is an anomaly. If the value is **1**, it is an anomaly and if the value is **0**, it is not an anomaly.
- **CONF\_ISF:** Returns the anomaly score.

A model object that cannot be used with any [Model Validation](#) operators.

## Example

The following example demonstrates the Isolation Forest operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Use all available columns as Predictors:** No
- **Continuous Predictors:** temperature
- **Categorical Predictors:** outlook, wind
- **Number of Estimators:** 100
- **Apply Bootstrap:** No
- **Fraction/ Number of Samples:** 1.0
- **Fraction/ Number of Features:** 1.0
- **Contamination:** 0.1
- **Contamination Error (Advanced):** 1.0E-4
- **Random Seed:** 1.0

### Results

These figures display the results for the parameter settings for the **golf** data set.

### Parameters Summary Info



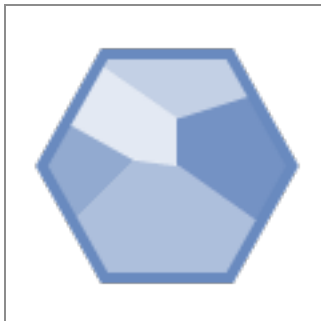
Name	Value
Categorical Predictors	outlook,wind
Contamination Error (Advanced)	1.0E-4
Continuous Predictors	temperature
Fraction/ Number of Samples	1.0
Apply Bootstrap	No
Contamination	0.1
Random Seed	1
Fraction/ Number of Features	1.0
Number of Estimators	100

### Training Summary

Name	Value
NormalityCount	12.0000
AnomalyCount	2.0000
CutoffValue	0.5966

## K-Means Clustering

This operator implements the K-Means clustering algorithm from [Spark MLlib](#).



## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The objective of the K-Means algorithm is to create clusters of objects that are similar to one another and different from individuals in other clusters in terms of their attributes. To achieve this, K-Means employs a centroid-based partitioning technique that uses the centroid of a cluster to represent that cluster. Conceptually, the center point is the centroid of a cluster.

The K-Means algorithm works as follows:

1. K points from the data set are chosen as the initial centroids of the K clusters according to the specified initialization method.
2. K clusters are created by associating each observation to the nearest centroid.
3. The new centroids are calculated for the clusters; determine whether centroid values change the coordinates.
4. Repeat steps 2 and 3 until convergence (when the centroid values do not change) or a specified termination criterion is met.

This operator implements the K-Means clustering algorithm from [Spark MLlib](#).

The specified columns are used to train the K-Means clustering model. You must define the initial centroids of the clusters. This operator provides two methods to define the initial centroids of the clusters such as [K-Means++](#) and Random allocation.

This operator uses the [Silhouette value](#) to determine the optimal number of clusters. The silhouette metric is a measure to compare the similarity of observation to its assigned cluster to other clusters. Generally, high silhouette values are preferred.

## Input

An input is a single tabular data set.

### Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Use all available columns as Predictors</b>	When set to <b>Yes</b> , the operator uses all the available columns as predictors and ignores the Continuous Predictors and Categorical Predictors parameters. When set to <b>No</b> , the user must select at least one of the Continuous or Categorical Predictors.
<b>Continuous Predictors</b>	Specify the numerical data columns for training the K-Means model. It must be numerical column. Click <b>Select Columns</b> to select the required columns.
<b>Distance Measure</b>	Specify the distance measure for training the K-Means model. The available options are <b>Euclidean</b> and <b>Cosine</b> .  Default: <b>Euclidean</b>
<b>Number of Clusters</b>	The number of clusters to create during the cluster analysis process. Specify the parameter using one of the following methods. <ul style="list-style-type: none"> <li>A single value, K1. For example, <b>2</b>. The number of clusters must be</li> </ul>

Parameter	Description
	<p>greater than 1.</p> <ul style="list-style-type: none"> <li>A comma-separated sequence K1, K2, K3, and so on. For example, <b>2,3,4,5</b>.</li> <li>A sequence specified by start:end:step. For example, <b>2:6:2</b>. This generates <b>K = 2,4,6</b>. The following conditions must be met or an error is displayed. <ul style="list-style-type: none"> <li>start must be less than end.</li> <li>step must be less than the result of end-start.</li> <li>start must be equal to or greater than 2.</li> </ul> </li> <li>A sequence specified by start:end. This generates a sequence with the step equal to <b>1</b>. For example, <b>2:6</b>. This generates <b>K = 2,3,4,5,6</b>.</li> </ul> <p>Default: <b>2</b></p>
<b>Initialization Method</b>	<p>The method for specifying the initialization cluster points. It can be either <b>K-Means++</b> or <b>Random</b>.</p> <p>Default: <b>K-Means++</b></p>
<b>Normalize Features</b>	<p>Specify whether to normalize numerical features (Z-Transformation).</p> <p>Default: <b>Yes</b></p>
<b>Max Iterations</b>	<p>Specify the maximum number of iterations performed for one run of the K-Means algorithm.</p> <p>Default: <b>100</b></p>
<b>Tolerance</b>	<p>The smaller the value, the stricter the determination of when the analysis has converged. A smaller number results in more iterations of the algorithm, but is still capped by the iteration limit.</p> <p>Default: <b>0.0001</b></p>
<b>Random Seed</b>	<p>Specify the seed used for the pseudo-random row extraction.</p> <p>Default: <b>1</b></p>

## Output

### Visual Output

- **Parameter Summary Info:** Displays a list of the input parameters and their current settings.
- **Training Summary:** A text field that displays the training summary.
  - **Silhouette:** A measure to compare the similarity of observation to its assigned cluster compared to other clusters.
  - **Training Cost:** The sum of specified distances to the nearest centroid for all points in the training data set.

### Output to successive operators

A model object that can be used with a [Predictor](#) operator. To perform the clustering against a data set, the K-Means Clustering operator must be succeeded by a [Predictor](#) operator. Two additional columns are produced in the [Predictor](#) operator.

- PRED\_KM: Specifies the cluster that an observation belongs to.
- DIST\_KM: The distance between the cluster centroid and the observation.

A model object that cannot be used with any [Model Validation](#) operators.

## Example

The following example builds a K-Means model and uses a [Predictor](#) operator to return the clustering result of a given data set.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

## Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Use all available columns as Predictors:**No
- **Continuous Predictors:** temperature, humidity
- **Distance Measure:** Euclidean
- **Number of Clusters:** 2,4,5
- **Initialization Method:** K-Means++
- **Normalize Features:** Yes
- **Max Iterations:** 100
- **Tolerance:** 1.0E-4
- **Random Seed:** 1

## Results

These figures display the results for the parameter settings for the **golf** data set.

## Parameter Summary Info

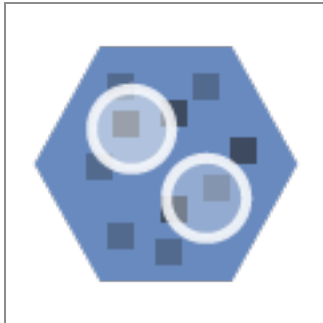
Name	Value
Continuous Predictors	temperature,humidity
Distance Measure	Euclidean
Number of Clusters	2,4,5
Initialization Method	K-Means++
Normalize Features	Yes
Max Iterations	100
Tolerance	1.0E-4
Random Seed	1

## Training Summary

Number of Clusters	Silhouette	Training Cost	Optimal K
2	0.5298	13.5783	✓
4	0.5117	4.9682	
5	0.5612	3.0611	

## Naive Bayes

The Naive Bayes operator calculates the probability of a particular event occurring. It is used to predict the probability of a certain data point being in a particular classification.



### Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The Naive Bayes classifier calculates the probability of an event occurring. It combines Bayes' theorem with an assumption of strong independence among the predictors. Bayes' theorem calculates the probability of occurrence given a prior event has occurred. Regardless of actuality, a Naive Bayes classifier considers the influence of predictors on the outcome independently.

- The TIBCO Data Science - Team Studio Naive Bayes Operator computes the dependent variable's class priors and each of the independent variable's probability distributions using the Naive Bayes conditional probability theorem with the independence assumption.
- As an overview, the Naive Bayes conditional probability theorem says that, given a data set (  $X$  ), and an outcome Hypothesis (  $H$  ), the posterior probability that the Hypothesis is true is proportional to the product of the likelihood multiplied by the prior probability.
- Depending on the precise nature of the probability model, the Naive Bayes classifiers can be trained very efficiently in a supervised learning setting.
- Given some data and some hypothesis, the posterior probability that the hypothesis is true is proportional to the product of the likelihood multiplied by the prior probability.
- For simplicity, the "prior probability" is often abbreviated as the "prior" and the "posterior probability" as the "posterior".
- The likelihood brings in the effect of the data, while the prior specifies the belief in the hypothesis before the data was observed.

More formally, Bayes' formula for conditional probability is represented as,

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

where,

- $P(H|X)$  is the conditional probability of outcome  $H$  happening given condition  $X$  ,
- $P(X|H)$  is the conditional probability of the outcome  $X$  happening given condition  $H$  ,
- $P(H)$  is the prior observed probability of the outcome  $H$  happening,



- $P(X)$  is the prior observed probability of the outcome  $X$  happening.

This Bayes formula is helpful because it provides a way to calculate the Posterior probability ( $P(H|X)$ ), from  $P(H)$ ,  $P(X|H)$ , and  $P(X)$  which can be calculated from historic data.

The Naive Bayes conditional independence assumption formula is as follows:

$$P(X|C_i) = \prod_{k=1}^n P(x_k|C_i)$$

If the feature is a continuous value, the conditional distribution over the class variable  $C$  is expressed as follows:

$$P(X|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-(x_k - \mu_{C_i})^2 / 2\sigma_{C_i}^2}$$

- This formula describes the ideal normal distribution curve for each independent variable's value.

**i Note:** This is a simplification assumption since most of the independent variables are likely to have exactly normal distributions.

- However, the Naive Bayes model predictions are still quite accurate with an acceptable level of confidence.
- The Naive Bayes Operator can accept a dependent column that has two or more discrete categories.

**i Note:** If the dependent variable is a numeric integer, each integer is treated as a separate category.

- The independence assumption treats all the predictors or variables as independently related to the outcome.
- The Naive Bayes theorem results give the normal probability curve of each possible categorical value occurring for that variable.

This operator implements the Naive Bayes algorithm from [Spark MLlib](#).

## Input

An input is a single tabular data set.

## Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

If the **Use all available columns as Predictors** parameter is set to **Yes**, the operator uses all available columns as predictors, or else the specified Continuous and Categorical predictors are used. It permits you to specify the event model type and lambda parameters. The following table includes the configuration details for the Naive Bayes operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Variable</b>	Specify the categorical data column as a dependent column.
<b>Use all available columns as Predictors</b>	When set to <b>Yes</b> , the operator uses all the available columns as predictors and ignores the Continuous Predictors and Categorical Predictors parameters. When set to <b>No</b> , the user must select at least one of the Continuous or Categorical Predictors.
<b>Continuous Predictors</b>	Specify the numerical data columns as independent columns. It must be numerical column. Click <b>Select Columns</b> to select the required columns.  <b>Note:</b> The columns selected in the <b>Categorical Predictors</b> parameter are not available.
<b>Categorical Predictors</b>	Specify the categorical data columns as independent columns.  <b>Note:</b> The columns selected in the <b>Continuous Predictors</b> parameter are not available.
<b>Model Type</b>	The event model type is supported by Naive Bayes. The following values are available:

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Multinomial</b></li> <li>• <b>Complement</b></li> <li>• <b>Bernoulli</b></li> <li>• <b>Gaussian</b></li> </ul> <p>Default: <b>Multinomial</b></p> <div> <p><b>Note:</b> The feature values for the Multinomial and Complement models must be non-negative (greater than or equal to 0) values.</p> <p>The feature value for the Bernoulli model must be either 0 or 1.</p> <p>For more information, see the <a href="#">Apache Spark documentation</a>.</p> </div>
<b>Lambda</b>	<p>Specify the additive smoothing parameter. The value must be non-negative (greater than or equal to 0).</p> <p>Default: <b>1.0</b></p>

## Output

### Visual Output

- **Parameter Summary Info:** Displays information about the input parameters and their current settings.
- **Training Summary:** Displays a table containing data for the dependent variable and for each of the categorical and continuous predictors. The dependent variable data represents the prior probability of each label.

For the Bernoulli, Complement, and Multinomial model types, the predictor data shows the conditional probability distribution of each predictor. For the Gaussian model type, the data represents the exponential (exp) of the mean value for each predictor.

### Output to successive operators

A classification model object that can be used with a [Predictor](#) operator. The additional three columns are produced in the Predictor operator:

- PRED\_NB: The predictive value of the classification model.
- CONF\_NB: The probability of the predicted value.
- INFO\_NB: Overall probabilities for each class.

A classification model object that can also be used with a [Confusion Matrix](#) and [Goodness of Fit operator](#).

## Example

The following example demonstrates the Naive Bayes operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Dependent Variable:** play
- **Use all available columns as Predictors:** No
- **Continuous Predictors:** temperature, humidity
- **Categorical Predictors:** outlook, wind
- **Model Type:** multinomial
- **Lambda:** 1.0

### Results

These figures displays the results for the parameter settings for the **golf** data set.

### Parameter Summary Info

Name	Value
Dependent Column	play
Continuous Predictors	temperature,humidity
Categorical Predictors	outlook,wind
Model Type	Multinomial
Smoothing	1.0

### Training Summary

Variable	play_yes	play_no
play	0.625	0.375
outlook	0.008	0.005
wind	0.0036	0.0037
temperature	0.4772	0.4663
humidity	0.5112	0.5249

## PCA

The Principal Component Analysis (PCA) operator generates an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of uncorrelated variables.



## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The PCA is an orthogonal linear transformation that transforms data into a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, the third on the third coordinate, continuing until the number of coordinates has been reached or a preset maximum principal component threshold has been reached.

This operator applies Center and Scale transformation to the selected columns, before generating the principal components. It also generates the full number of principal components.

## Input

An input is a single tabular data set.

### Bad or Missing Values

When null values are encountered in a column of a specific row, the entire row is removed before training a PCA model.

## Configuration

The following table provides the configuration details for the PCA operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Continuous Predictors</b>	Specify the numerical data columns as independent columns. It must be numerical column. Click <b>Select Columns</b> to select the required columns.
<b>Use all available columns as Predictors</b>	When set to <b>Yes</b> , the operator enable the wildcard feature. When set to <b>No</b> , users must select at least one of the Continuous Predictors.

## Output

### Visual Output

- **Components:** Displays the component matrix used for generating the principal components.
- **Variance:** Captures information on variance explained by each principal component (in descending order) alongside a cumulative total of variance explained.

### Output to successive operators

A model object that can be used with a [Predictor](#) operator. The PCA operator outputs the principal components and not the transformed data set. To perform the transformation against a data set, the PCA operator must be succeeded by a [Predictor](#) operator. This operator adds  $p$  transformed columns, where  $p$  is the number of selected columns. The transformations are then processed against the source training data set or a new input data set with the same variables.

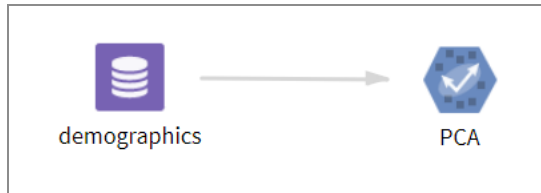
When the PCA operator is used with the [Predictor](#) operator in TIBCO Data Science - Team Studio, the number of components in the output is the same as the number of input columns. The users must review the **Variance** tab in the visual output to identify the amount of variance to capture and consequently the number of principal components to store. Based on this information, users can attach a [Dynamic Column](#)

[Filter](#) operator to the output of the [Predictor](#) operator to keep only the required variables.

A model object that cannot be used with any [Model Validation](#) operators.

## Example

The following example demonstrates the PCA operator.



### Data

**demographics:** This data set contains the following information:

- Sepal length
- Sepal width
- Petal length
- Petal width

### Parameter Setting

The parameter settings for the **demographics** data set are as follows:

- **Continuous Predictors:** sepal\_length,sepal\_width,petal\_length,petal\_width
- **Use all available columns as Predictors:** No

### Results

These figures display the results for the mentioned parameter settings for the **demographics** data set.

### Components



Component1	Component2	Component3	Component4
-0.4975	0.5199	0.6633	0.2055
0.2369	-0.0082	-0.1148	0.9647
-0.6966	-0.6985	-0.0252	0.1622
-0.4595	0.4917	-0.7391	0.0291

## Variance

Number	Variance	Cumulative variance
1	0.5163	0.5163
2	0.2393	0.7556
3	0.2086	0.9642
4	0.0358	1

## Random Forest Classification

This operator implements the Random Forest Classification algorithm from [Spark MLlib](#).



## Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The Random Forest Classification is an ensemble tree algorithm to the classification task to make a categorical prediction by averaging the numerical classification tree predictions of the ensemble. You can fine-tune the hyper-parameters of interest with the cross-validation training method. The operator uses the specified metric to evaluate the performance. The output of the operator is the model object with the best validation performance. This operator implements the Random Forest Classification algorithm from [Spark MLlib](#).

## Input

An input is a single tabular data set.

### Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

The following table provides the configuration details for the Random Forest Classification operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.

Parameter	Description
<b>Dependent Variable</b>	Specify the categorical data column as a dependent column. It must be numerical and the value cannot be a label or class.
<b>Use all available columns as Predictors</b>	When set to <b>Yes</b> , the operator uses all the available columns as predictors and ignores the Continuous Predictors and Categorical Predictors parameters. When set to <b>No</b> , the user must select at least one of the Continuous or Categorical Predictors.
<b>Continuous Predictors</b>	Specify the numerical data columns as independent columns. It must be numerical column. Click <b>Select Columns</b> to select the required columns.  <b>Note:</b> The columns selected in the <b>Categorical Predictors</b> parameter are not available.
<b>Categorical Predictors</b>	Specify the categorical data columns as independent columns.  <b>Note:</b> The columns selected in the <b>Continuous Predictors</b> parameter are not available.
<b>Impurity</b>	Specify the criterion for calculating the information gain when training the Random Forest model. The following values are available: <ul style="list-style-type: none"> <li>• <b>Gini</b></li> <li>• <b>Entropy</b></li> </ul> Default: <b>Gini</b>
<b>Evaluation Metric</b>	Specify the metric for evaluating model performance during cross-validation training. The following values are available: <ul style="list-style-type: none"> <li>• <b>Auto</b></li> <li>• <b>FMeasure</b></li> <li>• <b>Accuracy</b></li> </ul> For more information, see the <a href="#">Apache Spark documentation on Classification and Regression</a> .

Parameter	Description
	<p><b>Note:</b> The value of the beta parameter for <b>FMeasure</b> is set to 1.</p> <p>If the user selects <b>Auto</b>, then the operator uses <b>Accuracy</b>.</p> <p>Default: <b>Auto</b></p>
<b>Number of Feature Functions</b>	<p>Specify the function to determine the number of features for building each decision tree. The following values are available:</p> <ul style="list-style-type: none"> <li>• <b>All</b></li> <li>• <math>\frac{1}{3}</math></li> <li>• <b>Square Root</b></li> <li>• <b>log2</b></li> <li>• <b>User Defined</b></li> </ul> <p>Default: <b>Square Root</b></p>
<b>Feature Sampling Ratio</b>	<p>Specify the fraction of number of features per node to use when <b>Number of Feature Functions</b> is set to the <b>User Defined</b> option. The input for this parameter should be a comma-separated sequence of double values in <math>(0,1)</math>.</p> <p>Default: 0.5,0.7</p>
<b>Max Depth</b>	<p>Specify the maximum depth of each tree. The input for this parameter should be a comma-separated sequence of integer values.</p> <p>Default: 2,3</p>
<b>Number of Trees</b>	<p>Specify the total number of trees. The input for this parameter should be a comma-separated sequence of integer values.</p> <p>Default: 10,100</p>
<b>Row Sampling Ratio</b>	<p>Specify the fraction of training data for building each decision tree. The input for this parameter should be a comma-separated sequence of double values in <math>(0,1)</math>.</p>

Parameter	Description
	Default: 1
<b>Min Leaf Size</b>	Specify the smallest number of data instances that can exist within a terminal leaf node of a decision tree. The input for this parameter should be a comma-separated sequence of integer values (for example, 1,2).  Default: 1
<b>Max Bins</b>	Specify the maximum number of bins used for discretizing and splitting continuous features. The input for this parameter should be a comma-separated sequence of integer values (for example, 256).  <b>Note:</b> The number of Max Bins should be larger than the number of unique levels of any selected categorical columns.  <b>Max Bins</b> should be increased to the maximum cardinality of categorical features. However, depending on the available resources, the system might not be able to handle very high values and might cause an error.  Default: 32
<b>Number of Cross Validation Folds</b>	Specify the number of cross-validation samples.  Default: 3
<b>Random Seed</b>	The seed used for the pseudo-random generation.  Default: 1

## Output

### Visual Output

- **Parameter Summary Info:** Displays information about the input parameters and their current settings.
- **Variable Importance:** Displays the importance of predictors as evaluated in the training process. For each predictor, the variable importance for the model is displayed in the second column. This provides an indication of the importance or

impact of a particular parameter.

- **Training Summary:** Displays a table with a row for each tested combination of hyper-parameters. For each hyper-parameter, the chosen metric is displayed and the Best Model is marked. The information provides an insight into the parameters which resulted in the best model.

### Output to successive operators

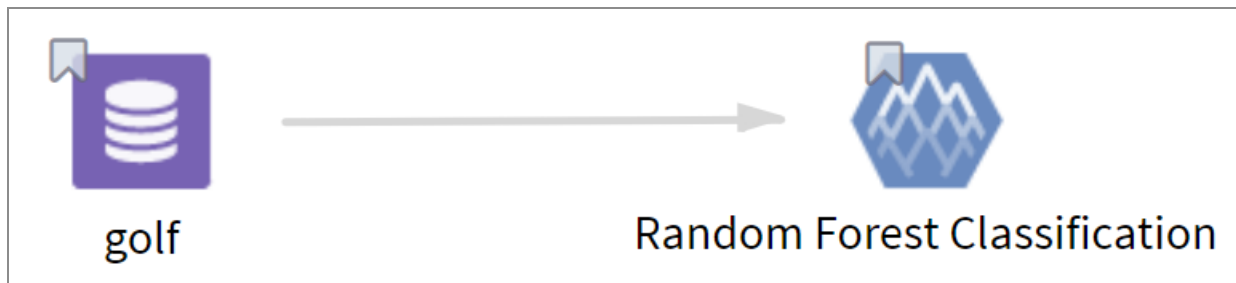
A classification model object that can be used with a [Predictor](#) operator. Three additional columns are produced in the [Predictor](#) operator:

- PRED\_RFC: The predictive value of the classification model.
- CONF\_RFC: The probability of the predicted value.
- INFO\_RFC: Overall probabilities for each class.

A classification model object that can also be used with a [Confusion Matrix](#) and [Goodness of Fit operator](#).

### Example

The following example demonstrates a Random Forest Classification operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Dependent Variable:** play

- **Use all available columns as Predictors:** No
- **Continuous Predictors:** temperature, humidity
- **Categorical Predictors:** outlook
- **Impurity:** Gini
- **Evaluation Metric:** Accuracy
- **Number of Feature Functions:** All
- **Feature Sampling Ratio:** 0.5, 0.7
- **Max Depth:** 2, 3
- **Number of Trees:** 10, 100
- **Row Sampling Ratio:** 1
- **Min Leaf Size:** 1
- **Max Bins:** 32
- **Number of Cross Validation Fold:** 3
- **Random Seed:** 1

## Results

These figures display the results for the parameter settings for the **golf** data set.

## Parameter Summary Info

Name	Value
Number of Feature Functions	all
Max Depth	2,3
Number of Cross Validation Folds	3
Random Seed	1
Continuous Predictors	temperature, humidity
Dependent Variable	play
Number of Trees	10,100
Row Sampling Ratio	1.0
Max Bins	32
Impurity	Gini
Evaluation Metric	Accuracy
Feature Sampling Ratio (0,1]	0.5,0.7
Min Leaf Size	1
Categorical Predictors	outlook

## Variable Importance

Column Name	Variable Importance
humidity	0.3841
temperature	0.3861
outlook	0.2298

## Training Summary

Number Of Feature Function	Max Bins	Max Depth	Min Leaf Size	Number Of Trees	Row Sampling Ratio	Accuracy	Best Model
all	32	2	1	10	1.0	0.3	
all	32	2	1	100	1.0	0.3667	✓
all	32	3	1	10	1.0	0.3	
all	32	3	1	100	1.0	0.3	



## Random Forest Regression

This operator implements the Random Forest Regression algorithm from [Spark MLlib](#).



### Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

### Algorithm

The Random Forest Regression is an ensemble tree algorithm that generates numerical predictions by averaging the numerical regression tree predictions of the ensemble. You can fine-tune the hyper-parameters of interest with the cross-validation training method. The operator uses the specified metric to evaluate the performance. The output of the operator is the model object with the best validation performance. This operator implements the Random Forest Regression algorithm from [Spark MLlib](#).

### Input

An input is a single tabular data set.

### Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

The following table provides the configuration details for the Random Forest Regression operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Variable</b>	Specify the data column as a dependent column. It must be a continuous numerical variable.
<b>Use all available columns as Predictors</b>	When set to <b>Yes</b> , the operator uses all the available columns as predictors and ignores the Continuous Predictors and Categorical Predictors parameters. When set to <b>No</b> , the user must select at least one of the Continuous or Categorical Predictors.
<b>Continuous Predictors</b>	Specify the numerical data columns as independent columns. It must be numerical column. Click <b>Select Columns</b> to select the required columns.  <b>Note:</b> The columns selected in the <b>Categorical Predictors</b> parameter are not available.
<b>Categorical Predictors</b>	Specify the categorical data columns as independent columns.  <b>Note:</b> The columns selected in the <b>Continuous Predictors</b> parameter are not available.
<b>Evaluation Metric</b>	Specify the metric for evaluating the regression models. The following values are available: <ul style="list-style-type: none"> <li>• <b>MSE</b></li> <li>• <b>MSE</b></li> <li>• <b>R2</b></li> <li>• <b>RMSE</b></li> </ul>

Parameter	Description
	<p>For more information, see the <a href="#">Apache Spark documentation on Classification and Regression</a>.</p> <p>Default: <b>RMSE</b></p>
<b>Number of Feature Functions</b>	<p>Specify the function to determine the number of features for building each decision tree. The following values are available:</p> <ul style="list-style-type: none"> <li>• <b>All</b></li> <li>• <math>\frac{1}{3}</math></li> <li>• <b>Square Root</b></li> <li>• <b>log2</b></li> <li>• <b>User Defined</b></li> </ul> <p>Default: <b>Square Root</b></p>
<b>Feature Sampling Ratio</b>	<p>Specify the fraction of a number of features per node to use when the <b>Number of Feature Functions</b> is set to the <b>User Defined</b> option. The input for this parameter should be a comma-separated sequence of numeric values in (0,1).</p> <p>Default: 0.5,0.7</p> <div> <p><b>Note:</b> If <b>User Defined</b> is not selected in the <b>Number of Feature Functions</b>, then this parameter is ignored.</p> </div>
<b>Max Depth</b>	<p>Specify the maximum depth of each tree. The input for this parameter should be a comma-separated sequence of integer values from 0 to 30.</p> <p>Default: 3,5</p>
<b>Number of Trees</b>	<p>Specify the total number of trees. The input for this parameter should be a comma-separated sequence of integer values.</p> <p>Default: 10,100</p>
<b>Row Sampling</b>	<p>Specify the fraction of training data for building each decision tree. The input for this parameter should be a comma-separated sequence of double values</p>

Parameter	Description
<b>Ratio</b>	in $(0,1)$ . Default: 1
<b>Min Leaf Size</b>	Specify the smallest number of data instances within a decision tree's terminal leaf node. The input for this parameter should be a comma-separated sequence of integer values (for example, 1,2). Default: 1
<b>Max Bins</b>	Specify the maximum number of bins used for discretizing and splitting continuous features. The input for this parameter should be a comma-separated sequence of integer values (for example, 256).  <b>Note:</b> The value of Max Bins should be larger than the number of unique levels of any selected categorical columns. Default: 32
<b>Number of Cross Validation Folds</b>	Specify the number of cross-validation samples. Default: 3
<b>Random Seed</b>	The seed used for the pseudo-random generation. Default: 1

## Output

### Visual Output

- **Parameter Summary Info:** Displays information about the input parameters and their current settings.
- **Variable Importance:** Displays the importance of predictors as evaluated in the training process. For each predictor, the variable importance for the model is displayed in the second column. This provides an indication of the importance or impact of a particular parameter on the model's predictions.

- **Training Summary:** Displays a table with a row for each tested combination of hyper-parameters. For each row, the chosen metric is displayed and the Best Model is marked. The information provides an insight into the parameters which resulted in the best model.

### Output to successive operators

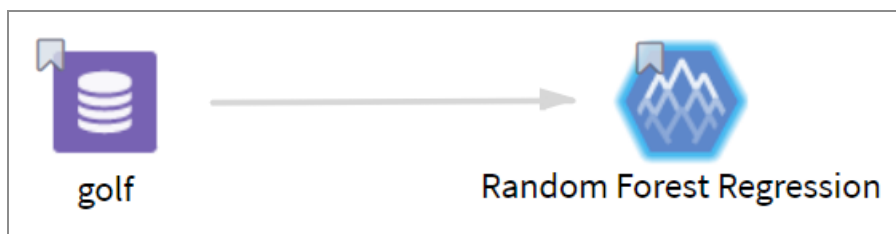
A regression model object that can be used with a [Predictor](#) operator. One additional column is produced in the [Predictor](#) operator:

- PRED\_RFR: The predictive value of the regression model.

A regression model object that can also be used with a [Regression Evaluator](#) operator.

### Example

The following example demonstrates a Random Forest Regression operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Dependent Variable:** temperature
- **Use all available columns as Predictors:** No
- **Continuous Predictors:** humidity
- **Categorical Predictors:** outlook, wind, play
- **Evaluation Metric:** RMSE
- **Number of Feature Functions:** Square Root

- **Feature Sampling ratio (0,1):** 0.5, 0.7
- **Max Depth:** 3, 5
- **Number of Trees:** 10, 100
- **Row Sampling Ratio:** 1
- **Min Leaf Size:** 1
- **Max Bins:** 32
- **Number of Cross Validation Fold:** 3
- **Random Seed:** 1

## Results

These figures displays the results for the parameter settings for the **golf** data set.

## Parameter Summary Info

Name	Value
Dependent Variable	temperature
Continuous Predictors	humidity
Categorical Predictors	outlook,wind,play
Evaluation Metric	RMSE
Number of Feature Functions	Square Root
Feature Sampling Ratio (0,1]	0.5,0.7
Max Depth	3,5
Number of Trees	10,100
Row Sampling Ratio	1.0
Min Leaf Size	1
Max Bins	32
Number of Cross Validation Folds	3
Random Seed	1

## Variable Importance

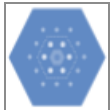
Column Name	Variable Importance
humidity	0.3706
outlook	0.3358
wind	0.1820
play	0.1116

## Training Summary

Number of Feature Functions	Max Bins	Max Depth	Min Leaf Size	Number of Trees	Row Sampling Ratio	RMSE	Best Model
Square Root	32	5	1	10	1.0000	8.1712	
Square Root	32	5	1	100	1.0000	8.2497	
Square Root	32	3	1	10	1.0000	8.1155	✓
Square Root	32	3	1	100	1.0000	8.2553	

## SOM Clustering

This operator implements the self-organizing maps algorithm that generates the clusters spatially aligned according to their similarity.



## Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model
Data source type	TIBCO® Data Virtualization

Parameter	Description
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The self-organizing map clustering algorithm is a simple neural network that produces clusters spatially aligned on a grid, where clusters close to each other are more similar to each other. The columns specified are used to train the self-organizing maps clustering algorithm.

This operator uses the [Silhouette value](#) to determine the optimal number of clusters. The silhouette metric is a measure of how similar observation is to its assigned cluster compared to other clusters. Generally, the higher silhouette values are preferred.

## Input

An input is a single tabular data set.

### Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

The following table provides the configuration details for the SOM Clustering operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Use all available columns as Predictors</b>	When set to <b>Yes</b> , the operator uses all the available columns as predictors and ignores the Continuous Predictors and Categorical Predictors parameters. When set to <b>No</b> , the user must select at least one of the Continuous or Categorical Predictors.



Parameter	Description
<b>Continuous Predictors</b>	Specify the numerical data columns for training the self-organizing maps model. It must be numerical column. Click <b>Select Columns</b> to select the required columns.
<b>Number of Clusters</b>	<p>Specify the grid dimensions, which control the number of clusters to create during self-organizing maps. The following methods are available:</p> <ul style="list-style-type: none"> <li>• A single value such as <b>K1</b>. The number of clusters must be greater than 1. For example, K1=2 is equal to 4 clusters (2x2) grid.</li> <li>• A comma-separated sequence such as <b>K1, K2, K3</b>, and so on. For example, K1=2, K2=3, and K3=4 are equal to 4 clusters (2x2) grid, 9 clusters (3x3) grid, and 16 clusters (4x4) grid respectively.</li> <li>• A sequence specified by <b>start:end:step</b>. The following conditions must be met or an error is displayed. <ul style="list-style-type: none"> <li>◦ start must be less than end.</li> <li>◦ step must be less than the result of end-start.</li> <li>◦ start must be equal to or greater than 2.</li> </ul> <p>For example, start:end:step=2:6:2 generates K1=2, K2=4, and K3=6 are equal to 4 clusters (2x2) grid, 16 clusters (4x4) grid, and 36 clusters (6x6) grid respectively.</p> </li> <li>• A sequence specified by <b>start:end</b>. This generates a sequence with the step equal to 1. For example, start:end=2:4 generates K1=2, K2=3, and K3=4 are equal to 4 clusters (2x2) grid, 9 clusters (3x3) grid, and 16 clusters (4x4) grid respectively.</li> </ul> <p>Default: <b>2</b></p>
<b>Normalize Features</b>	<p>Specify whether to normalize the numerical features (Z-Transformation).</p> <p>Default: <b>Yes</b></p>
<b>Max Iterations</b>	<p>Specify the maximum number of iterations for one run of the self-organizing map clustering algorithm (one iteration per sample).</p> <p>Default: <b>100</b></p>

Parameter	Description
<b>Tolerance</b>	The smaller the value, the stricter the determination of when the analysis has converged. A smaller number results in more iterations of the algorithm but are capped by the iteration limit.  Default: <b>1.0E-4</b>
<b>Random Seed</b>	Specify the seed used for the pseudo-random row extraction.  Default: <b>1</b>

## Output

### Visual Output

- **Parameter Summary Info:** Displays information about the input parameters and their current settings.
- **Training Summary:** A text field that displays the training summary.
  - Silhouette: A measure to compare the similarity of observation to its assigned cluster compared to other clusters.
  - Training Cost: The sum of specified distances to the nearest centroid for all points in the training data set.

### Output to Successive operator

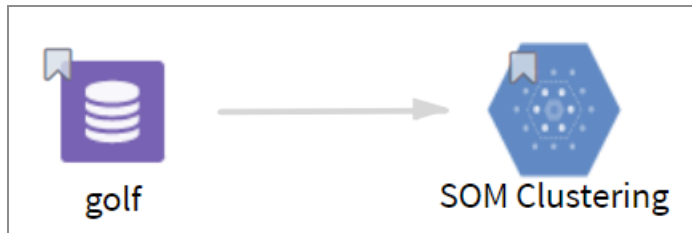
A model object that can be used with a [Predictor](#) operator. To perform the clustering against a data set, the SOM Clustering operator must be succeeded by a Predictor operator. Two additional columns are produced in the Predictor operator.

- PRED\_SOM: Specifies the cluster to that an observation belongs.
- DIST\_SOM: The distance between the cluster centroid and the observation.

A model object that cannot be used with any [Model Validation](#) operators.

## Example

The following example illustrates the SOM Clustering operator.



## Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

## Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Use all available columns as Predictors:** No
- **Continuous Predictors:** temperature, humidity
- **Number of Clusters:** 2
- **Normalize Features:** Yes
- **Max Iterations:** 100
- **Tolerance:** 1.0E-4
- **Random Seed:** 1

## Results

The following figure displays the output for the parameter settings for the **golf** data set.

## Parameter Summary Info

Name	Value
Continuous Predictors	temperature,humidity
Number of Clusters	2
Normalize Features	Yes
Max Iterations	100
Random Seed	1
Tolerance	1.0E-4

### Training Summary

Number of Clusters	Silhouette	Training Cost	Optimal K
2	0.255	10.7748	✓

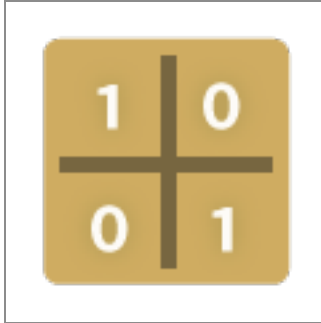
## Model Validation Operators

The Model Validation operators (Scoring operators) allow the modeler to assess whether a model is statistically valid and how well it is performing.

**i Note:** Find these operators in the **Predict** section of TIBCO Data Science - Team Studio.

## Confusion Matrix

This operator displays information about the actual versus predicted counts of a classification model and helps assess the model's accuracy for each of the possible class values.



## Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model Validation
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The Confusion Matrix operator is a classification model used to evaluate the accuracy of the predicted classifications of any classification modeling algorithm. For more information, see the [Confusion Matrix](#).

This operator takes one or more classification model objects and an input data set from upstream. It applies each of the model objects to the input data and computes the confusion matrix. The model performance is evaluated using the count of true positives, true negatives, false positives, and false negatives in a matrix.

## Input

An input is a single tabular data set and one or more model operators.

## Bad or Missing Values

- The operator accepts only the classification model.
- The operator does not accept more than one data set.
- Null values are not allowed and result in an error.
- Two types of input (tabular data and at least one model object) must be connected to this operator to prevent errors.
- The dependent variable should be in the input data set, or else the operator produces an error.

## Configuration

The following table provides the configuration details for the Confusion Matrix operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

The output displays confusion matrices for each upstream model operator. There is one tab with the Confusion Matrix for each upstream model operator. The visual outputs have a defined limitation on the number of rows to view. By default, this number is 25 and each matrix consumes  $N*(N+1)$  rows (where, N is the number of classes for the

classification model). In case, the output is larger (for example 5 models each with two classes:  $3 \times 2 \times 5 = 30 < 25$ ), you get a truncated result. In that case, change the value of **Data Preview Max Rows** to display. You can change the values from:

**Actions > Preferences > UI > Data Preview Max Rows**

**Note:** Rerun the Confusion Matrix operator after the values are changed.

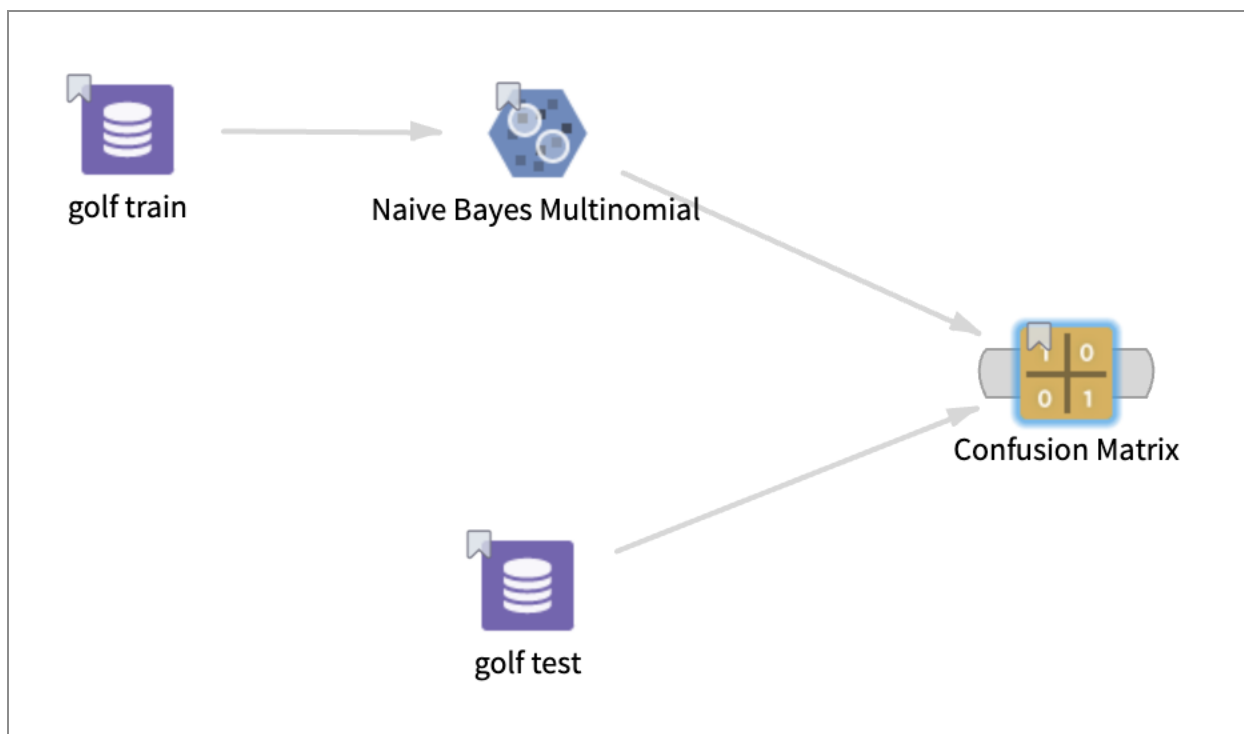
## Data Output

A data table that can be used by the downstream operator. The data table includes:

- Data in a list format.
- All the results in a single table.
- Results from all the trained models, where each trained model is separated by the **ModelIndex** column and identified by the **Node** column.

## Example

The following example uses the **golf train** data set to build the **Naive Bayes** model and then evaluates the model and **golf test** data set with the Confusion Matrix operator.



## Data

**golf train:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

**golf test:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

## Parameter Setting

The parameter settings for the given data set is as follows:

- **Store Results:** Yes

## Results

The following figure displays the results for the parameter settings for the given data set.

Node	Label	Predicted(yes)	Predicted(no)	Recall
Naïve Bayes	Observed(no)	5	0	0.0
Naïve Bayes	Precision	0.5833	0	Accuracy: 0.5
Naïve Bayes	Observed(yes)	7	2	0.7778

## Goodness of Fit

This operator computes the goodness-of-fit metrics for each class in a dependent column.





## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model Validation
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

This operator takes one or more classification model objects and one input data set from upstream. Then it applies each of the model objects to the input data and computes the Goodness of Fit metrics for each class, including Accuracy, Error, Recall, Precision, and FMeasure. This operator applies only to classification models.

## Input

An input is a single tabular data set and TIBCO Data Virtualization model operators.

### Bad or Missing Values

- Null values are not allowed and result in an error.
- A tabular data set and at least one model object should be connected to this operator; otherwise, the operator returns an error.
- The operator does not accept more than one data set.
- The dependent column must be available in the input data set; otherwise, the operator returns an error.
- The operator accepts only classification models.

## Configuration

The following table provides the configuration details for the Goodness of Fit operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

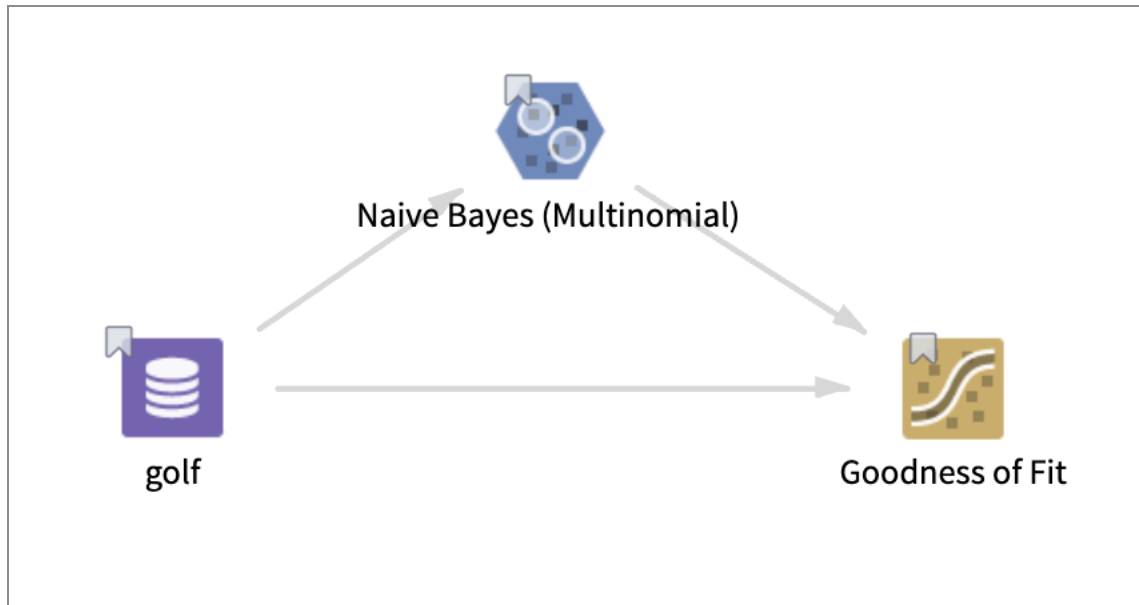
A table that displays the output of a data set.

### Output to successive operators

A database table output that can be used by the downstream operator.

## Example

The following example builds a [Naive Bayes](#) model, and then evaluates the model with the Goodness of Fit operator.



## Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

## Parameter Setting

The parameter settings for the **golf** data set is as follows:

- **Store Results:** Yes

## Output

The following figure displays the output for the parameter settings for the **golf** data set.

Node	Label	Accuracy	Error	Recall	Precision	FMeasure
Naive Bayes (Multinomial)	sunny	0.5	0.5	0.6	0.6	0.6
Naive Bayes (Multinomial)	overcast	0.5	0.5	0.4444	1	0.6154

## Regression Evaluator

This operator calculates several metrics for evaluating the regression models.



## Information at a Glance

**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Model Validation
Data source type	TIBCO® Data Virtualization
Send output to other operators	No
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

This operator computes several commonly used statistical tests to determine the accuracy of selected columns that contain predicted values. This operator implements the regression evaluator in [Spark MLlib](#).

For model validation, the operator uses the Spark ML regression evaluator. You can use the model with the following evaluation metric.

Metric	Description	Equation
Mean Squared Error (MSE)	<p>Sum of the squared differences between the actual and predicted columns, divided by the number of observations in the data set.</p> <p>A value of <b>0</b> indicates that the predicted and actual values are exactly the same for each observation. A very high value indicates that the difference between actual and predicted values is very large.</p>	$MSE = \frac{1}{N} \sum_{i=1}^N (Y_i - \hat{Y}_i)^2$
Root Mean Squared Error (RMSE)	<p>Square root of the Mean Squared Error (MSE) metric.</p>	$RMSE = \sqrt{\frac{\sum_{i=1}^N (Y_i - \hat{Y}_i)^2}{N}}$
Mean Absolute Error (MAE)	<p>Average of the absolute difference between the predicted and actual columns for each observation.</p> <p>A value of <b>0</b> indicates that the predicted and actual values are exactly the same for each observation. A very high value indicates that the difference between actual and predicted values is very large.</p>	$MAE = \frac{1}{N} \sum_{i=1}^N  Y_i - \hat{Y}_i $
Coefficient of Determination (R <sup>2</sup> )	<p>The proportion of the variance in the dependent variable that is predictable from the independent variables.</p> <p>A value of <b>1</b> indicates that the regression line perfectly fits the data, while a value of <b>0</b> indicates that it doesn't fit the data at all. When a value is negative, it indicates that the horizontal line is better than the regression mode and does not capture the trend of the data.</p>	$R^2 = 1 - \frac{RSS}{TSS}$ <p>Where</p> <ul style="list-style-type: none"> <li><b>RSS</b> = sum of squares of residuals</li> <li><b>TSS</b> = total sum of squares</li> </ul>
Explained Variance	<p>Returns the variance explained by the regression. For more information, see the <a href="#">Spark documentation</a>.</p>	$1 - \frac{VAR(y - \hat{y})}{VAR(y)}$

## Input

An input is a single tabular data set and one or more TIBCO Data Virtualization model operators.

### Bad or Missing Values

- The operator accepts only the classification model.
- The operator does not accept more than one data set.
- Null values are not allowed and result in an error.
- Two types of input (tabular data and at least one model object) must be connected to this operator to prevent errors.
- The dependent variable should be in the input data set, or else the operator produces an error.

## Configuration

The following table provides the configuration details for the Regression Evaluator operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Actual Value</b>	Specify a numerical column that holds the dependent variable on which the models were used to train or a column of known values for the dependent variable. The column must be a numerical type column.
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.

Parameter	Description
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

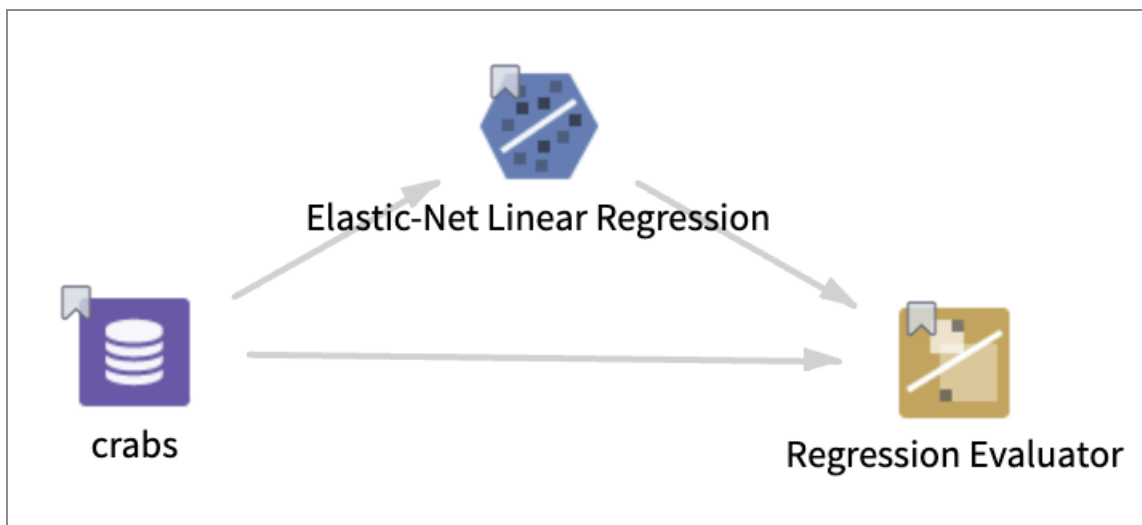
Displays the performance of regression models. The Model column displays the name of the upstream operator. This enables you to differentiate between multiple instances of the same model.

### Output to successive operators

A data table with model performances.

## Example

The following example uses the **crabs** data set to build the [Elastic-Net Linear Regression](#) model and then evaluates the model and **crabs** data set with the Regression Evaluator operator.



### Data

**crabs:** This data set contains the following information:

- Multiple columns such as color, spine, width, satelIts, weight, and catwidth.
- Multiple rows (173 rows)

### Parameter Setting

The parameter setting for the **crabs** data set are as follows:

- **Actual Value:** satelIts
- **Store Results:** Yes

### Output

The following figure displays the results for the parameter settings for the **crabs** data set.

Model	Root Mean Squared Error	Mean Squared Error	R Squared	Mean Absolute Error	Explained Variance
Elastic-Net Linear Regression	2.229	4.9686	0.4958	1.4866	3.7162

## Prediction Operators

The Prediction (Predict) operators are used to apply a particular modeling algorithm operator to a new data set for prediction purposes.

### Predictor

This operator applies an input model (for example, regression, classification, or clustering) to an input data set in order to predict a target value.



### Information at a Glance

**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.



Parameter	Description
Category	Predict
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The Predictor operator is used to generate predictions based on the model(s) developed from the input model operator(s).

Input Model	What the Predictor Calculates
Classification algorithms	Class with the highest probability
Numeric regression algorithms	Predicted value
Clustering algorithms	Predicted cluster
Anomaly detection algorithms	Anomaly class
PCA	Principal components

This operator takes one or more model objects and an input data set from upstream. Then it applies each model object to the input data and returns the prediction. Depending on the model types, the Predictor operator generates different prediction columns. For each additional input model, an index number is added to generated column names to separate them.

The operator includes the following models and prediction columns in the user-specified output table.

Model Type	Model	Model Abbreviation (key)	Prediction Columns
Classification	Naive Bayes	NB	<ul style="list-style-type: none"> <li>• <b>PRED_&lt;key&gt;</b>: The value predicted by the classification model (returns the most probable class).</li> <li>• <b>CONF_&lt;key&gt;</b>: The probability of the predicted classification.</li> <li>• <b>INFO_&lt;key&gt;</b>: The probability of each class prediction.</li> </ul>
	Elastic-Net Logistic Regression	LOR	
	Random Forest Classification	RFC	
	Gradient-Boosted Classification	GBTC	
Regression	Elastic-Net Linear Regression	LR	<b>PRED_&lt;key&gt;</b> : The value predicted by the regression model.
	Random Forest Regression	RFR	
	Gradient-Boosted Regression	GBR	
Clustering	K-Means Clustering	KM	<ul style="list-style-type: none"> <li>• <b>PRED_KM</b>: The value of the predicted cluster.</li> <li>• <b>DIST_KM</b>: The distance between the cluster centroid and the observation.</li> </ul>
Principal Component Analysis	Principal Component Analysis	PCA	<b>y_i_PCA</b> : The i <sup>th</sup> number of principal components (starting from zero).
Anomaly	Isolation Forest	ISF	<ul style="list-style-type: none"> <li>• <b>PRED_ISF</b>: To specify whether</li> </ul>

Model Type	Model	Model Abbreviation (key)	Prediction Columns
Detection			<p>an observation is an anomaly or not. By default, <b>1</b> is an anomaly, and <b>0</b> is not an anomaly.</p> <ul style="list-style-type: none"> <li>• <b>CONF_ISF</b>: The anomaly score returned.</li> </ul>

## Input

One or more input TIBCO Data Virtualization modeling operators (for example, regression, classification, or clustering) and one input data set against which the models are applied.

This operator is limited by the cluster resources and Spark data frame size.

### Bad or Missing Values

- Null values are not allowed and result in an error.
- If the input column names do not match the column names in the data set selected for model training, an error is reported.
- Input data, tabular data, and at least one model object must be connected to this operator, or else results in an error.
- The dependent variable should be in the input data set, or else the operator produces an error.

## Configuration

The following table provides the configuration details for the Predictor operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.

Parameter	Description
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

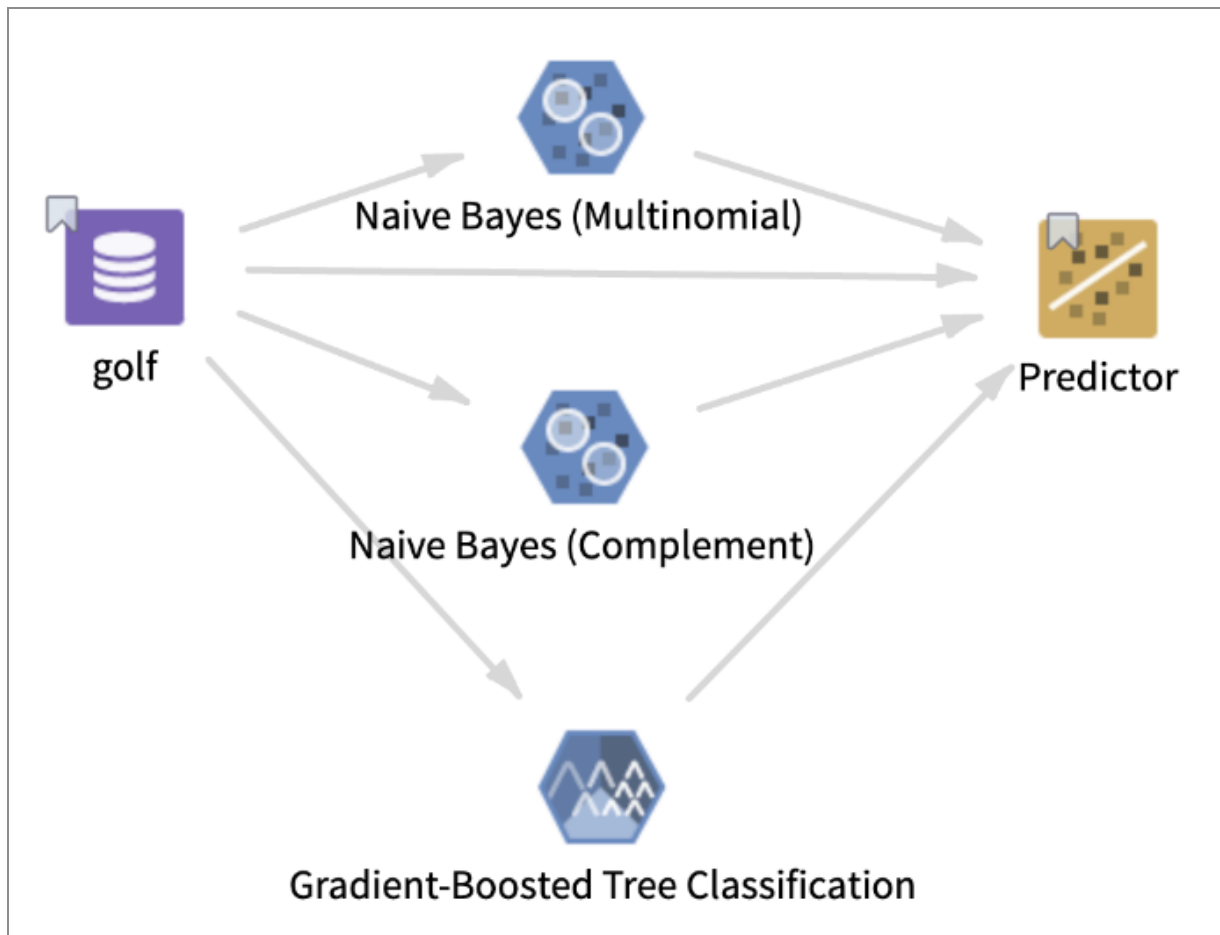
- **Output:** Displays a table of the predicted data set.
- **Summary:** Displays a list of the TIBCO DV modeling operators and their selected columns.

### Output to successive operators

A table output that can be used by the downstream operator.

## Example

The following example builds a [Naive Bayes](#) model and a [Gradient-Boosted Tree Classification](#) model, then combines the models with the Predictor operator.



## Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

## Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Store Results:** Yes

## Results

These figures displays the results for the parameter settings for the **golf** data set.

## Output

outlook	temperature	humidity	wind	play	PRED_NB	CONF_NB	INFO_NB	PRED_NB_1	CONF_NB_1	INFO_NB_1	PRED_GBTC	CONF_GBTC	INFO_GBTC
sunny	85	85	f	no	yes	0.6575	("yes":0.6574710	yes	0.5352	("yes":0.5352461	no	0.9869	("yes":0.0131191
sunny	80	90	t	no	yes	0.6074	("yes":0.6073778	no	0.5186	("yes":0.4813780	no	0.9837	("yes":0.0162614
overcast	83	78	t	yes	yes	0.7845	("yes":0.7844879	yes	0.6859	("yes":0.6859362	yes	0.9887	("yes":0.9887208
overcast	64	65	t	yes	yes	0.7685	("yes":0.76849,"n	yes	0.6657	("yes":0.6657398	yes	0.9877	("yes":0.987718,"
sunny	72	95	f	no	yes	0.5223	("yes":0.5223279	no	0.6038	("yes":0.3961688	no	0.9807	("yes":0.0193423
sunny	69	70	t	yes	yes	0.671	("yes":0.6710034	yes	0.5503	("yes":0.5503045	yes	0.9777	("yes":0.9777028
overcast	72	90	t	yes	yes	0.6731	("yes":0.6731238	yes	0.5527	("yes":0.5526843	yes	0.9859	("yes":0.9859202
rain	70	96	f	yes	no	0.6112	("yes":0.3887540	no	0.7238	("yes":0.2762023	yes	0.9872	("yes":0.9872263
rain	68	80	f	yes	no	0.5188	("yes":0.4811601	no	0.6425	("yes":0.3575024	yes	0.9791	("yes":0.9790748
rain	65	70	t	no	yes	0.5378	("yes":0.5377669	no	0.5889	("yes":0.4110879	no	0.9794	("yes":0.0206025
rain	75	80	f	yes	yes	0.5213	("yes":0.5212900	no	0.6048	("yes":0.3951741	yes	0.978	("yes":0.9780218
rain	71	80	t	no	yes	0.5062	("yes":0.5061536	no	0.6192	("yes":0.3807868	no	0.9683	("yes":0.0316503
sunnnv	75	70	t	yes	yes	0.7006	("yes":0.7006489	yes	0.5841	("yes":0.5840847	yes	0.9788	("yes":0.9788105

## Summary

Operator	Columns
Naive Bayes (Multinomial)	PRED_NB,CONF_NB,INFO_NB
Naive Bayes (Complement)	PRED_NB_1,CONF_NB_1,INFO_NB_1
Gradient-Boosted Tree Classification	PRED_GBTC,CONF_GBTC,INFO_GBTC

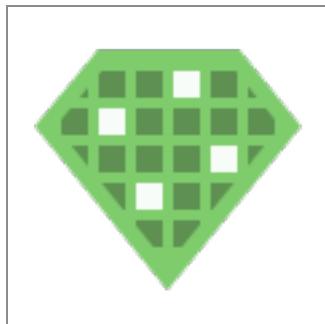
## Sampling Operators

The Sampling (Sample) operators provide ways to obtain a sample of a source data set.

A model is typically created using a training data set and then tested against a validation data set. This is achieved in the TIBCO Data Science - Team Studio by sampling the source data.

## Random Sampling

This operator extracts data rows from the input data set and generates sample tables or views according to the sample properties (percentage or row count) that the user specifies.



## Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Sample
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Input

An input is a single tabular data set.

### Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Number of Samples</b>	Specify the number of samples to generate. The samples are in the form of either database tables or views. For example, if the user inputs <b>3</b> in this field, 3 sample tables or views are generated.
<b>Sample By</b>	Specify the size of samples. The following values are available: <ul style="list-style-type: none"><li>• <b>Percentage</b></li></ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Number of Rows</b></li> </ul>
<b>Sample Size</b>	<p>Specify the number of rows to generate for each sample data set. This property is interpreted in conjunction with the <b>Sample By</b> property.</p> <ul style="list-style-type: none"> <li>• <b>Percentage</b> - Specify the number of rows to include in the total sample as a percentage of the number of rows in the input data set. For example, if the user inputs 20%, 30%, and 40% for three samples and the input data set contain 10,000 rows, each sample data set contains 2000, 3000, 4000 rows, and 9,000 rows are selected in total.</li> </ul> <p>If the <b>Disjoint</b> property is <b>true</b>, the total aggregate percentage should be less than 100%.</p> <ul style="list-style-type: none"> <li>• <b>Row</b> - Specify the exact number of rows to include in each sample data set.</li> </ul> <p>See <a href="#">Define Sample Size dialog</a> dialog help for more information.</p>
<b>Consistent</b>	<p>Determines whether the operator always creates the same set of random rows for each sample data generation.</p> <ul style="list-style-type: none"> <li>• <b>true</b> - sample data generation is consistent, provided that the number of samples, sample size, and the value of the random seed remain unchanged. If set to <b>true</b>, then <b>Replacement</b> must be <b>false</b>. It must be <b>true</b> to set <b>Key Columns</b>.</li> <li>• <b>false</b> - a different random sample is created each time the operator is run. If set to <b>false</b>, then <b>Random Seed</b> is disabled.</li> </ul> <p>Default: <b>false</b></p>
<b>Random Seed</b>	<p>The seed used for the pseudo-random generation.</p> <ul style="list-style-type: none"> <li>• The seed is the number with which the random sampling algorithm starts to generate the pseudo-random numbers.</li> <li>• The range of this value is from 0 to 1.</li> <li>• A different system-generated seed value is used if the <b>Random Seed</b> value is not specified.</li> </ul>
<b>Replacement</b>	<p>Specify whether the sampling is with or without replacement.</p>



Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>true</b> - sampling with replacement.</li> <li>• <b>false</b> - sampling without replacement. It is the default selection.</li> </ul> <p>If <b>Replacement</b> is selected, both the <b>Consistent</b> and <b>Disjoint</b> properties are set to <b>false</b> and disabled.</p>
<b>Disjoint</b>	<p>Specify whether each sample should be drawn from the entire data set, or from the remaining rows after previous samples are excluded.</p> <ul style="list-style-type: none"> <li>• If you select <b>Disjoint</b>, the same data does not appear in different samples.</li> <li>• If you specify the <b>Percentage</b> type for <b>Sample by</b> property, the sum of all the sample percentages should not be greater than 100%.</li> </ul> <p>If it is set to <b>true</b>, then <b>Replacement</b> parameter must be <b>false</b>.</p>
<b>Key Columns</b>	<p>Used in conjunction with the <b>Consistent</b> property.</p> <ul style="list-style-type: none"> <li>• Click <b>Select Columns</b> to display the Select Columns dialog, which is used for selecting columns to ensure the ordering of the data before generating the pseudo-random sample data set.</li> <li>• The Random Sampling operator uses these key columns to guarantee the order of the rows from the input data set, so that the generation of pseudo-random sample data sets is consistent every time.</li> <li>• If no key columns are specified, the Random Sampling operator assumes that the row ordering of the input data set is consistent. See <a href="#">Key Columns dialog</a> for more information.</li> </ul>
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

**Output:** The data rows of the output table or view of each generated sample are displayed.

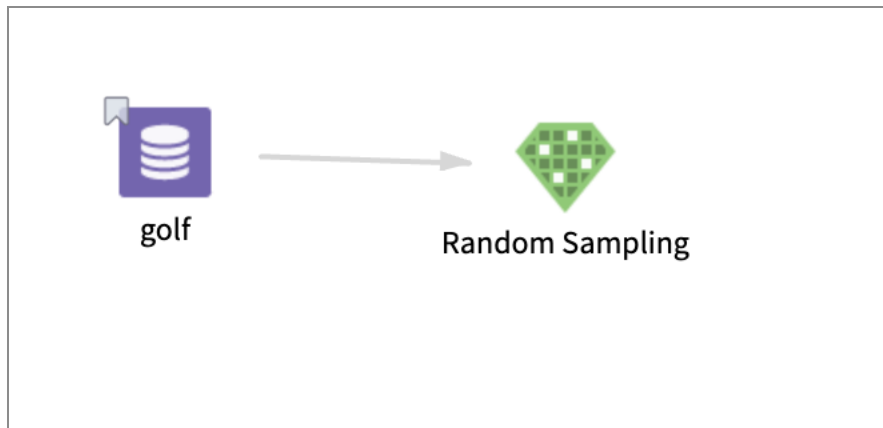
### Output to successive operators

A tabular data set of the sample data tables was created. Typically, the data set is passed on to a [Sample Selector](#) operator, such as Train and Test, to select a sample to use with subsequent operators. An additional column is produced in the output as a result of the operator execution.

**tds\_sample\_column:** This column indicates the sample to which the row has been assigned and is used by the [Sample Selector](#) operator.

## Example

The following example extracts the data rows from the input data set and displays them according to the sample properties defined by the user.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Number of Samples:** 2
- **Sample By:** Percentage
- **Sample Size:** 10%, 20%
- **Consistent:** true
- **Random Seed:** 1
- **Replacement:** false
- **Disjoint:** true
- **Key Columns:** outlook, temperature, humidity, wind
- **Store Results:** Yes

## Output

The following figure displays the output for the parameter settings for the **golf** data set.

Operation created output with 6 columns and 1 rows.

outlook	temperature	humidity	wind	play	_tds_sample_column
overcast	72	90	true	yes	2

## Resampling

This operator changes the distribution of values in a single column. You can use this operator to either balance all values in the selected column or change the proportion of only one value.



## Information at a Glance

**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Sample
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Input

A single tabular data set that has at least one categorical column.

### Bad or Missing Values

Rows with null values in the selected **Column to Resample** are removed from the data set prior to resampling. Null values in other columns do not affect the result.

## Restrictions

Input data must have at least one categorical column with less than 100 distinct values.

## Configuration

The following table provides the configuration details for the Resampling operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Column to Resample</b>	Specify a categorical column with less than 100 distinct values. The columns must be <b>String</b> , <b>Boolean</b> , <b>Integer</b> , or <b>Long</b> data types. If a data set has a column with some other data types, you must cast them before running this operator.

Parameter	Description																
<b>Balance All Values in Selected Column</b>	<p>Specify whether to balance all values in the selected column.</p> <p>Select <b>Yes</b> to balance all values in the selected column by up-sampling rows to match the number of rows of the most common value. The <b>Sample with Replacement</b> must be <b>Yes</b>, and any text entered into <b>Single Value from Selected Column for Resampling</b> is ignored.</p> <p>Select <b>No</b> to resample only one value in the selected column. The user must enter values for <b>Single Value from Selected Column for Resampling</b> and <b>Multiplier for Up-Sampling or Down-Sampling</b>.</p> <p>For example, when the data set contains 3 distinct values in the selected column with the following distribution:</p> <table> <tr> <th>Value</th><th>Count</th></tr> <tr> <td>A</td><td>100</td></tr> <tr> <td>B</td><td>75</td></tr> <tr> <td>C</td><td>50</td></tr> </table> <p>When <b>Yes</b> is selected, the output has the following distribution:</p> <table> <tr> <th>Value</th><th>Count</th></tr> <tr> <td>A</td><td>100</td></tr> <tr> <td>B</td><td>100</td></tr> <tr> <td>C</td><td>100</td></tr> </table> <p>If <b>No</b> is selected, and the user chooses to resample the value B with a multiplier of 3, the output has the following distribution:</p>	Value	Count	A	100	B	75	C	50	Value	Count	A	100	B	100	C	100
Value	Count																
A	100																
B	75																
C	50																
Value	Count																
A	100																
B	100																
C	100																

Parameter	Description								
	<table> <tr> <th>Value</th><th>Count</th></tr> <tr> <td>A</td><td>100</td></tr> <tr> <td>B</td><td>225</td></tr> <tr> <td>C</td><td>50</td></tr> </table> <p><b>Note:</b> You can get exact or approximate counts in the output that depends on the setting of the <b>Exact (Slower)</b> parameter.</p>	Value	Count	A	100	B	225	C	50
Value	Count								
A	100								
B	225								
C	50								
<b>Single Value from Selected Column for Resampling</b>	<p>Specify a character string or a numeric value that appears in the column selected in <b>Column to Resample</b>. If the value does not exist in the selected column, an error appears when running the operator.</p> <p>This parameter is required when <b>Balance All Values in Selected Column</b> is set to <b>No</b>.</p>								
<b>Multiplier for Up-Sampling or Down-Sampling</b>	<p>Specifies a positive decimal number to represent the multiplicative factor that is used to resample the selected column and value.</p> <p>This parameter is required when <b>Balance All Values in Selected Column</b> is set to <b>No</b></p>								
<b>Sample with Replacement</b>	<p>Specify whether samples are with or without replacement. The following settings are recommended:</p> <ul style="list-style-type: none"> <li>For multipliers less than or equal to 1, specify whether samples are with or without replacement.</li> <li>For multipliers greater than 1, select <b>Yes</b> to sample rows with replacement.</li> </ul>								
<b>Exact (Slower)</b>	<p>Specify whether exact resampling is required.</p> <p>For small datasets, we strongly recommend using the option <b>Exact (slower) = Yes</b> because, for non-exact resampling, the output distribution of values can vary from the expected distribution.</p>								

Parameter	Description
	An exact calculation requires an additional pass through the data and results in slower operator execution.
<b>Use Random Seed</b>	Specify whether to use a random seed. Select <b>Yes</b> to use the specified random seed and get repeatable results. Select <b>No</b> to use a system-generated seed value.
<b>Random Seed</b>	The seed used for the pseudo-random generation. It is an integer value that is used if <b>Use Random Seed</b> is <b>Yes</b> .
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Outputs

### Visual Output

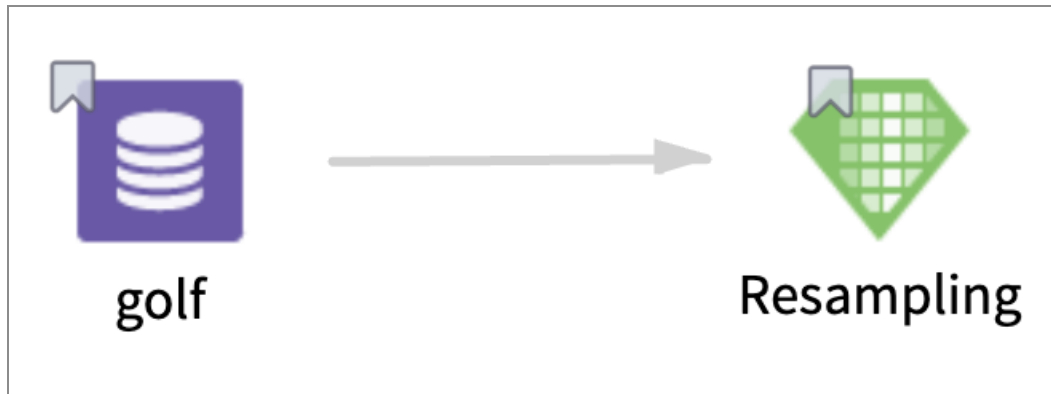
- **Parameters Summary Info:** Displays information about the input parameters and their current settings.
- **Output:** A table that displays the output of a data set for the resampled data.
- **Distribution Summary:** Displays information about the output value distribution such as class, value, and the distribution type.

### Output to successive operators

A tabular data set that contains the resampled data tables created.

## Example

The following example demonstrates the Resampling operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Column to Resample:** play
- **Balance All Values in Selected Column:** No
- **Single Value from Selected Column for Resampling:** yes
- **Multiplier for Up-Sampling or Down-Sampling:** 2.2
- **Sample with Replacement:** Yes
- **Exact (Slower):** Yes
- **Use Random Seed:** No
- **Store Results:** Yes

### Results

These figures displays the results for the parameter settings for the **golf** data set.

### Parameters Summary Info



Name	Value
Column to Resample	play
Balance All Values in Selected Column	No
Single Value from Selected Column for Resampling	yes
Multiplier for Up-Sampling or Down-Sampling	2.2
Sample with Replacement	Yes
Exact (Slower)	Yes
Use Random Seed	No
Random Seed	N/A

## Output

outlook	temperature	humidity	wind	play
sunny	85	85	f	no
overcast	83	78	t	yes
overcast	83	78	t	yes
overcast	64	65	t	yes
sunny	72	95	f	no
sunny	72	95	f	no
sunny	69	70	t	yes
sunny	69	70	t	yes
sunny	69	70	t	yes
overcast	72	90	t	yes
overcast	72	90	t	yes
overcast	72	90	t	yes
overcast	72	90	t	yes
overcast	72	90	t	yes
rain	70	96	f	yes
rain	68	80	f	yes
rain	68	80	f	yes
rain	68	80	f	yes

### Distribution Summary

Class (Top 10)	Value	Distribution Type
yes	9	Input
no	5	Input
yes	20	Output
no	5	Output

## Sample Selector

This operator connects to a preceding [Random Sampling operator](#) and allows you to specify one of the sample data sets (training or testing) generated from that operator for use in succeeding operators.



## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Sample
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes

Parameter	Description
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

This operator connects to a preceding [Random Sampling operator](#) and you can select the training or the testing data sets generated from that operator for use in succeeding operators.

## Input

An input is a sample generating operator, such as the [Random Sampling operator](#).

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Selected Sample</b>	Select the samples as database tables or views from the preceding Random Sampling operator.

## Output

### Visual Output

The data rows of the selected data sample table or view are displayed.

### Output to successive operators

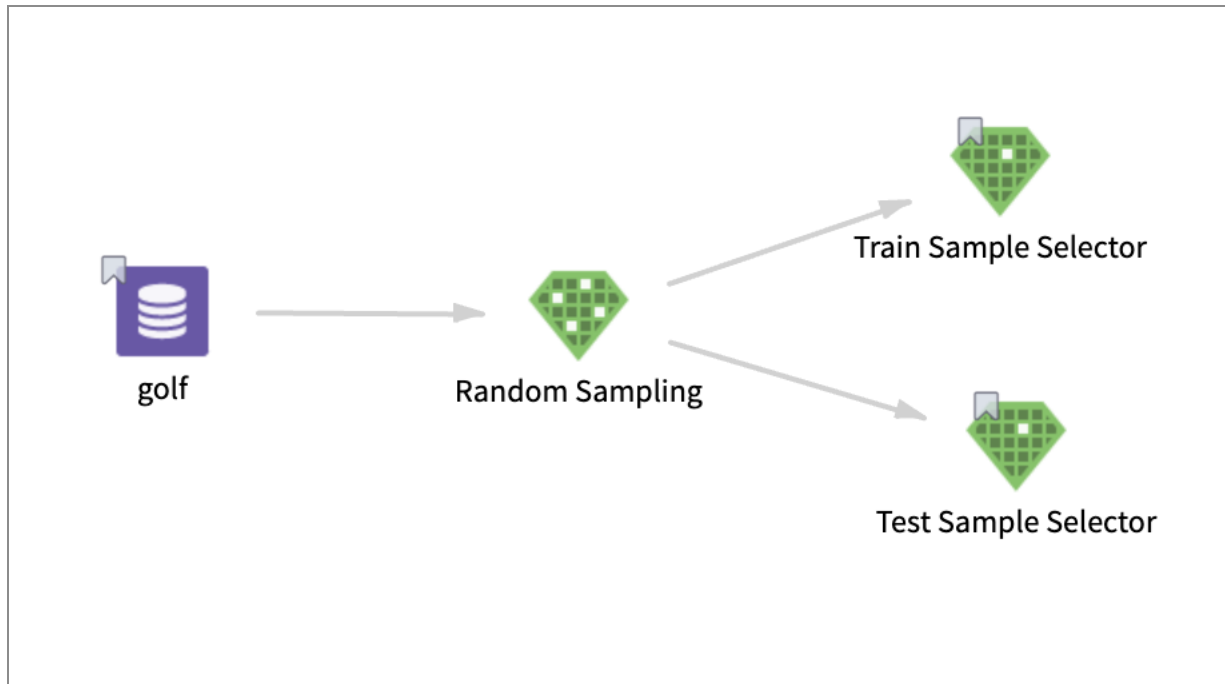
A table output that is used by a downstream operator. The column structure (schema) is generated after running this operator.



**Note:** Run this operator before running a downstream operator.

## Example

The following example demonstrates the use of the Sample Selector operator to select the training and testing data sets generated by the [Random Sampling operator](#).



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set is as follows:

- **Selected Sample:** Sample 1 (80%)

### Output

The following figure displays the output for the parameter settings for the **golf** data set.

Operation created output with 5 columns and 11 rows.

outlook	temperature	humidity	wind	play
sunny	85	85	f	no
sunny	80	90	t	no
sunny	72	95	f	no
sunny	69	70	t	yes
rain	70	96	f	yes
rain	68	80	f	yes
rain	65	70	t	no
rain	75	80	f	yes
rain	71	80	t	no
sunny	75	70	t	yes
overcast	81	75	f	yes

## Tool Operators

The Tools operators are a collection of functions that are helpful with the TIBCO Data Science - Team Studio modeling process and with extending the functionality of TIBCO Data Science - Team Studio flows.

## Export Model to ModelOps

This operator exports the model and schemas (input and output) in [AVRO](#) format. The exported model and schemas can be used in a TIBCO® ModelOps scoring pipeline.



## Information at a Glance

**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Tools
Data source type	TIBCO® Data Virtualization
Sends output to other operators	No
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Input

An input is a [Modeling operator](#) and a single tabular data set against which the model is created.

**Note:** The input data set must be the same as the training data set.

## Algorithm

A model is created using modeling operators. Then, the model and the corresponding data set are used to generate the schema of the output data set. The model is then packaged as a zip archive that the TIBCO® ModelOps can use. The input and output schemas are converted to AVRO schema format before uploading and publishing to the configured TIBCO® ModelOps instance along with the input model.

## Restrictions

The column names must be compliant with the AVRO names. A valid name can only contain ASCII characters for letters, numbers, and underscore. For more information, see the [Apache Avro documentation](#).

## Configuration

The following table provides the configuration details for the Export Model to ModelOps operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>TIBCO® ModelOps Project</b>	Specify the name of the TIBCO® ModelOps project where you want to export the model.  <b>Note:</b> The specified project must exist in TIBCO® ModelOps. If the project does not exist, then an error message is displayed.
<b>Model File Path</b>	Specify the path of the model file for the current workflow. This creates the following artifacts in the TIBCO® ModelOps project: <ul style="list-style-type: none"> <li>• &lt;Model File Path&gt;.zip</li> <li>• &lt;Model File Path&gt;-input.avsc</li> <li>• &lt;Model File Path&gt;-output.avsc</li> </ul>
<b>Description</b>	Enter a brief description of the exported model for the current workflow. This description appears in the model artifact in TIBCO® ModelOps.
<b>Commit Message</b>	Enter a commit message for the model file. It provides information about the changes made to the model.  Default: <b>Committed by TeamStudio</b>
<b>Overwrite?</b>	Specify whether to overwrite the existing model.



Parameter	Description
	<ul style="list-style-type: none"> <li>When set to <b>Yes</b>, the model is overwritten with a new model version (new published version) if the model already exists with the same name. Input and output schemas for the model are overwritten only if the schemas are different from existing ones.</li> <li>When set to <b>No</b>, the model is not overwritten. If the model of the same name already exists, the operator fails with an error message that notifies the user of the existing model.</li> </ul>
	Default: <b>Yes</b>

## Output

### Visual Output

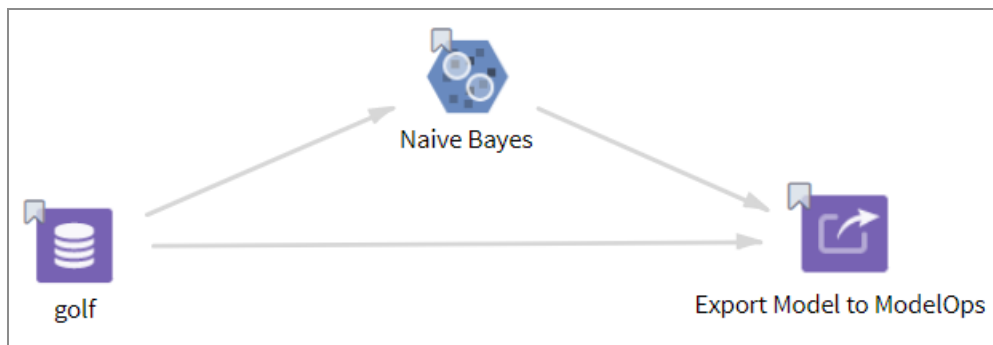
- Summary:** A text field displaying the summary of the exported model.

### Output to successive operators

None. This is a terminal operator.

## Example

The following example uses the **golf** data set to build the **Naive Bayes** model and then exports the model to the TIBCO® ModelOps by using the Export Model to ModelOps operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

## Parameter Setting

The parameter settings for the Export Model to ModelOps operator are as follows:

- **TIBCO® ModelOps Project:** TeamStudioExport
- **Model File Path:** integratedtest/naivebayes
- **Description:** Model from Team Studio
- **Commit Message:** Committed by TeamStudio
- **Overwrite?:** Yes

## Output

The following figure displays the result for the Export Model to ModelOps operator.

The integratedtest/naivebayes has been exported to project TeamStudioExport on ModelOps

Once the model gets exported, you can see the models in the mentioned project in the TIBCO® ModelOps instance. The following figure from the TIBCO® ModelOps displays the model and its information.

TeamStudioExport

Content

Properties

History

TeamStudioExport

integratedtest





Search

Add

Sort by name

Menu

Publish

Name	Status	Description	Last update	Author	Version	Type	Size
 golf	A		14 hours ago	mleehong	Sandbox	Comma-Separated Values	292 Bytes
 naivebayes-input		Input schema for integratedtest/naivebayes	14 hours ago	admin	Rev 1	Apache Avro™ Schema	220 Bytes
 naivebayes-output		Output schema for integratedtest/naivebayes	14 hours ago	admin	Rev 1	Apache Avro™ Schema	352 Bytes
 naivebayes		Model from Team Studio	14 hours ago	admin	Rev 1	Apache Spark™ ML (Compressed)	14.35 KB

## Export Model to Workspace

This operator saves a trained model and stores it as a work file in the workspace. The format of the exported model is Spark Pipelines.



### Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Tools
Data source type	TIBCO® Data Virtualization
Sends output to other operators	No
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

### Input

An input is a trained model operator to export.

### Configuration

The following table provides the configuration details for the Export Model to Workspace operators.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>File Name</b>	Specify the name for the exported model file. Note that the <code>.model</code> file extension should be omitted because the file extension is automatically appended to the model file name.  <div><b>Note:</b> The File Name must be unique for each operator.</div> <p>For more information about the variables used for the default name, see <a href="#">Workflow Variables</a>.</p>
<b>Description</b>	Enter a brief description of the model file for the current workflow.
<b>Overwrite?</b>	Specify whether to overwrite an existing model file. <ul style="list-style-type: none"> <li>When set to <b>Yes</b>, the existing model file is overwritten (If the model file already exists with the same name). A new version of the model file is created in the workspace, and previous versions are still accessible through the work file version-control option.</li> <li>When set to <b>No</b>, the model file is not overwritten. If the model file with the same name already exists, the operator fails with an error message that notifies the user of the existing file.</li> </ul>

## Output

### Visual Output

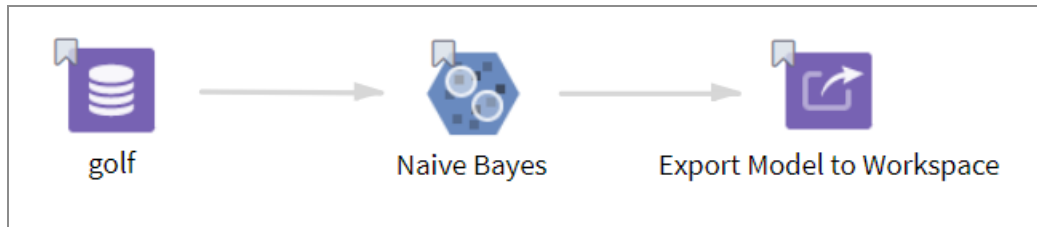
- **Summary:** A text field displaying the summary of the output file.

### Output to successive operators

None. This is a terminal operator.

## Example

The following example uses the **golf** data set to build the [Naive Bayes](#) model and then exports the model to the workspace by using the Export Model to Workspace operator.



## Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

## Parameter Setting

The parameter settings for the operator are as follows:

- **File Name:** nb\_model
- **Description:** Export model file for the golf data set.
- **Overwrite?:** Yes

## Output

The following figure displays the result for the Export Model to Workspace operator.

The model file is stored in the current workspace as nb\_model.model

## Export to SBDF

This operator converts a tabular data set to the Spotfire Binary Data Frame (SBDF) format. The SBDF files are stored in the same workspace as the workflow and can be downloaded for use in TIBCO Spotfire®.



## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Tools
Data source type	TIBCO® Data Virtualization
Sends output to other operators	No
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Input

An input is a single tabular data set.

## Bad or Missing Values

Null and empty values in the inputs are converted to blank cells in the SBDF file.

## Restrictions

By default, the custom operator can export only up to 10 GB of data. You can modify this limit in the `alpine.conf` file by setting the following configuration.

**i Note:** Raising this limit might overload the memory reserved for TIBCO Data Science - Team Studio.

```
custom_operator.sbdf_export.max_mb_input_size_alpine_server=10240
```

## Configuration

The following table provides the configuration details for the Export to SBDF operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>File Name</b>	Specify the name of the SBDF file stored in the TIBCO Data Science - Team Studio workspace. The <code>.sbd</code> file extension should be omitted because the file extension is automatically appended to the file name.  <b>Note:</b> The File Name must be unique for each operator.
<b>Description</b>	Enter a brief description of the SBDF file for the current workflow.
<b>Overwrite?</b>	Specify whether to overwrite the SBDF file. <ul style="list-style-type: none"> <li>When set to <b>Yes</b>, the SBDF file is overwritten (If the SBDF file already exists with the same name). A new version of the SBDF file is created in the workspace, and previous versions are still accessible through the work file version-control option.</li> <li>When set to <b>No</b>, the SBDF file is not overwritten. If the SBDF file of the same name already exists, the operator fails with an error message that notifies the user of the existing work file.</li> </ul>

## Output

### Visual Output

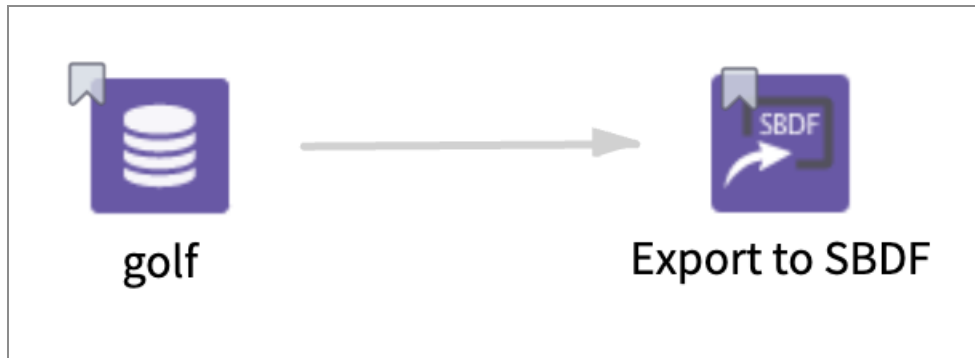
- **Summary:** A text field displaying the summary of the output file.

### Output to successive operators

None. This is a terminal operator.

## Example

The following example demonstrates the Export to SBDF operator.



## Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

## Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **File Name:** golf\_sbdf\_file
- **Description:** SBDF file for the golf data set.
- **Overwrite?:** Yes

## Output

The following figure displays the result for the Export to SBDF operator.

The SBDF file is stored in the current workspace as golf\_sbdf\_file.sbdf

## Load Model

This operator loads a TIBCO Data Science - Team Studio Analytic Models from the current workspace to use with [prediction](#) and [model validation](#) operators.





## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Tools
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

The Analytic Models are models that have been trained within TIBCO Data Science - Team Studio using the [modeling operators](#) and exported using the [Export Model to Workspace](#) operator. When the model is exported, you can use the Load Model operator to use your exported models in various workflows.

## Input

A Load Model operator does not take any input, since it is a source operator.

## Configuration

The following table provides the configuration details for the Load Model operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Model</b>	Select the model to load. The list includes all analytic model (.am) files in the current workspace.

## Output

### Visual Output

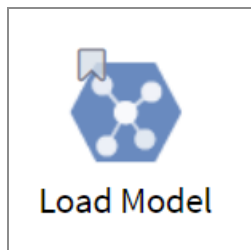
- **Summary:** A text field displaying the summary of the loaded model.

### Data Output

A model that corresponds to the selected analytic models type.

## Example

The following example illustrates the uploading of an analytic model exported to the workspace (using the [Export Model to Workspace](#) operator) using the Load Model operator in the new workflow.



### Data

None.

### Parameter Setting

The parameter settings for the **Load Model** operator are as follows:

- **Model:** nb\_model.model

## SQL Execute

This operator is used to run a SQL statement for creating a data source that can be used by the downstream operators.



## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Tools
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

The SQL statement executed inside this operator does not have any dependency on its preceding or succeeding operator.

The connection only ensures its execution sequence within the analytic workflow.

## Input

The SQL Execute operator does not take any input. It is a source operator.

## Configuration

The following table provides the configuration details for the SQL Execute operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>SQL File</b>	Select the SQL file from the dropdown list to run. The list includes all the SQL (.sql) files in the current workspace. You can also create a SQL file. For more information on creating a SQL file, see <a href="#">Creating a SQL Work File</a> .

Parameter	Description
<b>Output Schema</b>	Specify the schema for the output table.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , it enables the operator to save the results. When set to <b>No</b> , it disables the operator to save the results.

## Output

### Visual Output

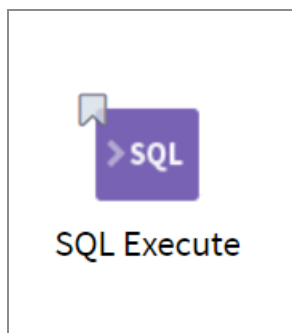
None.

### Data Output

A data set that corresponds to the SQL statements.

## Example

The following example illustrates the data set that corresponds to the SQL statements using the SQL Execute operator. This data set can be used by the downstream operators in a workflow.



### Data

None.

### Parameter Setting

The parameter settings for the **SQL Execute** operator are as follows:

- **SQL File:** golf.sql
- **Output Schema:** Datasets\_shared
- **Store Results:** Yes

## Output

The following figure displays the result for the **SQL Execute** operator.

Operation created output with 5 columns and 14 rows.

outlook	temperature	humidity	wind	play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	78	false	yes
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
overcast	72	90	true	yes
rain	70	96	false	yes

## Transformation Operators

The Transformation operators provide different ways to prepare data for modeling.

### Batch Aggregation

This operator performs aggregations on multiple columns using the Batch Aggregation algorithm from [Spark MLlib](#).



## Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The Batch Aggregation operator takes an input data set and performs multiple aggregations on multiple columns. Rows of output data set are represented by the aggregation computations for each group determined by the **Group By** columns.

## Input

An input is a single tabular data set.

### Missing or Null Values

Skips missing or null values while performing aggregations or calculating the number of distinct values.

## Configuration

The following table provides the configuration details for the Batch Aggregation operator.

Parameter	Description
Notes	Notes or helpful information about this operator's parameter settings. When

Parameter	Description
	you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Group By</b>	Specify the columns in the input data set to determine grouped results. Click <b>Select Columns</b> to open the dialog for selecting the available columns from the input data set for analysis.
<b>Find Maximum</b>	Maximum value for each of these columns for each group. Click <b>Select Columns</b> to open the dialog for selecting the available columns from the input data set for analysis.
<b>Find Minimum</b>	Minimum value for each of these columns for each group. Click <b>Select Columns</b> to open the dialog for selecting the available columns from the input data set for analysis.
<b>Calculate Sum</b>	Sums for each of these columns for each group.
<b>Calculate Mean</b>	Mean value for each of these columns for each group.
<b>Calculate Variance</b>	Variance for each of these columns for each group. See <a href="#">Aggregation Methods for Batch Aggregation</a> for implementation and performance details.
<b>Calculate Standard Deviation</b>	Standard deviation for each of these columns for each group. See <a href="#">Aggregation Methods for Batch Aggregation</a> for implementation and performance details.
<b>Calculate Number of Distinct (slower)</b>	Number of distinct values (excluding null values) for each of these columns for each group. See <a href="#">Aggregation Methods for Batch Aggregation</a> for implementation and performance details.
<b>Calculate Median (slower)</b>	Median for each of these columns for each group. See <a href="#">Aggregation Methods for Batch Aggregation</a> for implementation and performance details.
<b>Column Name</b>	Specify whether the aggregation type is added to the beginning or the end

Parameter	Description
<b>Format</b>	of the column name in the output. The available options are <b>suffix</b> and <b>prefix</b> .  Default: <b>suffix</b>
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

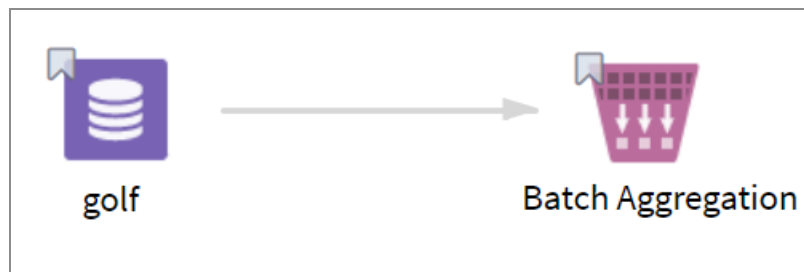
- **Output:** Displays the specified aggregates for specified columns of the input data set.
- **Parameter Summary Info:** Displays information about the input parameters. A list of the input parameters and their current settings.
- **Column Data Sizes:** Displays the size of the entire data set.

### Output to successive operators

A model object that can be used with operators.

## Example

The following example demonstrates the Batch Aggregation operator.





## Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

## Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Group By:** outlook
- **Find Maximum:** humidity
- **Find Minimum:** temperature
- **Calculate Sum:** temperature
- **Calculate Mean:** temperature, humidity
- **Calculate Variance:** humidity
- **Calculate Standard Deviation:** humidity
- **Calculate Number of Distinct (slower):** temperature, wind, play
- **Column Name Format:** suffix
- **Store Results:** Yes

## Results

These figures display the results for the parameter settings for the **golf** data set.

## Output

outlook	group_size	temperature_mean	temperature_min	temperature_distinct	temperature_sum	humidity_mean	humidity_max	humidity_sd	humidity_var	wind_distinct	play_distinct
rain	5	69.8000	65.0000	5	349.0000	81.2000	96.0000	9.3380	87.2000	2	2
overcast	4	75.0000	64.0000	4	300.0000	77.0000	90.0000	10.2960	106.0000	2	1
sunny	5	76.2000	69.0000	5	381.0000	82.0000	95.0000	11.5110	132.5000	2	2

In the above output, the groups are not in alphabetically sorted order. To order the aggregations, connect this operator to a sorting operator.

## Parameter Summary Info

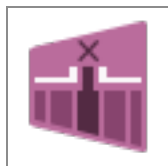
Name	Value
Calculate Median (slower)	N/A
Find Maximum	humidity
Find Minimum	temperature
Calculate Variance	humidity
Calculate Sum	temperature
Calculate Mean	temperature, humidity
Calculate Standard Deviation	humidity
Calculate Number of Distinct (slower)	temperature, wind, play
Group By	outlook
Column Name Format	suffix

## Column Data Sizes

Column Name	Data Size
All Columns	14

## Column Cleanser

This operator removes the columns according to the specified column's completeness or variance criteria.



## Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

This operator applies a set of rules to remove columns, easing the burden of specifying filtering criteria column by column. The user selects the columns to test, and then a filtering condition is set. According to this condition, columns are selectively removed.

According to the filtering conditions defined, the Sparsity, High Variance, and Low-Variance checks are calculated. Multiple filtering conditions can be applied. If a Low-Variance check involving the calculation of the coefficient of variation is applied to a column that has zero mean and all identical values, the columns are removed and a warning appears in the Summary tab.

## Input

An input is a single tabular data set.

## Configuration

The following table provides the configuration details for the Column Cleanser operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Select these columns and treat as continuous</b>	Specify the columns as continuous columns. It must be <b>Double</b> or <b>Integer</b> . Click <b>Select Columns</b> to select

Parameter	Description
	<p>the required columns.</p> <p><b>Note:</b> The columns selected in the <b>Select these columns and treat as categorical</b> parameter are not available.</p>
<b>Select these columns and treat as categorical</b>	<p>Specify the columns as categorical columns. Click <b>Select Columns</b> to select the required columns.</p> <p><b>Note:</b> The columns selected in the <b>Select these columns and treat as continuous</b> parameter are not available.</p>
<b>Remove columns with a percentage of missing rows higher than (-1=ignore, accepts real number 0-100)</b>	<p>Removes the column with a percentage of missing values higher than the specified number. This is known as a Sparsity check. The value can be -1, or a real number between 0 to 100.</p> <p>Default: <b>-1.0</b></p> <p><b>Note:</b> To ignore this filtering condition, the value must be set to <b>-1</b>.</p>
<b>Remove categorical columns with a count of distinct values higher than (-1=ignore, accepts integer number 0-10000)</b>	<p>Removes the categorical columns with a count of distinct values higher than the specified number. This is known as a High Variance check. The value can be -1, or an integer number between 0 to 10000.</p> <p>Default: <b>-1</b>.</p> <p><b>Note:</b> To ignore this filtering condition, the value must be set to <b>-1</b>.</p>
<b>Remove categorical columns with a count of distinct values higher than this percentage of the number of rows (-1=ignore, accepts</b>	<p>Removes the categorical columns with a percentage of distinct values higher than the specified number. This is known as a High Variance check. The value can be -1, or a real number between 0 to 100.</p>

Parameter	Description
<b>real number 0-100)</b>	<p>Default: <b>-1</b>.</p> <p><b>Note:</b> To ignore this filtering condition, the value must be set to <b>-1</b>.</p>
<b>Remove categorical columns with a percentage of the most frequent category higher than (-1=ignore, accepts real number 0-100)</b>	<p>Removes the categorical columns of the most frequent category which appears more often than the specified percentage of rows. This is known as a Low Variance check. The value can be -1, or a real number between 0 to 100.</p> <p>Default: <b>-1</b>.</p> <p><b>Note:</b> To ignore this filtering condition, the value must be set to <b>-1</b>.</p>
<b>Remove numeric columns with a Coefficient of Variation(Standard Deviation divided by Mean) lower than (-1=ignore, accepts real number 0-0.01)</b>	<p>Removes the continuous columns with the coefficient of variation lower than the specified value. This is known as a Low Variance check. The value can be -1, or a real number between 0 to 100.</p> <p>Default: <b>-1</b>.</p> <p><b>Note:</b> To ignore this filtering condition, the value must be set to <b>-1</b>.</p>
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

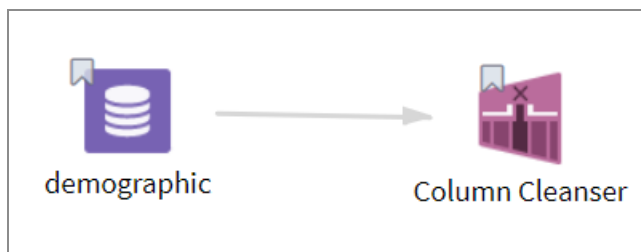
- **Output:** Preview of the clean data consisting of the columns which satisfy the defined filtering conditions and passed the data cleaning checks.
- **Summary:** Displays information about the removed columns.

### Output to Successive operator

A single tabular data with cleaned data consisting of the columns satisfying the filtering conditions.

## Example

The following example shows the cleansed data for the given data set using the Column Cleanser operator.



### Data

**demographic:** A data set for **demographic** that contains the following information:

- Multiple columns such as ID, AGE\_IN\_YEARS, LEVEL\_OF\_EDUCATION, YEARS\_WITH\_CURRENT\_EMPLOYER, and YEARS\_AT\_CURRENT\_ADDRESS.
- Multiple rows (850 rows).

### Parameter Setting

The parameter settings for the **demographic** data set are as follows:

- **Select these columns and treat as continuous:** ID, AGE\_IN\_YEARS
- **Select these columns and treat as categorical:** LEVEL\_OF\_EDUCATION, YEARS\_WITH\_CURRENT\_EMPLOYER, YEARS\_AT\_CURRENT\_ADDRESS
- **Remove columns with a percentage of missing rows higher than (-1=ignore,**

**accepts real number 0-100): 20**

- **Remove categorical columns with a count of distinct values higher than (-1=ignore, accepts integer number 0-10000): -1**
- **Remove categorical columns with a count of distinct values higher than this percentage of the number of rows (-1=ignore, accepts real number 0-100): -1.0**
- **Remove categorical columns with a percentage of the most frequent category higher than (-1=ignore, accepts real number 0-100): 30**
- **Remove numeric columns with a Coefficient of Variation(Standard Deviation divided by Mean) lower than (-1=ignore, accepts real number 0-0.01): -1.0**
- **Store Results: Yes**

## Results

These figures display the results for the parameter settings for the **demographic** data set.

## Output

ID	AGE_IN_YEARS	YEARS_WITH_CURRENT_EMPLOYER	YEARS_AT_CURRENT_ADDRESS
1	41	8	21
2	41	18	6
3	41	7	6
4	41	2	22
5	33	14	8
6	33	2	14
7	49	11	5
8	25	0	6
9	41	21	2
10	25	3	2

## Summary

Description	Value
List of columns removed by the second condition - removing categorical columns with a count of distinct values higher than	N/A
Number of columns removed by the third condition - removing categorical columns with a count of distinct values higher than percentage	0
List of columns removed by the third condition - removing categorical columns with a count of distinct values higher than percentage	N/A
Number of columns removed by the fourth condition - removing categorical columns with a percentage of the most frequent category higher than	1
List of columns removed by the fourth condition - removing categorical columns with a percentage of the most frequent category higher than	LEVEL_OF_EDUCATION
Number of columns removed by the fifth condition - removing continuous columns with a Coefficient of Variation lower than	0
List of columns removed by the fifth condition - removing continuous columns with a Coefficient of Variation lower than	N/A
Number of columns left	4
Number of columns removed	1
List of columns removed	LEVEL_OF_EDUCATION
Number of columns removed by the first condition - removing columns with a percentage of missing rows higher than	0
List of columns removed by the first condition - removing columns with a percentage of missing rows higher than	N/A
Number of columns removed by the second condition - removing categorical columns with a count of distinct values higher than	0

## Column Filter

This operator selects a subset of the columns from the data source. Only the columns selected remain in the output data set.



### Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.



Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

This operator is particularly helpful when a data source has several columns that are not needed for the data analysis workflow.

## Input

An input is a single tabular data set.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	Specify the columns to be made available for analysis.  Click <b>Select Columns</b> to open the dialog to specify columns. See the <a href="#">Columns</a> dialog for more information.
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

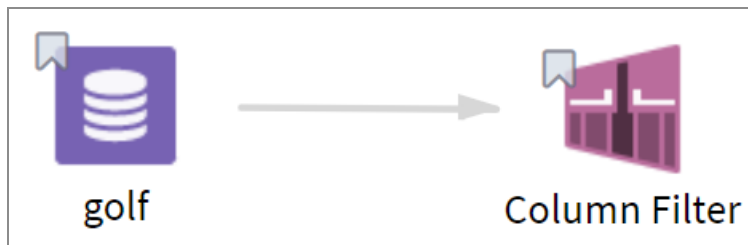
A table that displays the output of a data set.

### Data Output

A tabular data set of the newly created table or a view.

## Example

The following example shows the data set created from data fields of the given data set, forming a new table using the Column Filter operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Columns:** outlook, temperature, wind
- **Store Results:** Yes

### Output

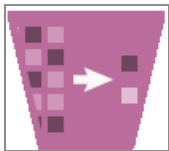
The following figure displays the output for the parameter settings for the **golf** data set.

Operation created output with 3 columns and 14 rows.

outlook	temperature	wind
sunny	85	f
sunny	80	t
overcast	83	f
overcast	64	t
sunny	72	f
sunny	69	f
overcast	72	t
rain	70	f

## Distinct

This operator returns only distinct combinations of values from specific columns of a database source. Rows are not returned in any particular order, but each combination of values within a row is distinct from other rows.



## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization

Parameter	Description
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Input

An input is a single tabular data set. You can choose the columns from which you want to create distinct combinations, and then the operator performs the calculation.

### Bad or Missing Values

The missing values are considered as part of the determination of distinct values. If a column has a missing value, the missing value is considered distinct from a value.

This operator handles null values by eliminating them from the input calculation. To prevent this behavior, use the [Null Value Replacement](#) operator on the initial training data to replace bad or missing values.

## Configuration

The following table provides the configuration details for the Distinct operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Distinct Columns</b>	Specify the columns from the data source by which to generate rows of data, where each row has a distinct combination of column values.
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated.

Parameter	Description
	By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

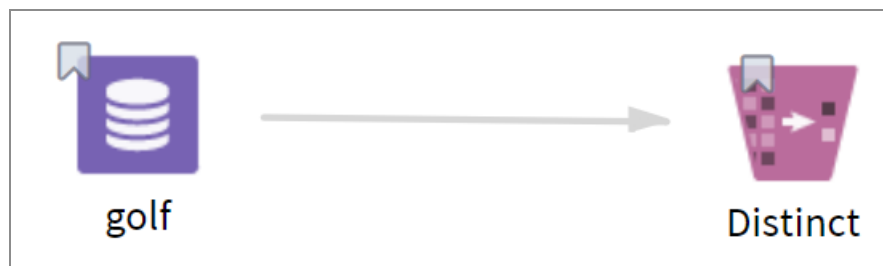
A table that displays the distinct combinations of values from selected columns of a data set.

### Output to Successive operator

A single tabular data set with distinct combinations of values from specific columns.

## Example

The following example illustrates the Distinct operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Distinct Columns:** outlook, play
- **Store Results:** Yes

## Output

The following figure displays the output for the parameter settings for the **golf** data set.

Operation created output with 2 columns and 5 rows.

outlook	play
rain	no
sunny	yes
overcast	yes
rain	yes
sunny	no

## Dynamic Column Filter

This operator selects a subset of the columns from the data source. Only the selected columns remain in the output data set.



## Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform

Parameter	Description
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

This operator is particularly helpful when there are many columns in a data source that are not needed for the data analysis workflow or in case, you need to select columns dynamically. You can also specify a regular expression for selecting a subset of columns by column names.

The Dynamic Column Filter operator is designed to be used in conjunction with Modeling operators. When the **Use all available columns as Predictors** parameter is set to **Yes** in these operators, the Dynamic Column Filter operator that is placed before the Modeling operators decides which columns are used as predictors dynamically.

**i Note:** The parameters **Columns to Include**, **Data Types to Include**, and **Column Names to Include (regex)** work as a union for adding a column to the operator.

## Input

An input is a single tabular data set.

## Configuration

The following table provides the configuration details for the Dynamic Column Filter operator.

**i Note:** At least one of the parameters must be used from **Columns to Include**, **Data Types to Include**, and **Column Names to Include (regex)**. The output is the columns satisfying at least one of the conditions specified by these inclusion parameters.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns to Include</b>	The columns to be made available for analysis. Only static columns can be selected. Click <b>Select Columns</b> to select the columns. It is an optional parameter.
<b>Data Types to Include</b>	<p>Specify the data type that you want to include. Click <b>Select</b> to select the data type. It is an optional parameter. The following values are available:</p> <ul style="list-style-type: none"> <li>• <b>String</b></li> <li>• <b>Int</b></li> <li>• <b>Boolean</b></li> <li>• <b>Long</b></li> <li>• <b>Double</b></li> <li>• <b>DateTime</b></li> </ul> <p>By default, all data types are selected.</p> <p><b>Note:</b> You can also select more than one data type or none. By default all data types are selected, so without changing this default choice, you get the whole table as output.</p>
<b>Column Names to Include (regex)</b>	<p>Specify the regular expression or wildcards for selecting a subset of columns by name. It is an optional parameter.</p> <p><b>Note:</b> For the regular expression, the dollar (\$) symbol is not accepted. You can use \Z instead.</p>
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID,



Parameter	Description
	workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , it enables the operator to save the results. When set to <b>No</b> , it disables the operator to save the results.

## Output

### Visual Output

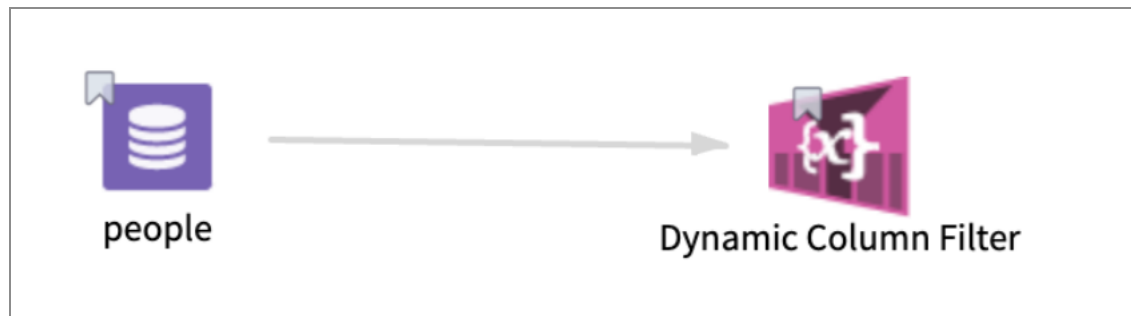
The data rows of the output table or view are displayed (a limited number of rows of the data).

### Output to successive operators

A tabular data set with columns that satisfy one of the conditions defined by the parameters for the rule of inclusion. This data with only required columns can be used with subsequent operators.

## Example

The following example demonstrates the workflow of a Dynamic Column Filter operator.



### Data

Input data set has multiple rows and the following columns:

CustomerStatus [ String ]
InitialChannel [ String ]
Handset [ String ]
ExtraChargeForTexting [ Double ]
ExtraChargeForCalls [ Double ]
ExtraChargeForData [ Double ]
ChargeLast12Months [ Double ]
ChargeLast3Months [ Double ]
ChargeLastMOnth [ Double ]
TotalInCalls [ Int ]
TotalOutCalls [ Int ]
PropInCallsFromChurner [ Double ]
PropOutCallsToChurner [ Double ]
DataUpload [ Int ]
DataDownLoad [ Int ]

### Parameter Setting

The parameter settings for the Dynamic Column Filter operator are as follows:

- **Columns to Include:** CustomerStatus
- **Data Types to Include:** String
- **Column Names to Include (regex):** ^Data

### Results

The following figure displays the results from the Dynamic Column Filter operator.

Operation created output with 5 columns and 5,000 rows. Visualization of rows limited to 25.

CustomerStatus	InitialChannel	Handset	DataUpload	DataDownLoad
Active	Internet	Big Screen	116	112
Active	Direct Mail	Sleek and Simple	145	42
Churn	Internet	Big Screen	19	836
Active	Mall Kiosk	Big Screen	16	96
Active	Direct Mail	Micro	280	22
Active	Mall Kiosk	Micro	5	217

In the above table, the column **CustomerStatus** is included based on **Columns to Include** parameter. The columns **InitialChannel** and **Handset** are included based on the **Data Types to Include** parameter. Finally, the columns **DataUpload** and **DataDownLoad** are in the results because they satisfy the regular expression asking for all variables starting with **Data**.

## Additional regular expression examples

The following table provides an example of regular expressions. The green color is highlighting the columns that are included based on the specified regular expressions.

		All variables	Starting part	Ending part	Occurrence of substring	Or example	Specific name
Regular expression:		.	^Extra	Months\Z	Last	Last Out	^DataUpload\Z
Column names	CustomerStatus [ String ]	Y	N	N	N	N	N
	InitialChannel [ String ]	Y	N	N	N	N	N
	Handset [ String ]	Y	N	N	N	N	N
	ExtraChargeForTexting [ Double ]	Y	Y	N	N	N	N
	ExtraChargeForCalls [ Double ]	Y	Y	N	N	N	N
	ExtraChargeForData [ Double ]	Y	Y	N	N	N	N
	ChargeLast12Months [ Double ]	Y	N	Y	Y	Y	N
	ChargeLast3Months [ Double ]	Y	N	Y	Y	Y	N
	ChargeLastMOnth [ Double ]	Y	N	N	N	N	N
	TotalInCalls [ Int ]	Y	N	N	N	N	N
	TotalOutCalls [ Int ]	Y	N	N	N	Y	N
	PropInCallsFromChurner [ Double ]	Y	N	N	N	N	N
	PropOutCallsToChurner [ Double ]	Y	N	N	N	Y	N
	DataUpload [ Int ]	Y	N	N	N	N	Y
	DataDownLoad [ Int ]	Y	N	N	N	N	N

## Fourier Bessel

This operator reduces data from measurements on a disk into coefficients corresponding to Fourier Bessel basis functions.



### Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The Fourier Bessel operator is a method of expressing a set of original measurements on a disk as a series of coefficients. These coefficients are the result of applying a Fourier expansion to the original data that uses Bessel functions as its orthogonal basis. The goal is to reduce the data dimensionality in a way that it can be applied to all disk configurations and is robust to missing data.

## Input

An input is a single tabular data set.

### Bad or Missing Values

- The missing data in **ID Columns**, **X Coordinate Column**, and **Y Coordinate Column** does not cause an error.
- The missing data in measurement values is handled by the algorithm.

## Configuration

The following table provides the configuration details for the Fourier Bessel operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.

Parameter	Description
<b>Shape Column</b>	Specify the column that contains shape ID.
<b>ID Columns</b>	Specify the columns that uniquely identify a disk. Click <b>Select Columns</b> to select the required columns.
<b>Measurement Column</b>	Specify the column that contains measurements. The column must be numerical.
<b>X Coordinate Column</b>	Specify the column that contains X-coordinates. The column must be an integer.
<b>Y Coordinate Column</b>	Specify the column that contains Y-coordinates. The column must be an integer.
<b>Bessel Nu</b>	<p>Specify the number of Bessel Q functions to use. The parameter provides the total number of Bessel coefficients produced together with Bessel Q. The maximum value permitted is 30.</p> <p>Default: <b>12</b></p>
<b>Bessel Q</b>	<p>Specify the number of zeros of each Bessel Nu function to use. The parameter provides the total number of Bessel coefficients produced together with Bessel Nu. The maximum value permitted is 30.</p> <p>Default: <b>12</b></p>
<b>Compute RMS error</b>	Specify whether to compute the root means square (RMS) error. When set to <b>Yes</b> , the operator computes the RMS error. When set to <b>No</b> , the operator does not compute the RMS error.
<b>Normalization Method</b>	<p>Specify the type of normalization method for the measurement column. The following values are available:</p> <ul style="list-style-type: none"> <li>• <b>Z-Score</b></li> <li>• <b>Min-Max</b></li> <li>• <b>None</b></li> </ul> <p>Default: <b>Z-Score</b></p>

Parameter	Description
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

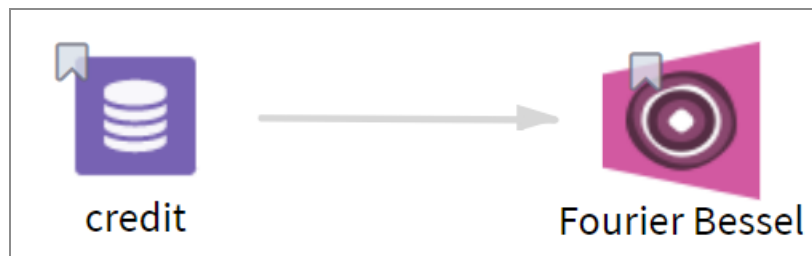
- **Parameter Summary Info:** Displays information about the input parameters and their current settings.
- **Output:** Displays a table with Bessel coefficients as well as the ID column and Shape column.
- **RMS Errors for Shapes:** Displays the root mean squared errors between original disk measurements and the reconstruction from the Fourier Bessel expansion when **Compute RMS error** is set to **Yes**.

### Output to Successive operator

A single tabular data set with Bessel coefficients (for example, COS\_0\_1, SIN\_0\_1, ...  $C_{mn}$ . Where  $m = 0$  to BesselNu-1 and  $n = 1$  to BesselQ), as well as the ID and Shape columns.

## Example

The following example illustrates the Fourier Bessel operator.



## Data

**credit:** This data set contains the following information:

- Multiple columns such as id, times90dayslate, revolving\_util, debt\_ratio, credit\_lines, monthly\_income, times30dayslate\_2years, and srsdlqncy.
- Multiple rows (1000 rows).

## Parameter Setting

The parameter settings for the **credit** data set are as follows:

- **Shape Column:** credit\_lines
- **ID Columns:** id
- **Measurement Column:** debt\_ratio
- **X Coordinate Column:** srsdlqncy
- **Y Coordinate Column:** times30dayslate\_2years
- **Bessel Nu:** 2
- **Bessel Q:** 2
- **Compute RMS error:** Yes
- **Normalization Method:** Min-Max
- **Store Results:** Yes

## Results

The following figure displays the output for the parameter settings for the **credit** data set.

## Parameter Summary Info



Name	Value
Y Coordinate Column	times30dayslate_2years
Measurement Column	debt_ratio
Shape Column	credit_lines
X Coordinate Column	srsdlqncy
Id Columns	id
Compute RMS error	Yes
Bessel Nu	4
Bessel Q	4

## Output

credit_lines	id	COS_0_1	SIN_0_1	COS_0_2	SIN_0_2	COS_1_1	SIN_1_1	COS_1_2	SIN_1_2
4	3778	0.24	0	0.3113	0	0	0	0	0
4	5239	0.663	0	0.8599	0	0	0	0	0
4	5694	1.0269	0	1.3319	0	0	0	0	0
4	5420	0	0	0	0	0	-1.9908	0	1.9908
4	394	0.1082	0	0.1403	0	0	0	0	0
4	3798	0.0885	0	0.1148	0	0	0	0	0
4	648	0.3696	0	-0.0667	0	0.6588	0	0.2849	0

## RMS Errors for Shapes

shape	rmSError
3	2.8096
7	11.9352
4	24.996
5	90.3485
6	9.1598

## Join

This operator joins the data set by allowing users to define the input data set such as an alias, output columns, and the join condition.



### Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

### Input

An input requires two or more data sets.

**i Note:** For the current release (7.1.0), a maximum of two data sets are accepted as inputs.

Both tables must be located in the same database. Join does not work on tables located in different databases. See the TIBCO Data Science - Team Studio Operator and Data Source Compatibility for any data source exceptions for the Join operator.

## Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

The following table provides the configuration details for the Join operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Create Sequence ID</b>	Click <b>Yes</b> to create an ID column on the output data set of the Join operator.
<b>Join Conditions</b>	Click <b>Define Join Conditions</b> to display the Join Properties dialog. See <a href="#">Join Properties - Database dialog</a> for more information.  For information about creating the Join condition, see <a href="#">Creating a Join condition for a database join</a> .
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

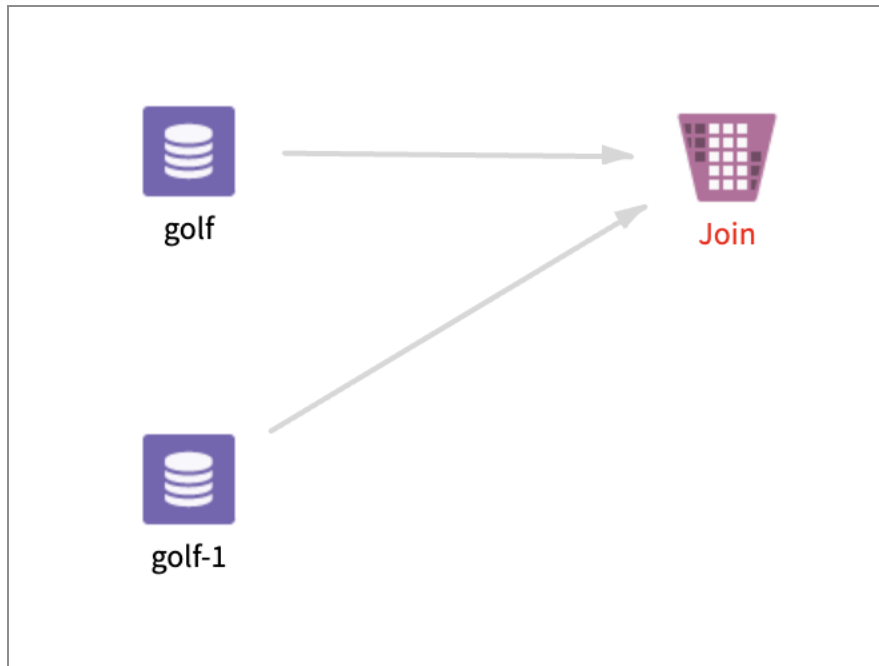
## Output

### Visual Output

- **Output:** A table that displays the output of joined data sets.

## Example

The following example joins the **golf** data set and **golf-1** data set into a single data set using the Join operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

**golf-1:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** and **golf-1** data set are as follows:

- **Create Sequence ID:** No
- **Join Conditions:** outlook, temperature, humidity, wind

- **Store Results:** Yes

## Output

The following figure displays the output for the parameter settings for the given data sets.

Operation created output with 4 columns and 196 rows.

outlook	temperature	humidity	wind
sunny	85	85	false
sunny	80	90	true
overcast	83	78	false
overcast	64	65	true
sunny	72	95	false
sunny	69	70	false
overcast	72	90	true
rain	70	96	false
rain	68	80	false
rain	65	70	true
rain	75	80	false
<b>rain</b>	<b>71</b>	<b>80</b>	<b>true</b>
sunny	75	70	true
overcast	81	75	false
sunny	85	85	false
sunny	80	90	true
overcast	83	78	false

## Normalization

This operator performs normalization on the selected columns of the input data set. Normalization means adjusting values measured on different scales to a notionally

common scale.



## Information at a Glance

**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

You can accomplish normalization in the following ways:

- By specifying a user-defined minimum and maximum value.
- By a z-transformation (for example, on mean 0 and variance 1).
- By a transformation as a proportion of the average or sum of the respective attribute.

Your selection translates into four possible types of normalization:

- Z-Transformation.
- Proportion Transformation.
- Range Transformation.
- Divide-By-Average Transformation.

## Input

An input is a single tabular data set.

### Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

The following table provides the configuration details for the Normalization operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Method</b>	Specify the normalization method to use. The following values are available: <ul style="list-style-type: none"><li>• <b>Divide-By-Average Transformation:</b> Calculate by the sample's average.</li><li>• <b>Proportional Transformation:</b> Calculate by sample's sum.</li><li>• <b>Z-Transformation:</b> Calculate by sample's mean and variance.</li><li>• <b>Range Transformation:</b> Calculate by sample's minimum and maximum value.</li></ul>
<b>Range Minimum</b>	Specify the minimum value in Range transformation.
<b>Range Maximum</b>	Specify the maximum value in Range transformation.
<b>Columns</b>	Specify the columns to normalize by selecting the available numerical columns. Click <b>Column Names</b> to open the dialog for selecting the available numerical columns.
<b>Output Schema</b>	Specify the schema for the output table or view.



Parameter	Description
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

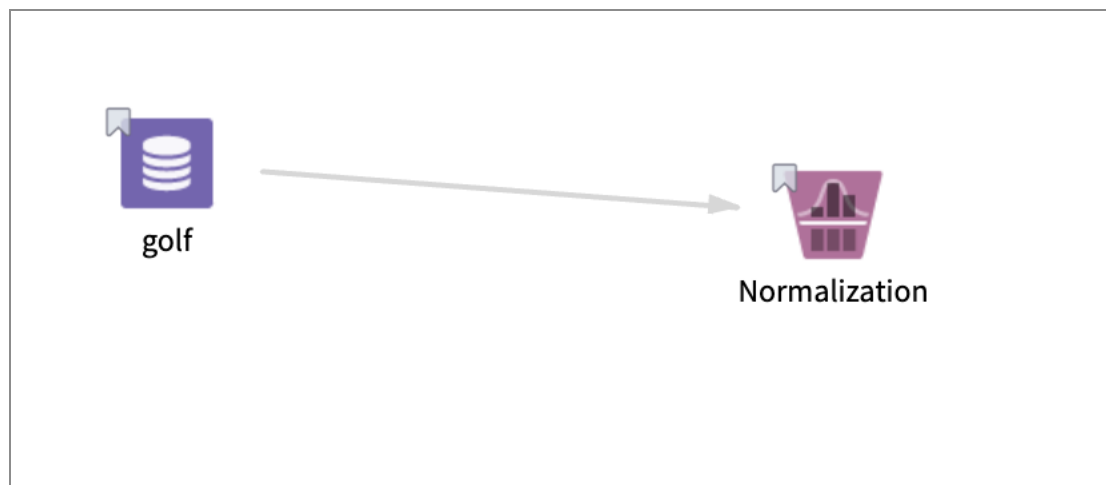
## Output

### Visual Output

- **Output:** A table that displays the output of a data set for the normalized data.

## Example

The following example displays the normalized data for the given data set using the Normalization operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Method:** Proportional Transformation
- **Columns:** outlook, humidity, wind, play, temperature
- **Store Results:** Yes

## Output

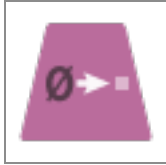
The following figure displays the output for the parameter settings for the **golf** data set.

Operation created output with 5 columns and 14 rows.

outlook	humidity	wind	play	temperature
sunny	85	f	no	0.0825
sunny	90	t	no	0.0777
overcast	78	t	yes	0.0806
overcast	65	t	yes	0.0621
sunny	95	f	no	0.0699
sunny	70	t	yes	0.067
overcast	90	t	yes	0.0699
rain	96	f	yes	0.068
rain	80	f	yes	0.066
rain	70	t	no	0.0631
rain	80	f	yes	0.0728
rain	80	t	no	0.0689
sunny	70	t	yes	0.0728
overcast	75	f	yes	0.0786

## Null Value Replacement

This operator replaces null values of the selected fields in the data set with designated values. It is helpful as a pre-cleansing data step.



## Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Input

An input is a single tabular data set.

## Configuration

The following table provides the configuration details for the Null value Replacement operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Replace Null</b>	Specify the columns to replace null values. For more information, see <a href="#">Null</a>

Parameter	Description
<b>Columns</b>	<a href="#">Value Replacement Configuration dialog</a> . <div> <b>Note:</b> For a column with the Boolean data type, the null data must be replaced with either <b>true</b> or <b>false</b>. </div>
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

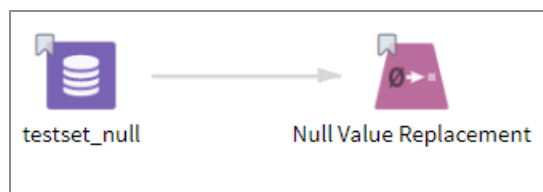
A table that displays the output of a data set replacing the null values with designated values.

### Data Output

A tabular data set of the newly created table or view.

## Example

The following example displays the data set created by replacing the null values with designated values for the given data set, forming a new table using the Null Value Replacement operator.



### Data

**testset\_null:** This data set contains the following information:

- Multiple columns namely colint, coldouble, colbool1, colbool2, colbool3, coldate1, coldate2, colts1, and colts2.
- Multiple rows (9 rows).

colint	coldouble	colbool1	colbool2	colbool3	coldate1	coldate2	colts1	colts2
1	4.6	0	false	F	2022/08/10	2022-08-10	2022-08-10 13:47:55.0	2022/08/10 13:47:55
N/A	4.7	1	true	T	2022/09/30	2022-09-30	2022-09-30 14:26:22.0	2022/09/30 14:26:22
3	N/A	0	false	f	2018/04/20	2018-04-20	2018-04-20 15:33:33.0	2018/04/20 15:33:33
4	4.8	N/A	true	t	2016/01/01	2016-01-01	2016-01-01 09:23:45.0	2016/01/01 09:23:45
5	4.9	1	N/A	f	2022/10/08	2022-08-10	2022-08-10 13:47:55.0	2022/08/10 13:47:55
6	22.23	0	true	N/A	2022/09/30	2022-09-30	2022-09-30 14:26:22.0	2022/09/30 14:26:22
7	44.45	1	false	f	2022/09/30	2018-04-20	2018-04-20 15:33:33.0	2018/04/20 15:33:33
8	50.5	0	true	t	2016/01/01	2016-01-01	2016-01-01 09:23:45.0	2016/01/01 09:23:45
9	1300000	0	true	t	2016/01/01	2016-01-01	2016-01-01 09:23:45.0	2016/01/01 09:23:45

## Parameter Setting

The parameter settings for the **testset\_null** data set are as follows:

- **Replace Null Columns**

Column Name	Value
colint	2
coldouble	0.54
colbool1	1

- **Store Results:** Yes

## Output

The following figure displays the output for the **testset\_null** data set.

Operation created output with 9 columns and 9 rows.

colint	coldouble	colbool1	colbool2	colbool3	coldate1	coldate2	colts1	colts2
1	4.6	0	false	F	2022/08/10	2022-08-10	2022-08-10 20:47:55.0	2022/08/10 13:47:55
2	4.7	1	true	T	2022/09/30	2022-09-30	2022-09-30 21:26:22.0	2022/09/30 14:26:22
3	0.54	0	false	f	2018/04/20	2018-04-20	2018-04-20 22:33:33.0	2018/04/20 15:33:33
4	4.8	1	true	t	2016/01/01	2016-01-01	2016-01-01 17:23:45.0	2016/01/01 09:23:45
5	4.9	1	N/A	f	2022/10/08	2022-08-10	2022-08-10 20:47:55.0	2022/08/10 13:47:55
6	22.23	0	true	N/A	2022/09/30	2022-09-30	2022-09-30 21:26:22.0	2022/09/30 14:26:22
7	44.45	1	false	f	2022/09/30	2018-04-20	2018-04-20 22:33:33.0	2018/04/20 15:33:33
8	50.5	0	true	t	2016/01/01	2016-01-01	2016-01-01 17:23:45.0	2016/01/01 09:23:45
9	1300000	0	true	t	2016/01/01	2016-01-01	2016-01-01 17:23:45.0	2016/01/01 09:23:45

## Pivot

This operator transforms the categorical data contained in a column of a table into columns of a new table, by means of subtotals (or other calculations) that can be defined by another column in the same list. The other calculations can be averages and counts.



## Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

For typical data entry and storage, data usually appears in flat tables such that it only consists of rows and columns. While such data contains a lot of information, it can be difficult to get the information summary. A pivot table helps quickly summarize the flat data, giving it depth, and highlighting the desired information.

The usage of a pivot table is extremely broad and depends on the situation. The first question to ask is "*What am I looking for?*" A pivot table usually consists of rows, columns, and data (or fact) fields. These fields allow several kinds of aggregations including sum, average, count, max, min, and so on.

The **Pivot Column** can only be categorical. In the output table, the **Pivot Column** is transformed into multiple columns, one for each category. To run this operator, you must define at least one variable with a categorical data type.

- The results are also grouped by a selected column.
- The values in the new columns are aggregates of a third column (or of the presence of the category if no aggregate column is chosen).
- The columns are listed (in the table, file, or array) in a descending order of the value of the category that they represent.

**i Note:** This operator can only connect to subsequent operators in certain situations - see the [Output](#) section for details.

## Input

An input is a single tabular data set.

### Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

The following table provides the configuration details for the Pivot operator.



Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Pivot Column</b>	Specify the column for pivot transformations. On database data sources, you can only select the categorical columns to pivot.
<b>Group By</b>	Specify the column to group by.
<b>Aggregate Column</b>	Specify the pivot column's value column.
<b>Aggregation</b>	Specifies the aggregate function. The following values are available: <ul style="list-style-type: none"> <li>• <b>Sum</b></li> <li>• <b>Average</b></li> <li>• <b>Count</b></li> <li>• <b>Max</b></li> <li>• <b>Min</b></li> </ul>
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

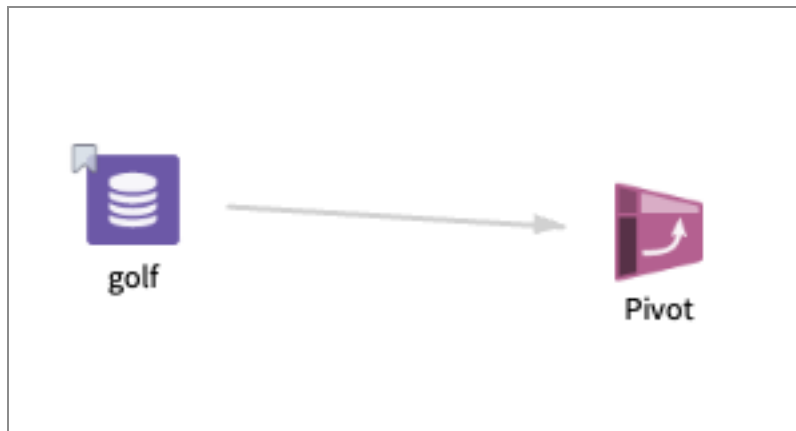
## Output

### Visual Output

- **Output:** A table that displays the output of a data set for the succeeding operator.

## Example

The following example demonstrates the Pivot operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Pivot Column:** play
- **Group By:** outlook
- **Aggregate Column:** temperature
- **Aggregation:** sum
- **Store Results:** Yes

### Output

The following figure displays the output for the parameter settings for the **golf** data set.

Operation created output with 3 columns and 3 rows.

outlook	play_yes	play_no
overcast	300	0
rain	213	136
sunny	144	237

## Reorder Columns

This operator reorders one or more columns from an input table or renames them.



### Information at a Glance

**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Input

An input is a single tabular data set.

## Configuration

The following table provides the configuration details for the Reorder Columns operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Ordered Columns</b>	Click <b>Define</b> to specify the columns (in order) to become the first columns in the output, and optionally specify a new name for each. For more information, see <a href="#">Ordered Columns dialog</a> .
<b>Columns to Keep</b>	Specify any other columns to keep in the output.
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Outputs

### Visual Output

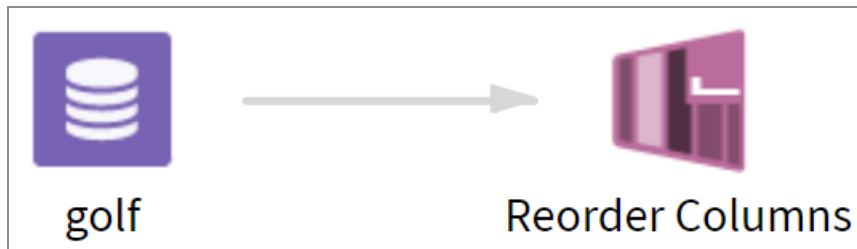
A table that displays the output of a data set.

### Data Output

A database table with reordered and (optionally) renamed columns.

## Example

The following example illustrates the Reorder Columns operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Ordered Columns:**

Columns	New Name
Wind	wd
humidity	hm
outlook	ol

- **Columns to Keep:** play
- **Store Results:** Yes

### Output

The following figure displays the output for the parameter settings for the **golf** data set.

Operation created output with 4 columns and 14 rows.

wd	hm	ol	play
true	65	overcast	yes
false	80	rain	yes
true	70	rain	no
false	70	sunny	yes
false	96	rain	yes
false	95	sunny	no
true	90	overcast	yes
true	80	rain	no
false	80	rain	yes
true	70	sunny	yes

## Row Cleanser

This operator removes the records according to the specified row completeness criteria.



### Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform

Parameter	Description
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

This operator applies a set of rules to remove incomplete rows. The user selects the columns to focus on and then a filtering condition is set. According to this condition, rows are selectively removed.

The number of null values in selected columns per each row is calculated. The input rules are applied so that the remaining rows have the desired limit of null columns.

## Input

An input is a single tabular data set.

## Configuration

The following table provides the configuration details for the Row Cleanser operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns to Use</b>	Specify the columns for checking the null values. Click <b>Select Columns</b> to select the required column.
<b>How many selected columns should be null before</b>	Specify the filtering limits to be calculated. The following values are available: <ul style="list-style-type: none"> <li>• <b>A number of columns</b></li> </ul>

Parameter	Description
<b>removing rows</b>	<ul style="list-style-type: none"> <li>• <b>A percentage of columns</b></li> <li>• <b>All</b></li> <li>• <b>Any</b></li> </ul> <p>Default: <b>All</b></p>
<b>Percentage(%) / Number of Columns</b>	<p>Specify the percentage or number of columns to calculate. If the previous parameter is set as <b>A percentage of columns</b>, specify the desired percentage. If set as <b>A number of columns</b>, specify the desired number. If the previous parameter is set as <b>All</b> or <b>Any</b>, this parameter is ignored.</p> <p>Default: <b>80</b></p>
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

A table that displays the output of a data set after removing the incomplete rows.

### Output to Successive operator

A single tabular data set with selected rows.

## Example

The following example displays the cleansed data for the given data set using the Row Cleanser operator.





## Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

## Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Columns to Use:** outlook, temperature, humidity
- **How many selected columns should be null before removing rows:** A percentage of columns
- **Percentage(%) / Number of Columns:** 80
- **Store Results:** Yes

## Output

The following figure displays the output for the parameter settings for the **golf** data set.

Operation created output with 5 columns and 14 rows.

outlook	temperature	humidity	wind	play
sunny	85	85	f	no
sunny	80	90	t	no
overcast	83	78	t	yes
overcast	64	65	t	yes
sunny	72	95	f	no
sunny	69	70	t	yes
overcast	72	90	t	yes
rain	70	96	f	yes
rain	68	80	f	yes

## Row Filter

This operator sets the criteria for filtering data set rows. Only the rows that meet the criteria remain in the output data sets.



## Information at a Glance



**Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform

Parameter	Description
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

You can specify row filters in the following modes:

- **Simple mode:** Use a simple template to define the filter, choosing a column, an inequality (for example, ">" or "between"), and a value (for example, a literal value or a column expression).
- **Script mode:** Specify any set of filters by using a script.

**i Note:** Column names in the script mode must be enclosed by a backtick ( ` ) if the column name contains special characters or multi-byte characters such as Chinese, Japanese, or Korean characters. Do not use double quotes to enclose the column names.

## Input

An input is a single tabular data set.

### Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Filter</b>	Specify the filters for the operator. See the <a href="#">Define Filter dialog</a> for more

Parameter	Description
	information.
	<b>Note:</b> The SQL expression must adhere to the SQL syntax supported by Spark since this is <a href="#">Spark SQL</a> .
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

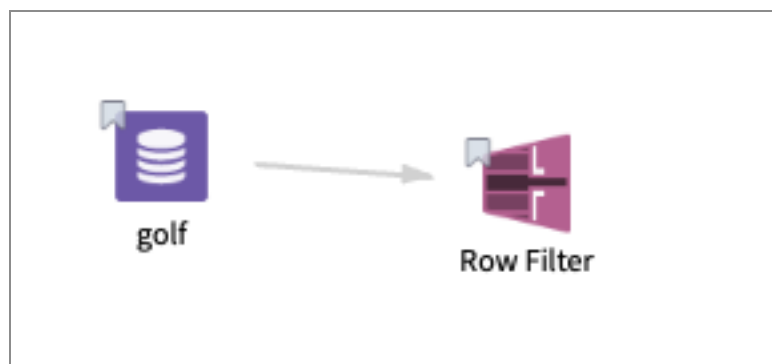
A table that displays the output (data rows) of a data set.

### Data Output

A tabular data set of the newly created table or a view.

## Example

The following example displays the data set rows created from data fields of the given data set, forming a new table using the Row Filter operator.



## Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

## Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Filter:** All conditions must be met
  - outlook, like, '%nny'
  - temperature, >, 80
- **Store Results:** Yes

## Output

The following figure displays the output for the parameter settings for the **golf** data set.

Operation created output with 5 columns and 1 rows.

outlook	temperature	humidity	wind	play
sunny	85	85	f	no

## Set Operations

This operator combines the results from merging two or more queries into a single result set.



## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Input

There must be two or more tabular data sets used as inputs.

**i Note:** For the current release (7.1.0), a maximum of two data sets are accepted as inputs.

## Restrictions

- The number of columns must be the same in all queries.
- The data types must be compatible.

## Configuration

The following table provides the configuration details for the Set Operations operator.

Parameter	Description
Notes	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the

Parameter	Description
	operator.
<b>Sets</b>	Click <b>Define Sets</b> to display the Define Sets dialog. For more information, see the <a href="#">Define Sets dialog</a> .
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

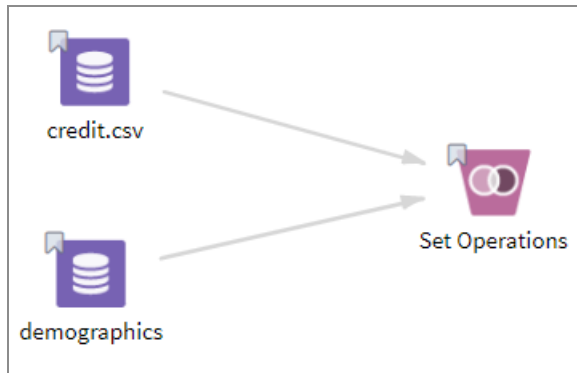
A table that displays the data rows of the output table, limited by the maximum display of rows and columns.

### Data Output

A tabular data set that displays the joined data sets.

## Example

The following example demonstrates the Set Operations operator where it combines the results of the queries into a single result set. Here, it includes all the distinct rows that belong to all queries in the union.



## Data

**Credit:** This data set contains the following information:

- Multiple columns namely id, times90dayslate, revolving\_util, debt\_ratio, credit\_lines, monthly\_income, times30dayslate\_2years, and srsdlqncy.
- Multiple rows (50,000 rows).

Operation created output with 8 columns and 50,000 rows. Visualization of rows limited to 25. Please refer to output table for full results.

id	times90dayslate	revolving_util	debt_ratio	credit_lines	monthly_income	times30dayslate_2years	srsdlqncy
2	0	0.26	0.324	4	1,956.5922	0	0
6	0	0.25	0.927	6	3,983.4564	0	0
52	0	0.46	0.372	7	3,595.3707	0	0
76	2	0.88	0.154	3	1,729.9645	2	0
84	0	0.14	0.69	7	2,507.1652	0	0

**demographics:** This data set contains the following information:

- Multiple columns namely ID, AGE\_IN\_YEARS, LEVEL\_OF\_EDUCATION, YEARS\_WITH\_CURRENT\_EMPLOYER, and YEARS\_AT\_CURRENT\_ADDRESS.
- Multiple rows (850 rows).

Operation created output with 5 columns and 850 rows. Visualization of rows limited to 25. Please refer to output table for full results.

ID	AGE_IN_YEARS	LEVEL_OF_EDUCATION	YEARS_WITH_CURRENT_EMPLOYER	YEARS_AT_CURRENT_ADDRESS
1	41	High school degree	8	21
2	41	Some college	18	6
3	41	Some college	7	6
4	41	High school degree	2	22
5	33	Did not complete high school	14	8

## Parameter Setting



The parameter settings for the Set Operations operator are as follows:

- **Sets:** UNION

Alias	credit	demographics
NEW_ID	id	ID

- **Store Results:** Yes

## Output

The following figure displays the output for the Set Operations operator. The result displays all unique identifiers (IDs) in both used datasets.

Operation created output with 1 columns and 50,732 rows. Visualization of rows limited to 25. Please refer to output table for full results.

NEW_ID
26
29
474
65
191
418
541
558
222

## Unpivot

This operator accepts one or more columns along with a list of columns and generates a row for each column specified in the list.



## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

The selected columns are removed from the input and are transformed into the following two new columns at the end of the output data set.

- First column - Values are the names of the selected columns.
- Second column - Values are the corresponding values in the selected columns.

## Input

An input is a single tabular data set.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Columns</b>	Specify the columns to unpivot. All data types are supported.
<b>Name of</b>	Specify the name of the first new column. This contains the names of the

Parameter	Description
<b>Variable Column</b>	<p>columns to unpivot.</p> <p><b>Note:</b> The value must be alphanumeric. (Regular expression to match are: <code>"^[A-Za-z]+\ \ w*\$"</code>)</p>
<b>Name of Value Column</b>	<p>Specify the name of the second new column. This contains the values of the columns to unpivot.</p> <p><b>Note:</b> The value must be alphanumeric. (Regular expression to match are: <code>"^[A-Za-z]+\ \ w*\$"</code>)</p>
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

If you select  $X$  columns to unpivot from input with  $Y$  columns and  $N$  rows, the output data set has  $(Y-X+2)$  columns and  $(X * N)$  rows.

### Visual Output

A table that displays the output of a data set after unpivoting the columns.

### Data Output

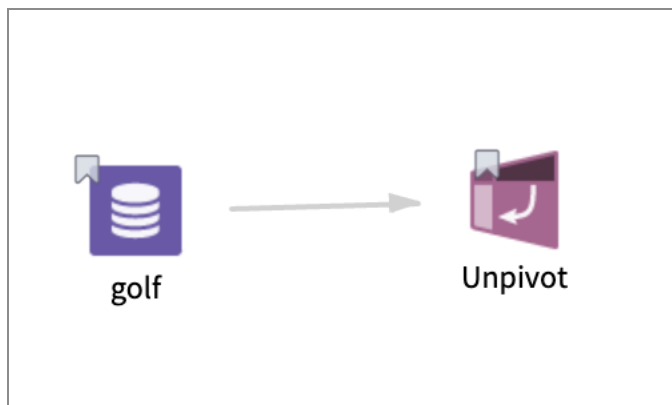
A tabular data set of the newly created table or view.

**Note:**

- The **New Variable** column contains the names of the unpivoted values in chararray format.
- The following conditions are applicable for the **New Value** column:
  - If all columns selected to unpivot are numeric, the resulting value column is double.
  - If all columns selected to unpivot are datetime with the exact same format, the resulting value column is datetime with this same format.
  - For all other cases, the resulting value column is chararray.
- All null values are kept in the output.

## Example

The following example displays the data set created by unpivoting the columns, forming a new table using the Unpivot operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Columns:** humidity, temperature
- **Name of Variable Column:** Variable\_Column
- **Name of Value Column:** Value\_Column
- **Store Results:** Yes

## Output

The following figure displays the output for the parameter settings for the **golf** data set.

outlook	wind	play	Variable_Column	Value_Column
overcast	t	yes	humidity	65
sunny	f	no	temperature	72
sunny	f	no	humidity	95
sunny	t	yes	temperature	69
sunny	t	yes	humidity	70
overcast	t	yes	temperature	72
overcast	t	yes	humidity	90
rain	f	yes	temperature	70
rain	f	yes	humidity	96
rain	f	yes	temperature	68
rain	f	yes	humidity	80
<b>rain</b>	<b>t</b>	<b>no</b>	<b>temperature</b>	<b>65</b>
rain	t	no	humidity	70
rain	f	yes	temperature	75
rain	f	yes	humidity	80
rain	t	no	temperature	71
rain	t	no	humidity	80
sunny	t	yes	temperature	75

## Variable

This operator is used to define variables created from data fields of the input data set, forming a new table or view.



## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

**! Important:** The created variables are static in nature. They cannot dynamically change during runtime.

You can also use the Variable operator to divide the data rows into quantiles, adding quantile variables to the data. Dividing the data into smaller and smaller divisions (quantiles) provides an understanding of the overall data distribution patterns.

**i Note:** Column names in the script mode must be enclosed by a backtick ( ` ` ) if the column name contains special characters or multi-byte characters such as Chinese, Japanese, or Korean characters. Do not use double quotes to enclose the column names.

## Input

An input is a single tabular data set.

### Bad or Missing Values

Null values are not allowed and result in an error.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Variables</b>	<p>Specify the expression(s) to create the new Variable column(s). Click <b>Define Variables</b> to define the variables. For more information, see <a href="#">Define Variables dialog</a>.</p> <p><b>Note:</b> The SQL expression must adhere to the SQL syntax supported by Spark since this is <a href="#">Spark SQL</a>. For information on the syntax, see <a href="#">Spark SQL Syntax and Expressions</a>.</p>
<b>Quantile Variables</b>	<p>If the new variable to create is a quantile variable, select the required column(s) to use for deriving the quantiles.</p> <p>The possible quantile types are <b>Average Ascend</b> (which automatically creates the bins) and <b>Customize</b> (which manually defines the variable bins).</p> <p><b>Note:</b> Automated variable binning can run on only databases that support NTILE, such as Greenplum, PostgreSQL, Oracle, and SQL Server. It is not supported on databases such as Teradata, MySQL, and MariaDB that do not currently support NTILE.</p> <p>For more information, see <a href="#">Define Quantile Variables dialog</a>.</p>
<b>Columns</b>	Specify the columns to include. For more information, see <a href="#">Select Columns dialog</a> .
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.



Parameter	Description
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

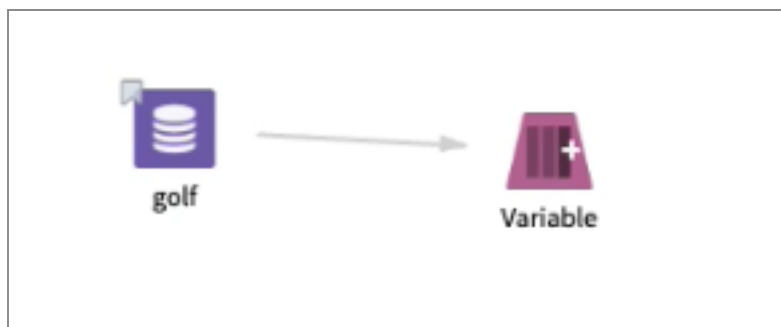
A table that displays the output of a data set. To see all of the data rows in addition to the derived variables, select all columns for the **Columns** parameter.

### Data Output

A tabular data set of the newly created table or view.

## Example

The following example displays the variables created from data fields of the given data set, forming a new table using the Variable operator.



### Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

### Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Variables:** test

- **Columns:** temperature, humidity, wind
- **Store Results:** Yes

## Output

The following figure displays the output for the parameter settings for the **golf** data set.

Operation created output with 4 columns and 14 rows.

temperature	humidity	wind	test
85	85	f	170
80	90	t	170
83	78	t	161
64	65	t	129
72	95	f	167
69	70	t	139
72	90	t	162

## Wide Data Variable Selector - Chi Square/Anova

This operator produces a new data set with chi-square or anova results with the significance statistics for each predictor (X) variable against a user-specified dependent (Y) variable from a very large data set, that is, the number of variables could be large - thousands or millions.



## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

For each predictor (X) variable, the operator computes the correlation against the dependent (Y) variable. For each predictor (X) variable, the operator computes the chi-square or one-way analysis of variance (ANOVA) results against the dependent (Y) variable. For the categorical dependent variables, you can calculate the chi-square analysis, and for the continuous dependent variables, you can calculate the analysis of variance (ANOVA). Here, all the predictors are treated as categorical. If continuous predictors exist, they are converted to categorical predictors using a binning procedure before the results are calculated. The algorithm makes two passes through the data, one to collect the dependent values and another to calculate the correlations.

**i Note:** For this operator, the predictor variables are always treated as categorical. In case you have the continuous dependent as well as continuous predictors, then use the [Wide Data Variable Selector - Correlations](#) operator.

The scalability should not be limited by anything other than available cluster resources.

## Input

An input is a single tabular data set that contains key-value pairs of variables and values in a stacked format, with the variable names (`vars`), continuous value (`con_vals`), categorical values (`cat_vals`), and the row id (`id`) columns. If the variable is continuous, then the values of the variable `cat_vals` should be null, and if the variable is categorical, then the values of the variable `cont_vals` should be null.

## Bad or Missing Data

One of the continuous values and categorical values variables always have missing values - this is as expected based on the structure logic of an input data. If both of them

are empty, the point is not used in the analysis calculations for the variable in question. In other words, statistics are calculated from all available and not the missing pairs of values separately for each predictor, not dependent on missing values in other predictors.

### Error and Exception Handling

The operation checks for the validity of the dependent variable specification. See the **Algorithm** section for more information.

- If the dependent variable is categorical, then it should be in a categorical values column and have discrete values (string, long, int).
- If you have continuous predictors as well as continuous dependent variables, then use the [Wide Data Variable Selector - Correlations](#) operator.

### Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Variable Name</b>	Specify the name of the dependent variable against which the chi-square is computed or the dependent continuous variable against which anova analysis is computed. If the dependent variable and predictors are continuous, then use the <a href="#">Wide Data Variable Selector - Correlations</a> operator.
<b>Variables Column</b>	Specify the name of the column where the variable names are carried. It should contain the name of the dependent variable and predictors.
<b>Continuous Values Column</b>	Specify the column that contains values for continuous variables.
<b>Categorical Values Column</b>	Specify the column that contains values for categorical variables.

Parameter	Description
<b>Row ID Column</b>	Specify the name of the column that contains the row ID numbers.
<b>Number Of Bins</b>	Specify the number of bins used for the discretization of continuous predictors. The bin boundaries are equidistant.  Default: <b>10</b>
<b>Chi Square Output</b>	Specify the output. If you have a continuous dependent variable, select <b>Anova</b> , if you have a categorical dependent variable, select <b>Chi-Square</b> or <b>Chi-Square with p values</b> . The following values are available: <ul style="list-style-type: none"> <li>• <b>Anova</b></li> <li>• <b>Chi-Square</b></li> <li>• <b>Chi-Square and p values</b></li> </ul> Default: <b>Chi-Square</b>
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

A tabular preview of the output data set that includes the **Output** and **Summary** tabs.

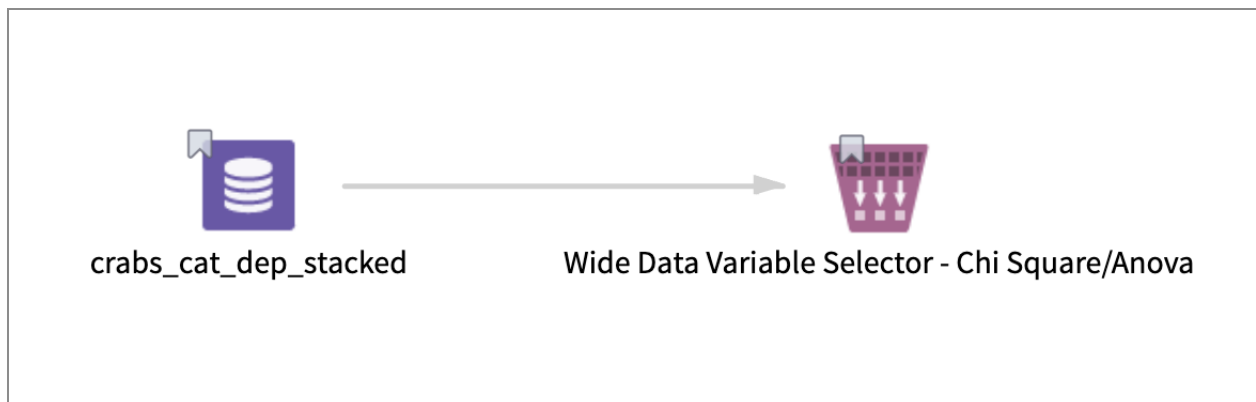
- **Summary:** The default summary that includes the parameters selected and their values.
- **Output:** A single tabular data set containing requested output statistics and the related significance levels for each predictor.

## Data Output

A single tabular data set that contains requested output statistics and the related significance levels for each predictor.

## Example 1

The following example shows the calculation of the chi-square test for each predictor variable against a user-specified dependent variable using the Wide Data Variable Selector - Chi Square/Anova operator.



## Data

A data set contains data in a stacked format where variable names are in the `vars` column, and values of these variables are in `con_vals` or `cat_vals` columns based on the type of variable. The dependent variable for this example is `SATELLTS`, which is a categorical variable.

vars	con_vals	cat_vals	id
COLOR	N/A	dark	122
SPINE	N/A	bothgood	131
SATELLTS	N/A	0	133
COLOR	N/A	darkmed	76
SATELLTS	N/A	0	125
COLOR	N/A	darkmed	17
SPINE	N/A	bothworn	109
SPINE	N/A	bothworn	149
SATELLTS	N/A	2	93
SPINE	N/A	bothworn	104

### Parameter Setting

The parameter settings for this analysis are as follows:

- **Dependent Variable Name:** SATELLTS
- **Variables Column:** vars
- **Continuous Values Column:** con\_vals
- **Categorical Values Column:** cat\_vals
- **Row ID Column:** id
- **Number Of Bins:** 10
- **Chi Square Output:** Chi-Square and p values
- **Store Results:** Yes

### Results

The following figures display the output results, one table with a summary of the parameters of analysis and the other with actual analysis results. The dependent variable SATELLTS is available in the output results, this represents the result where the SATELLTS versus SATELLTS test is conducted, and the p-value is 0, which means we are declining the hypothesis that both variables are independent.

### Summary

Parameter	Value
Dependent Variable Name	SATELLTS
Variables Column	vars
Continuous Values Column	con_vals
Categorical Values Column	cat_vals
Row ID Column	id
Number Of Bins	10
Chi Square Output	Chi-Square and p values

### Output

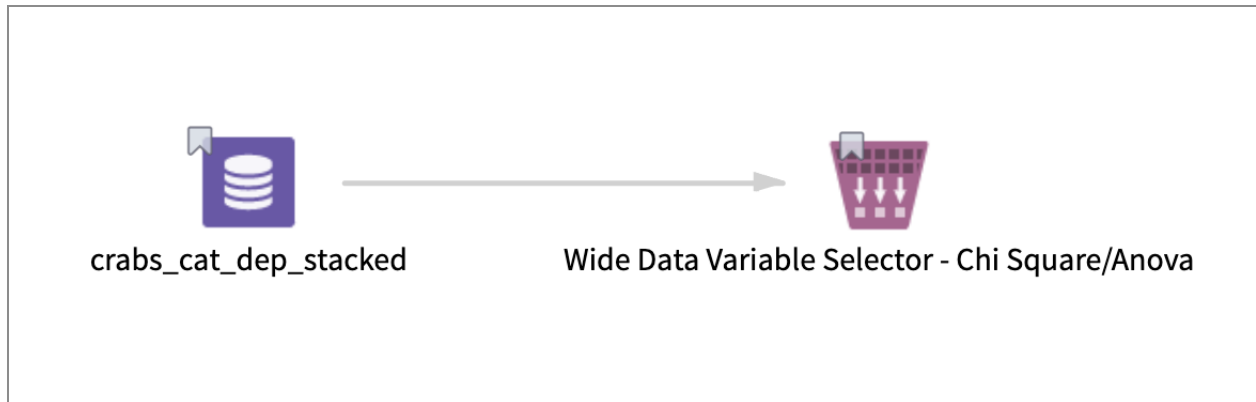
Operation created output with 4 columns and 6 rows.

Variable	Chi_Square	Row_count	p_value
SATELLTS	2422	173	0
WIDTH	140.063	173	0.185
WEIGHT	125.8579	173	0.4868
CATWIDTH	120.7688	173	0.0592
COLOR	44.6646	173	0.3605
SPINE	26.1474	173	0.5649



## Example 2

The following example shows the calculation of one-way ANOVA analysis separately for each predictor. The dependent variable is a user-specified continuous variable. The computation is done using the Wide Data Variable Selector - Chi Square/Anova operator.



### Data

A data set contains data in a stacked format where variable names are in the `vars` column, and the values of these variables are in `con_vals` or `cat_vals` columns based on the type of variable. Here, the continuous dependent variable `WIDTH`.

Variable_Column	con_value	cat_value	row_number
WIDTH	29	N/A	5
WIDTH	26	N/A	8
WEIGHT	2.625	N/A	172
WEIGHT	1.9	N/A	7
WEIGHT	1.7	N/A	147
WEIGHT	2.2	N/A	78
WEIGHT	2.8	N/A	99
CATWIDTH	22.75	N/A	138
WIDTH	28.2	N/A	4
WIDTH	26.7	N/A	36

### Parameter Setting and Results

The following figures of the resulting output also show the parameter settings of the operator. The anova results are calculated where the dependent variable is WIDTH and the rest of the variables are predictors (continuous ones are converted for the purpose of this analysis into categorical variables with a defined number of bins).

### Summary

Parameter	Value
Categorical Values Column	cat_value
Continuous Values Column	con_value
Variables Column	Variable_Column
Chi Square Output	Anova
Row ID Column	row_number
Dependent Variable Name	WIDTH
Number Of Bins	10

### Output

Operation created output with 4 columns and 6 rows.			
Variable	F	Row_count	p_value
CATWIDTH	521.39323	173	0
WIDTH	513.70364	173	0
WEIGHT	71.09506	173	0
SPINE	7.47123	173	0.00078
COLOR	4.5324	173	0.00439
SATELLTS	3.01504	173	0.00039

The large values of F statistic and significant p-value mean that the behavior of the dependent variable in groups of the predictor variable is significantly different. This means that there is a higher likelihood that the difference observed is real and not caused by chance.

## Wide Data Variable Selector - Correlations

This operator produces a new data set with correlations and significance statistics for each predictor (X) variable against a user-specified dependent (Y) variable from a very large data set, that is, the number of variables could be large - thousands or millions.



### Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

### Algorithm

For each predictor (X) variable, the operator computes the correlation (Pearson correlation coefficient) against the dependent (Y) variable. If categorical predictors exist, they are converted to continuous predictors using impact coding before the correlations are calculated. The algorithm makes two passes through the data, one to collect the dependent values and another to calculate the correlations.

**i Note:** For this operator, the dependent variable must be continuous. If your dependent variable is categorical, then use the [Wide Data Variable Selector - Chi Square/Anova](#) operator.

Use the following formula to calculate the  $t$  statistic for testing the statistical significance of the relationship and corresponding  $p$  value calculations.

$$\text{Test statistic: } t^* = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}}$$

$$\text{P Value: } p = 2.0 * (1.0 - \text{StudentsT}(n-2).cdf(abs(t^*)))$$

The scalability should not be limited by anything other than available cluster resources.

## Input

An input is a single tabular data set that contains key-value pairs of variables and values in a stacked format with `variable_names`, `continuous_values`, `categorical_values`, and `row_id` columns. The `variable_names` include the names of all variables (dependent variable as well as predictors), and values for these variables are either in `continuous_values` or `categorical_values` columns (this depends on variable type).

## Bad or Missing Data

One of the `continuous_values` and `categorical_values` variables always have missing values - this is as expected based on the structure logic of an input data. If both `continuous_values` and `categorical_values` variables are empty, then the point is not used in correlation calculation for the variable in question. In other words, correlations are calculated from all available, and not missing pairs of values separately for each predictor, not dependent on missing values in other predictors.

## Error and Exception Handling

The operation checks for the validity of the dependent variable specification. See the **Algorithm** section for more information.

- If the dependent variable is continuous, then it should be in a continuous values column and have numeric values (double, float, long, int).
- If the dependent variable is categorical, then use the [Wide Data Variable Selector - Chi Square/Anova](#) operator.

If there is a lack of enough cases to calculate correlation for a variable (at least 2), then the operation returns **NaN**.

If there is a lack of enough cases to calculate the  $t$  statistic and  $p$  value (at least 3), then the operation returns **0** and **1** respectively.

## Configuration

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Dependent Variable Name</b>	Specify the name of the dependent variable against which the correlation is computed. The dependent variable must be continuous. If the dependent variable is categorical, then use the <a href="#">Wide Data Variable Selector - Chi Square/Anova</a> operator.
<b>Variables Column</b>	Specify the name of the column where the variable names are carried. It should contain the name of the dependent variable and predictors.
<b>Continuous Values Column</b>	Specify the column that contains continuous predictors and dependent variable values.
<b>Categorical Values Column</b>	Specify the column that contains the categorical predictor values.
<b>Row ID Column</b>	Specify the name of the column that contains the row ID numbers.
<b>Number of Folds</b>	Specify the number of folds used in cross-validated impact coding. The value ranges between <b>2</b> and <b>98</b> .  Default: <b>2</b>
<b>Threshold for Grand Mean Replacement</b>	Specify an integer threshold value below which the dependent's mean is used as an impact coding value. The value ranges from <b>0</b> to the maximum integer value.  Default: <b>1000</b>
<b>Random Seed</b>	The seed used for the pseudo-random generation. The value ranges from <b>0</b> to max integer value.
<b>Correlation</b>	Specify the method to use to compute the correlation. It can be either

Parameter	Description
<b>Computation</b>	<b>Spark</b> or <b>TDS</b> . Default: <b>Spark</b>
<b>Correlation Output</b>	Specify the correlation output. The following values are available: <ul style="list-style-type: none"> <li>• <b>Correlations</b></li> <li>• <b>Correlations and t-statistics</b></li> </ul> Default: <b>Correlations</b>
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

A tabular preview of the output data set that includes the **Output** and **Summary** tabs.

- **Summary:** The default summary that includes the parameters selected and their values.
- **Output:** A single tabular data set containing correlations for each predictor (optionally with t-statistics and its significance).

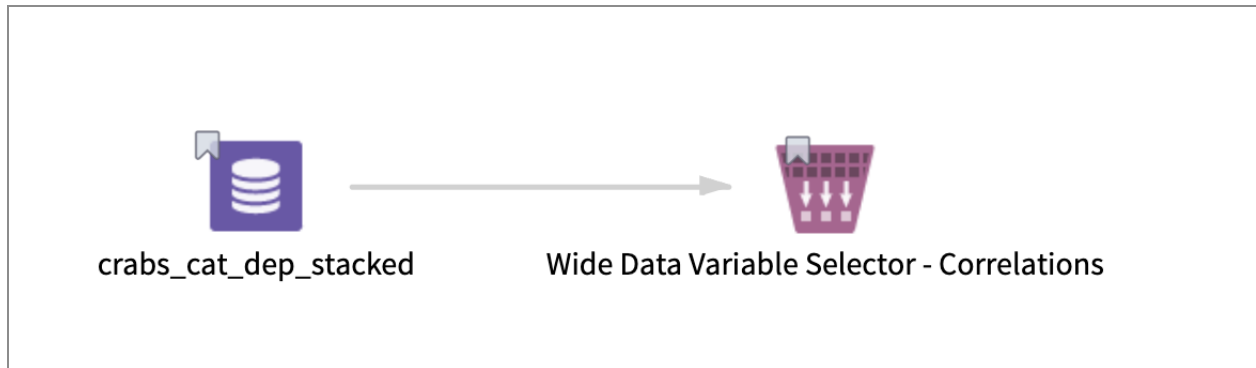
### Data Output

A single tabular data set that contains correlations of the dependent variable with each predictor (optionally with t-statistics and its p-value).

## Example

The following example produces the data set created by correlating each predictor variable against a user-specified dependent variable using the Wide Data Variable Selector -

Correlations operator.



## Data

A data set contains data in a stacked format where variable names are in the `vars` column, and the values of these variables are in `con_vals` (continuous\_values) or `cat_vals` (categorical\_values) columns based on the type of variable. The dependent variable in this example is the WIDTH variable.

vars	con_vals	cat_vals	id
WIDTH	29	N/A	5
WIDTH	26	N/A	8
WEIGHT	2.625	N/A	172
WEIGHT	1.9	N/A	7
WEIGHT	1.7	N/A	147
WEIGHT	2.2	N/A	78
WEIGHT	2.8	N/A	99
CATWIDTH	22.75	N/A	138
WIDTH	28.2	N/A	4
WIDTH	26.7	N/A	36
CATWIDTH	24.75	N/A	68
WEIGHT	1.6	N/A	106

## Parameter Setting

The parameter settings for getting correlations against the WIDTH variable are as follows:

- **Dependent Variable Name:** WIDTH
- **Variables Column:** vars
- **Continuous Values Column:** con\_vals
- **Categorical Values Column:** cat\_vals
- **Row ID Column:** id
- **Number of Folds:** 2
- **Threshold for Grand Mean Replacement:** 1000
- **Random Seed:** 0
- **Correlation Computation:** TDS
- **Correlation Output:** Correlations and t-statistics
- **Store Results:** Yes

## Results

The following figures display the results from a Wide Data Variable Selector - Correlations operator. You can see one table with the summary of parameters for analysis and one with the actual correlation results. The variable WIDTH is available in the results, even though it is not a predictor. The correlation coefficient of a WIDTH versus WIDTH is logically 1, t-statistics is high, and zero p-values prove that the relationship of variables is statistically significant.

## Summary



Parameter	Value
Dependent Variable Name	WIDTH
Variables Column	vars
Continuous Values Column	con_vals
Categorical Values Column	cat_vals
Row ID Column	id
Number of Folds	2
Threshold for Grand Mean Replacement	1000
Random Seed	0
Correlation Computation	TDS
Correlation Output	Correlations and t-statistics

## Output

Variable	Correlation	Row_count	t	p_value
WIDTH	1	173	3.4028e+38	0
CATWIDTH	0.9749	173	57.3151	0
WEIGHT	0.8869	173	25.1016	0
SPINE	-0.0959	173	-1.2596	0.2095
SATELLTS	-0.0959	173	-1.2596	0.2095
COLOR	-0.0959	173	-1.2596	0.2095

## Window Functions - Rank

This operator returns the rank of each row in relation to its windowed partition.



## Information at a Glance

**i Note:** This operator can only be used with TIBCO® Data Virtualization and Apache Spark 3.2 or later.

Parameter	Description
Category	Transform
Data source type	TIBCO® Data Virtualization
Send output to other operators	Yes
Data processing tool	TIBCO® DV, Apache Spark 3.2 or later

## Algorithm

The Window Functions-Rank operator supports the `rank`, `dense_rank`, `cumulative distribution`, and `ntile` (`n`: number of quantiles) functions. The core concept of this operator is to compute the rank or order of each row, relative to a defined grouping or partition. An example use case is ranking transactions for individual customers, within a data set that contains unique customers. In this example, the partitioning, or grouping, is the customer ID, while the data to rank or order is the transaction value amount. Transactions within each partition or customer ID are ranked and ordered starting with 1 for the highest, and counting up.

You can use this operator to compute database-like window functions by leveraging Spark SQL. To learn more about how window functions are implemented in Spark SQL.

A window function calculates the return value for every input row of an input based on a specific group of user-defined rows called the frame. Every input row has a unique frame associated with it.

**i Note:** Each operator can compute several window functions on several columns at once, but for a specific ordered partition. If you need to compute the same window functions on a different (ordered) partition, you can copy the operator and modify the partition, frame, or order-related parameters on the copy.

## Input

An input is a single tabular data set. It must have some numeric columns where it computes numeric aggregations and includes partition by column(s) and order by columns of any type.

### Bad or Missing Values

**Dirty data:** When parsing delimited data, the window functions operators remove dirty data (such as strings in numeric columns, doubles in integer columns, or rows with the incorrect number of values) as it parses. These rows are silently removed because Spark is not capable of handling them.

**Null values:** Before calculating any of the window functions, the operator filters any rows that contain null values in the **Order By** column selected. The number of rows removed due to null data is reported on the **Summary** tab of the visual output.

The Datetime columns must have the format specified in the input (for example, Datetime 'MM/dd/yy').

## Restrictions

**Wide data:** This operator works quickly on long data, but performance might slow down dramatically when window functions are calculated on thousands of columns. Increasing Spark's executor memory might improve performance.

**Datetime columns:** Input datetime columns must have the format specified in the input (for example, Datetime 'MM/dd/yy'); otherwise, the operator returns null values for the whole column.

## Configuration

The following table provides the configuration details for the Window Functions-Rank operator.

Parameter	Description
<b>Notes</b>	Notes or helpful information about this operator's parameter settings. When you enter content in the <b>Notes</b> field, a yellow asterisk appears on the operator.
<b>Partition By</b>	Specify the column(s) by which to partition. Click <b>Select Columns</b> to select the required columns.
<b>Order By</b>	Specify the column by which to order each partition (all data types are supported).
<b>Order</b>	Specify the order type for the selected <b>Order By</b> column.
<b>Calculate Row Number</b>	Specify whether the row number function returns a sequential (and unique) number starting at 1 within an ordered partition.
<b>Calculate Rank</b>	Specify whether the rank function returns a sequential number starting at 1 within an ordered partition.
<b>Calculate Dense Rank</b>	<p>Specify whether the dense rank function returns a sequential number starting at 1 within an ordered partition.</p> <div> <p><b>Note:</b> The difference between rank and dense rank is that dense rank leaves no gaps in the ranking sequence when ties occur (for example, if three values tie for second place, all three have a dense rank of 2, and the next value has a dense rank of 3).</p> </div>
<b>Calculate Cumulative Distribution</b>	<p>Specify whether the function returns the cumulative distribution of values within an ordered window partition; that is, the fraction of rows that are below the current row.</p> <p>If <math>N</math> = total number of rows in the partition and <math>V</math> = number of values before (and including) <math>x</math>, the equation would be <math>CUME\_DIST(X)=V/N</math>.</p>
<b>Calculate Quantiles</b>	<p>If <b>Yes</b>, returns the n-tile group ID (from 1 to <b>Number of Quantiles</b> inclusive) in an ordered window partition (equivalent to the NTILE function in SQL).</p> <p>For example, if <math>n = 4</math> (number of quantiles), the first quarter of the rows gets a value of 1, the second quarter gets 2, the third quarter gets 3, and the last</p>

Parameter	Description
	<p>quarter gets 4.</p> <p><b>Note:</b> If <b>Yes</b> is selected, you must specify an integer value for the <b>Number of Quantiles</b>.</p>
<b>Number of Quantiles</b>	If <b>Calculate Quantiles</b> is set to <b>Yes</b> , specify the number of quantiles to return for each ordered window partition. The value must be an integer > 0.
<b>Columns to Keep</b>	Specify the columns to keep in the output. Click <b>Select Columns</b> to select the required columns.
<b>Output Schema</b>	Specify the schema for the output table or view.
<b>Output Table</b>	Specify the table path and name where the output of the results is generated. By default, this is a unique table name based on your user ID, workflow ID, and operator.
<b>Store Results</b>	When set to <b>Yes</b> , the operator saves the results. If set to <b>No</b> , the operator does not save the results.

## Output

### Visual Output

The operator returns the visual output with two tabs: **Output** and **Summary**.

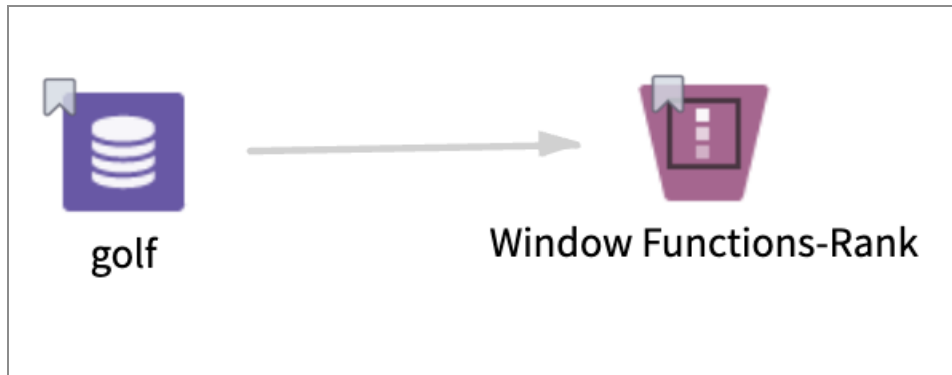
- **Output:** A table that displays the output of a data set.
- **Summary:** Displays information about the input parameters, their current settings, and a summary of the number of rows removed due to null data.

### Output to Successive operator

A data set that can be used by any TIBCO Data Science - Team Studio operator that accepts data set as input.

## Example

The following example illustrates the Window Functions-Rank operator.



## Data

**golf:** This data set contains the following information:

- Multiple columns namely outlook, temperature, wind, humidity, and play.
- Multiple rows (14 rows).

## Parameter Setting

The parameter settings for the **golf** data set are as follows:

- **Partition By:** play
- **Order By:** temperature
- **Order:** Ascending
- **Calculate Row Number:** Yes
- **Calculate Rank:** Yes
- **Calculate Dense Rank:** Yes
- **Calculate Cumulative Distribution:** Yes
- **Calculate Quantiles:** Yes
- **Number of Quantiles:** 2
- **Columns to Keep:** outlook, temperature, humidity, wind, play
- **Store Results:** Yes

## Results

These figures displays the results for the parameter settings for the **golf** data set.

## Summary

**Parameters selected**

Partition by: play  
 Order by: temperature  
 Order: Ascending  
 Calculate row number: Yes  
 Calculate rank: Yes  
 Calculate dense rank: Yes  
 Calculate cumulative distribution: Yes  
 Calculate quantiles: Yes  
 Number of quantiles: 2  
 Columns to keep (first 50): outlook,temperature,humidity,wind,play

**Output**

Input data size: 14  
 Input size after removing rows due to null values in 'Order By' column: 14  
 Rows removed due to null values in 'Order By' column: 0 rows (0%)

**Output**

outlook	temperature	humidity	wind	play	row_number	rank	dense_rank	cumulative_distribution	quantile
rain	65	70	t	no	1	1	1	0.2	1
rain	71	80	t	no	2	2	2	0.4	1
sunny	72	95	f	no	3	3	3	0.6	1
sunny	80	90	t	no	4	4	4	0.8	2
sunny	85	85	f	no	5	5	5	1	2
overcast	64	65	t	yes	1	1	1	0.1111	1
rain	68	80	f	yes	2	2	2	0.2222	1
sunny	69	70	t	yes	3	3	3	0.3333	1
rain	70	96	f	yes	4	4	4	0.4444	1
overcast	72	90	t	yes	5	5	5	0.5556	1
rain	75	80	f	yes	6	6	6	0.7778	2
sunny	75	70	t	yes	7	6	6	0.7778	2
overcast	81	75	f	yes	8	8	7	0.8889	2
overcast	83	78	t	yes	9	9	8	1	2

## Operator dialogs

Many operators share common dialogs for setting parameter configurations, defining variables, and other useful tasks.

For specific information about the properties and configurations you can set in a given dialog, review its help.

## Advanced Parameter Configuration dialog

When you specify the copy mode **Parallel**, you can define additional Sqoop settings using the Advanced Parameter Configuration dialog. If these parameters are left blank, the operator uses the TIBCO Data Science - Team Studio default Sqoop settings.

Label	Description
<b>Parameter Name</b>	<p>Sqoop parameter. Select from the list of available parameters.</p> <p>Enter Sqoop advanced parameters according to the <a href="#">Sqoop User Guide</a> rather than as a <a href="#">Sqoop CommandLineClient</a>.</p> <p>For example, use <code>--verbose</code> instead of <code>--verbose=true</code>.</p> <p>See <a href="http://sqoop.apache.org">http://sqoop.apache.org</a> for details.</p>
<b>Parameter Value</b>	<p>Value of the parameter to set for the parameter currently selected under <b>Parameter Name</b>.</p>
<b>Create</b>	<p>Click to add the new parameter key-value pair.</p>
<b>Delete</b>	<p>Click to remove a parameter.</p>

## Additional notes

### **sqoop.export.records.per.statement**

When you copy large or wide data sets, you must configure the `sqoop.export.records.per.statement` parameter. By default, `sqoop.export.records.per.statement` is set for 1000 records. You can override this default using the Advanced Parameters Configuration capability to allow 500 records to



be copied in the batch as follows: `scoop.export.records.per.statement=500`

This setting works only if batch copy is enabled on the database. It does not work for Vertica and Teradata databases.

You can specify these advanced parameters for the Parallel copy mode in the [Copy To Database](#) and [Copy To Hadoop](#) operators.

## Advanced Settings dialog

When Spark is enabled for an operator, you can apply the Automatic configuration for the Spark parameters, setting the default values to run the operator. However, you can edit these parameters directly.

To edit these parameters directly in the operator parameter dialog, select **No** for **Advanced Settings Automatic Optimization**, and then click **Edit Settings**. Set your desired configuration in the resulting Advanced Settings dialog.

Advanced Settings		
<a href="#">Spark Documentation</a>		<a href="#">Add Parameter</a>
Override?	Name	Value
<input type="checkbox"/>	Disable Dynamic Allocation	<input type="checkbox"/>
<input type="checkbox"/>	Number of Spark Executors	<input type="text"/>
<input type="checkbox"/>	Spark Executor Memory (MB)	<input type="text"/>
<input type="checkbox"/>	Spark Driver Memory (MB)	<input type="text"/>

**i Note:** Available options are determined by the type of operator. The following table shows the settings that apply for all operators for which you can enable Spark. For information about additional settings, see the specific operator help.

- If you check a checkbox from the **Override?** column, you can specify a value for the corresponding setting, which supersedes any default value set by your cluster or

workflow variables. If you provide no alternative value, the default value is used.

- If you click **Add Parameter**, you can provide custom Spark parameters. This option provides more control and tuning on your Spark jobs. See [Spark Autotuning](#) for more information.

Setting	Description
<b>Disable Dynamic Allocation</b>	<p>Select both checkboxes to indicate that idle CPU cores or execution memory should not be released for other applications.</p> <p>Dynamic allocation allows Spark to increase and decrease the number of executors as it needs them over the course of an application. If you can configure dynamic allocation on your cluster, it is probably most performant to do so.</p> <p>By default, dynamic allocation is disabled. TIBCO Data Science - Team Studio can use dynamic allocation only if the following conditions are true.</p> <ul style="list-style-type: none"> <li>• It is enabled in <code>alpine.conf</code>.</li> <li>• You have not set the number of executors.</li> <li>• Your cluster is correctly configured for dynamic allocation.</li> </ul>
<b>Number of Executors</b>	Specify the number of Spark executors to run this job ( <code>spark.executor.instances</code> ).
<b>Executor Memory in MB</b>	<p>Specify the Spark executor memory in megabytes.</p> <p>This value depends on the size of the data, the resources on the cluster, and the YARN container. TIBCO Data Science - Team Studio prevents you from setting this value higher than the size of the YARN container. Override this behavior by setting the <code>limit.spark.executor.memory</code> value in <code>alpine.conf</code> to <code>false</code>.</p>
<b>Driver Memory in MB</b>	<p>Specify the Spark driver memory, in megabytes.</p> <p>Some operators, such as Alpine Forest and Summary Statistics, pull a lot of information back to the driver, so these operators assign more driver memory.</p> <p>This value depends on the size of the data, the resources on the cluster and the YARN container, and the algorithm. TIBCO Data Science - Team Studio prevents you from setting this value higher than the size of the YARN</p>

Setting	Description
	container, even if this is set by the user. Override this behavior by setting the <code>limit.spark.executor.memory</code> value in <code>alpine.conf</code> to <code>false</code> .
<b>Number of Executor Cores</b>	<p>Specify the number of executor cores to use on each executor for the Spark job (<code>spark.executor.cores</code>).</p> <p>When this value is explicitly set, multiple executors from the same application can be launched on the same worker if the worker has enough cores and memory. Otherwise, each executor grabs all the cores available on the worker by default, in which case only one executor per application can be launched on each worker during one single schedule iteration. See the Spark documentation for more information.</p>

## Additional Information

See the [Spark Documentation](#) for more details on the available properties.

## Bin Configuration dialog

Use the Bin Configuration dialog to define the parameters for each bin, or interval, for a data set.

Setting	Description
<b>Histogram for</b>	Specifies the column for the histogram.
<b>Bin Type</b>	Specifies either the number of bins (using an integer between 2 and 99) or the bin width (using a float greater than zero).
<b>Minimum and Maximum</b>	Optional. Sets a range according to the provided values.
<b>new histogram</b>	Creates additional histograms for other columns in the data set.

For more information, see [Histogram](#).

## Choose Collapse Columns dialog

Use the Choose Collapse Columns dialog to define the rules for collapsing columns using the Collapse operator.

For more information, see [Collapse](#).

Setting	Description
<b>Column to Collapse</b>	Categorical type column to collapse.
<b>Aggregate</b>	Aggregate function to apply. Possible values: <ul style="list-style-type: none"><li>• <b>count</b></li><li>• <b>max</b></li><li>• <b>min</b></li><li>• <b>sum</b></li><li>• <b>avg</b></li><li>• <b>fixed</b></li></ul>
<b>Aggregation Column</b>	Column to aggregate.
<b>Column Alias</b>	Name for the collapsed column's aggregated value.


## Configure Columns dialog

The Configure Columns dialog configures delimiters, column names, and data types for a Hadoop file. You can access this dialog by clicking **Hadoop File Structure** in the Hadoop File properties dialog.

The Configure Columns dialog changes according to the **Hadoop File Format** setting chosen for the input file(s), corresponding to one of the following modes.

## Configure Columns: Text Files

The Configure Columns dialog changes options depending on the file type specified in the Hadoop File properties dialog. The options described in this topic are available for text files.

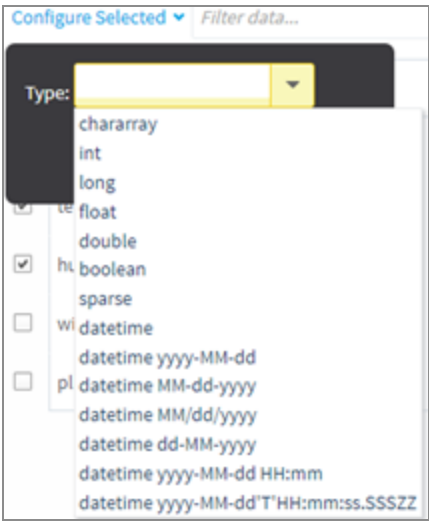
Column configuration	Description
Vertical/Horizontal File View	If the text file contains a large number of columns, then you can click the <b>switch</b> icon (  ), located in the top right corner, to change the display of the columns between vertical and horizontal. For files that have more than 300 columns, only the vertical view is available.
Escape and Quote Characters	Specify the escape and quote characters used in the file.
Delimiter	<p>Select the delimiter from the list.</p> <ul style="list-style-type: none"> <li>• <b>Comma</b></li> <li>• <b>Tab</b></li> <li>• <b>Semicolon</b></li> <li>• <b>Space</b></li> <li>• <b>Control-A</b></li> <li>• <b>Other</b> (Choosing <b>Other</b> specifies that a custom character is used as the delimiter.)</li> </ul>
Headers	<p>When TIBCO Data Science - Team Studio opens the Configure Columns dialog, TIBCO Data Science - Team Studio uses heuristics to determine if the first row of data is a header row, and selects or clears the control <b>First row contains header</b> based on this determination. You can select or clear this property manually.</p> <ul style="list-style-type: none"> <li>• If TIBCO Data Science - Team Studio determines that the first row contains header information, then the contents of the row are used as the default column names, and the setting <b>First row contains header</b> is selected.</li> <li>• If the source data does not have a header row, then clear <b>First</b></li> </ul>

Column configuration	Description
	<p><b>row contains header.</b></p> <ul style="list-style-type: none"> <li>If the file does not include headers, but the header information is available in a separate file, then you can set the header file. Click <b>Load header from file</b> and then browse to and select a file from the Hadoop file selector.</li> </ul>
Data Columns	<p>TIBCO Data Science - Team Studio attempts to infer the correct column names and data types by using a sample of the first few rows. When the dialog is displayed, each column is preceded by the inferred data type.</p> <p>You can change these settings by providing new column names and data types.</p> <p>The dropdown list box provides a list of standard data types.</p> <ul style="list-style-type: none"> <li><b>chararray</b></li> <li><b>int</b></li> <li><b>long</b></li> <li><b>float</b></li> <li><b>double</b></li> <li><b>bytearray</b></li> <li><b>sparse</b></li> <li><b>datetime</b></li> <li><b>datetimeyyyy-MM-dd'T'HH:mm:ss</b></li> <li><b>datetimeyyyyMMdd HH:mm</b></li> <li><b>datetimeyyyy-MM-dd</b></li> <li><b>datetimeHH:mm:ss</b></li> <li><b>datetimeyyyy-MM-dd'T'HH:mm:ss.SSSZ</b></li> <li><b>datetimeMM-dd-yyyy</b></li> <li><b>datetimeMM/dd/yyyy</b></li> </ul>

Column configuration	Description
----------------------	-------------

- `datetimedd-MM-yyyy`
- `datetimeyyyy-MM-dd HH:mm:ss`
- `datetimeyyyy-MM-dd'T'HH:mm:ss.SSSZZ`

You can change the data type for multiple columns. Set the view to horizontal format, select the checkboxes for the desired columns, and then click **Configure Selected**.



The list of columns can also be filtered with the filter field.

Column configuration	Description
	<p><b>Note:</b> For datetime data types, if the source data uses the ISO datetime format, you should select the basic <b>datetime</b> data type option to preserve the flexibility of the ISO formatting. ISO provides an international data exchange format framework for datetime data types that converts all datetime values into the number of milliseconds since 1970. For more details, see <a href="#">ISO DateTime Format</a>.</p> <p>If the source data is not in ISO datetime format, you must select from the list of predefined formats the specific datetime format of the imported data file.</p> <p>You can modify the list of specific datetime data type formats for the application using Datetime Format Preferences.</p> <p>Default datetime formats in TIBCO Data Science - Team Studio are listed in the dropdown list box.</p> <p>Although a list of datetime formats are pre-defined, you can override the defaults at run-time and specify a different datetime format (for a one-time Hadoop file import) using <a href="#">Joda-Time API</a> formatting.</p>

## Configure Columns: XML and JSON Files

For XML and JSON formatted data source files, you can choose at which level to parse and import the data.

## Configure Columns: Log Files

You can specify that the file type for a Hadoop file is a log file.

TIBCO Data Science - Team Studio supports reading log files for Apache Web Server and Log4J logging services. When you define the format, you can set any of the format variables as follows.



Variable	Value
<code>%v</code>	Canonical Server Name
<code>%h</code>	Remote Host
<code>%l</code>	Remote Log Name
<code>%u</code>	Username
<code>%t</code>	Time
<code> "%r "</code>	Referer
<code>%&gt;s</code>	Status
<code>%b</code>	Bytes Excluding HTTP Headers in CLF Format
<code> "%{Referer} "</code>	Referer
<code> "%{User-agent} "</code>	User Agent

## Define Column Aggregations dialog

The Define Column Aggregations dialog enables you to specify the group-by columns, aggregation type, and data type for an aggregate calculation.

Aggregation type options:

- sum
- count
- count distinct
- min
- max
- avg
- variance

- stddev
- custom

Data type options:


- BIGINT
- BOOLEAN
- BIT
- BIT VARYING
- CHAR
- DATE
- DOUBLE PRECISION
- NUMERIC
- INTEGER
- INTERVAL
- TIMESTAMP
- TIME
- VARCHAR

## Define Filter dialog

Defines the filter conditions.

You can specify row filters in two modes:

- Simple mode: Use a simple template to define the filter, choosing a column, an inequality (for example, ">" or "between"), and a value (for example, a literal value or a column expression).
- Script mode: Enter almost any set of filters by using SQL or Pig script.

 **Note:** If you switch from Simple mode to Script mode, your filters are converted to script. However, your script is lost if you switch from Script mode to Simple mode.

## Simple Mode

For Simple Mode, you can add multiple condition filters by clicking the **add new filter** link.

To define a filter, choose a column, a condition, and a value or expression.

AND/OR: You must also specify whether all filters must be met for each row (AND conditions), or just one of the filters (OR conditions).

## Simple Mode Conditions

Condition	Database	Hadoop
=	Yes	Yes
<>	Yes	Yes
>	Yes	Yes
<	Yes	Yes
>=	Yes	Yes
<=	Yes	Yes
contains	Yes	Yes
between	No	Yes
is null	Yes	Yes
is not null	Yes	Yes
is blank	No	Yes

- If your filter value is not numeric, you must add single quotes; for example, status = 'Active'.
- If your filter value is numeric or an expression, you do not need to use quotes (for example, start\_date = current\_date, or customer\_id > 37).

- If your filter value is a column and the column name contains spaces or uppercase characters, you must add double quotes (for example, "Start Date" - 1)

### **Script Mode**

In Script mode, filters are added as a "where" clause in the SQL or Pig scripting languages.

- You do not need to add the WHERE keyword.
- You can combine multiple filters with boolean expressions (AND, OR, and so on) and parentheses.
- Any expression that fits within a WHERE clause in SQL (for DB) or Pig (for HD) can be used.

**i Note:** For Hadoop data, if your filter value is a DateTime value, it can be compared to any other ISO-formatted DateTime Value. For details, see [ISO DateTime Format](#).

Additionally, for Hadoop, any DateTime Pig Function can be applied to a **DateTime** field, such as `GetMonth(datetime)` or `GetDay(datetime)`, and the result can then be used as Row Filter criteria.

You can find the full list of available DateTime-related Pig Functions at [Apache Pig DateTime Functions](#).

Because some Pig functions have issues handling null values, using the TIBCO Data Science - Team Studio alternatives for certain functions is recommended. Just append "Alpine" to the method name; for example, "DaysBetween" becomes "DaysBetweenAlpine".

Pig DateTime Functions	Alternative Pig DateTime Functions
AddDuration	AddDurationAlpine
SubtractDuration	SubtractDurationAlpine
MillisecondsBetween	MillisecondsBetweenAlpine
SecondsBetween	SecondsBetweenAlpine
MinutesBetween	MinutesBetweenAlpine
HoursBetween	HoursBetweenAlpine
DaysBetween	DaysBetweenAlpine
WeeksBetween	WeeksBetweenAlpine
MonthsBetween	MonthsBetweenAlpine
YearsBetween	YearsBetweenAlpine

## Define Filter dialog for Row Filter Operator

Defines the filter conditions.

You can specify row filters in two modes:

- Simple mode: Use a simple template to define the filter, choosing a column, an inequality (for example, ">" or "between"), and a value (for example, a literal value or a column expression).
- Script mode: Specify any set of filters by using a script.

**i Note:** If you switch from Simple mode to Script mode, your filters are converted to script. However, your script is lost if you switch from Script mode to Simple mode.

## Simple Mode

For Simple Mode, you can add multiple condition filters by clicking the **add new filter** link.

To define a filter, select a column, a condition, and a value or expression.

AND/OR: You must also specify whether all filters must be met for each row (AND conditions), or just one of the filters (OR conditions).

## Simple Mode Conditions

Condition
=
<>
>
<
>=
<=
contains

**Condition**

between

is null

is not null

is blank

- If your filter value is not numeric, you must add single quotes; for example, status = 'Active'.
- If your filter value is numeric or an expression, you do not need to use quotes (for example, start\_date = current\_date, or customer\_id > 37).
- If your filter value is a column and the column name contains spaces or uppercase characters, you must add double quotes (for example, "Start Date" - 1)

**Script Mode**

In Script mode, filters are added as a "where" clause.

- You do not need to add the WHERE keyword.
- You can combine multiple filters with boolean expressions (AND, OR, and so on) and parentheses.
- Any expression that fits within a WHERE clause can be used.

## Define Join Conditions dialog (Hadoop)

For Join operations in a Hadoop data source, you can specify conditions for the two sources, including whether to include all records in one or both data sources, and whether to use a Pig join script.

Parameter	Description
<b>Columns for Matching Rows</b>	From each of the datasets, select the matching columns and the conditions for the join.

Parameter	Description
<b>from Each Table</b>	<ul style="list-style-type: none"> <li>Click <b>Add Condition</b> to add additional rows and conditions.</li> <li>Click <b>Delete</b> to delete a condition.</li> </ul>
<b>Join Type</b>	<ul style="list-style-type: none"> <li>Specify when to include rows from each dataset, even if no matching row is found in the other dataset.</li> <li>Select <b>use Pig join script to execute join</b> to revert the defined join to a basic Pig-based join. <div> <p><b>Note:</b> Selecting this option disables the option for performing the replication in memory.</p> </div> </li> <li>Select the dataset for performing replication across nodes. This specifies whether to have the smaller dataset of the join replicated in memory for possible performance improvements. <div> <p><b>Note:</b> This makes sense only if both of the datasets are not large, and saving one in memory provides faster join results.</p> </div> </li> </ul>
<b>Output</b>	Click the <b>Join</b> to filter and display only the columns selected from <b>Input</b> .
<b>Input</b>	Select the dataset to populate the <b>Selected Fields for Output</b> .
<b>Selected Fields for Output File</b>	<ul style="list-style-type: none"> <li>Select output columns: Select the columns from each input table to include on the output table.</li> <li>Alias: You can assign a unique table alias for each input table by clicking and changing a column's alias field.</li> </ul>

## Define Pig Script dialog

Provides an interface to write the Pig statements for the operator, similar to the other "execute" script operators in TIBCO Data Science - Team Studio.

### Pig Input

The results of the preceding operator (or operators) are passed along as Pig relations (for example, `alpine_pig_input_1`) and a sample script is provided for convenience.



The Pig Execute Operator contains a reference to these Input Relations as well as lists of their column names.

There may be more than one input operator to a Pig Execute Operator. The names of the Input Relations (Alias) and Column Names are provided below the script area for ease of reference. You can double-click these labels to include them in the script.

## Pig Output

The output of the Pig Execute script should be stored in the location specified in the `Results Location` and `Results Name` parameters. This is the default value provided in the `alpine_pig_output` variable statement of the script.

**Attention:** STORE statements in a Pig script are not supported for saving output. Instead, use `alpine_pig_output = x;`

## Assistance with Writing Pig

Auto-completion hints are available while you edit the Pig script. Type CTRL-Space.

For additional help with Pig Script, the Pig Syntax link shows a short overview of common Pig syntax.

## Using the Define SQL dialog

Auto-completion hints are available while you edit the script. Type CTRL-Space.

# Define Quantile Variables dialog

To create a quantile variable, use this dialog to select the required column(s) for deriving the quantiles. You can define quantile variables from the Variable property dialog.

Option	Description
<b>Use approximation for average ascend (faster)</b>	Approximates the bin values for a faster, but not as accurate, binning process.
<b>Column Name</b>	From the dropdown list box, select a column from the dataset from which to derive the quantile variable column.
<b>Quantile Type</b>	From the dropdown list box, specify the type of quantile variable column

Option	Description
	<p>to create.</p> <ul style="list-style-type: none"> <li>• Select <b>Customize</b> to define the values manually that correspond to each quantile bin.</li> <li>• Select <b>Average Ascend</b> to divide the original column value range automatically across the specified number of bins.</li> </ul> <p>For categorical data (for example, text or dates), and for data sources that do not support NTILE, only the <b>Customize</b> type is available.</p>
<b>Create New</b>	<ul style="list-style-type: none"> <li>• Specify <b>true</b> to create a new column for the chosen variable in the output dataset.</li> <li>• Specify <b>false</b> to replace the original column with the variable in the output dataset.</li> </ul>
<b>Number of Bins</b>	<p>Available for <b>Quantile TypeAverage Ascend</b> only. Specify the number of bins in the quantile column.</p> <p>A bin is defined by a value range. Each bin is assigned with a bin number, incremented numerically beginning with 1. When the value of a data row for the original column falls in a certain bin value range, the quantile column is assigned the corresponding bin number.</p>
<b>Edit Bins</b>	<p>Available for <b>Quantile Type Customize</b> only. Click to display the dialog to define the bins manually.</p> <p>For numerical columns:</p> <ul style="list-style-type: none"> <li>• Select <b>Use Range</b> to define the start value (inclusive) and end value (exclusive) of the range. The end value must be larger than the start value. Ranges cannot overlap.</li> <li>• Clear <b>Use Range</b> to provide individual, comma-separated values for a bin.</li> </ul> <p>For Date, Time, and Timestamp data types, define the start date (inclusive) and end date (exclusive) for each bin.</p> <p>For a text data type, define the values to include in each bin. If the input table already exists, click <b>available values</b> to display the possible values.</p>

Option	Description
<b>Create</b>	Adds a row for specifying an additional quantile variable.
<b>Delete</b>	Deletes the quantile variable in the corresponding row.

## Define R Script dialog

The R Execute Script dialog provides an interface to write the R script for the operator, similar to the other TIBCO Data Science - Team Studio execute script operators.

To view auto-completion hints while you edit the script, type Ctrl+Spacebar.

## Define Sample Size dialog

When you specify either a percentage or a number of samples for the Random Sampling operator, you can set either the percentage or number for each of the samples.

Label	Description
<b>Sample</b>	Numbered rows, up to the value specified for <b>Number of Samples</b> in the Random Sampling properties dialog. See the help for Random Sampling ( <a href="#">Random Sampling (HD)</a> or <a href="#">Random Sampling (DB)</a> or <a href="#">Random Sampling</a> ).
<b>Sample Size (Percentage) or Sample Size (Rows)</b>	<p>The sample size for the corresponding row.</p> <ul style="list-style-type: none"> <li>If you specified <b>Percentage</b> in the Random Sampling properties dialog, then each entry is a percentage of the total number of samples requested. The total should equal 100%.</li> </ul> <div> <p><b>Note:</b> On smaller data sets, samples might not be exactly the percentage specified because of the random nature of the sampling operator. For each row in the table, the algorithm generates a random number. If that number is greater than a particular value, it writes the row (or rows, if using replacement.) Otherwise, it proceeds to the next row. This effect is mitigated in larger data sets.</p> </div>

Label	Description
	<ul style="list-style-type: none"> <li>If you specified <b>Number of Rows</b> in the Random Sampling properties dialog, then each entry is a number of rows from which to draw the sample.</li> </ul>

## Define Sets dialog

Defines the type of Set operations performed on two or more data sets.

Applies to the Set operator ( [Set Operations \(DB\)](#), [Set Operations \(HD\)](#), and [Set Operations](#) data sources).

Setting	Description
<b>Alias</b>	Name of the new merged output column in the Alias field.
<b>Type</b>	<p>Type of set operation. The options are as follows.</p> <ul style="list-style-type: none"> <li><b>UNION:</b> Combines the results of two or more queries into a single result set that includes all distinct rows that belong to all queries in the union. Duplicate rows are eliminated.</li> <li><b>UNION ALL:</b> Combines the results of two or more queries into a single result set that includes all rows that belong to all queries in the union. Duplicate rows are included.</li> <li><b>INTERSECT:</b> Returns all rows that are returned by both the query on the left side and the right side of the INTERSECT operand.</li> <li><b>EXCEPT:</b> Returns all distinct values from the result of the left query that are not found in the result of the right query.</li> </ul>
<b>Select Columns</b>	Columns from each input table to include on the output table. A column is displayed for each connected input data source.
<b>Map Columns</b>	Choose an option to automatically map the input data sets based on column name or order in the data sets. Using this option overwrites the current configuration.

Setting	Description
	<ul style="list-style-type: none"> <li>• <b>By the order in which they appear:</b> For example, the first column from each data set, the second column from each data set... up to the number of columns in the smallest data set.</li> <li>• <b>By name when appearing in all datasets:</b> Creates output columns by matching a column name from every data set. This operation is case insensitive.</li> <li>• <b>By name when appearing in more than one dataset:</b> Applies only to workflows with more than two input data sets. Create output columns by matching column names when that name occurs in two or more of the input data sets. This operation is case insensitive.</li> </ul> <p>When the Set operator combines two or more data sets based on DateTime columns, the more precise DateTime format is used and the output is converted into that one format.</p>

## Define SQL Statement dialog

The Define SQL Statement dialog provides an interface to write the SQL statements for the SQL Execute operator, similar to the other TIBCO Data Science - Team Studio execute script operators.

To view auto-completion hints while editing the script, press **Ctrl+Spacebar**.

The output of this operator should be assigned to a table called *alpine\_sql\_output* in the SQL script.

The following script is an example of a SQL script used in the [SQL Execute](#) operator. *alpine\_sql\_input\_1* refers to the data table fed into the SQL operator from an upstream operator, if one is present.

```
CREATE TABLE alpine_sql_output AS (SELECT
"county","state","msa","pmsa","popdensity","pop","popchange","age6574","
age75","crime","college",
"income","farm","democrat","republican","perot","white","black","turnou
t" FROM alpine_sql_input_1);
```

## Define Variables dialog

This dialog allows you to specify the expression to create new variable columns.

To add a variable, provide the following information:

- Variable Name - the name of the variable to be created.
- Data Type - the data type of the variable.
- SQL Expression - define the variable.

Using the Dialog

- To add another variable, select "Create." A new row is added to the variable table.
- To remove a variable, select "Delete" in the corresponding row.
- To modify an existing variable definition, edit any of the fields.
- To apply the expression to multiple columns, select the "multi" option.

## Defining Multiple Variables

Select the "multi" option to define an expression for multiple columns.

The original expression is populated from the first dialog.

## Define Variables Dialog - Hadoop

Specify the expression to create the new Variable column(s).

To add a variable, provide the following information:

- Variable Name - the name of the variable to be created.
- Data Type - the data type of the variable
- Pig Expression - the expression in Pig defining the variable.

Using the Dialog

- To add another variable, select "Create." A new row is added to the variable table.
- To remove a variable, select "Delete" in the corresponding row.
- To modify an existing variable definition, edit any of the fields.

- To apply the expression to multiple columns, select the "multi" option.

## Defining Multiple Variables - Hadoop

Select the "multi" option to define an expression for multiple columns.

The original expression is populated with from the first dialog.

## Edit Table Columns dialog

The Edit Table Columns dialog defines the structure of the data table to output from a SQL Execute or R Execute operator.

SQL and R are dynamic languages that can generate arbitrary output at runtime. Because many big-data technologies are static (that is, the types are known in advance), TIBCO Data Science - Team Studio must validate that the values specified in this dialog are the values SQL or R returns at runtime. This design ensures that in case of a mismatch, TIBCO Data Science - Team Studio can notify the user and advise them how to correct the problem.

When you first add a [SQL Execute](#) or [R Execute](#) operator and view the **Result Table Structure**, you see that it is empty. This result is expected because the SQL or R script has not run yet, and TIBCO Data Science - Team Studio does not yet know what column names should be in the output. After running the operator once, you can see that the column names from the *alpine\_output* variable are populated. To add more columns to your output, you must ensure that they are present in the script as well as in the **Result Table Structure** columns.

To use the Edit Table Columns dialog, provide the following.

Name	Description
<b>Column Name</b>	For each column, specify a valid column name.
<b>Column Type</b>	For each column, specify the data type for the associated <b>Column Name</b> .
<b>New Field</b>	Click add a new table column to the structure.
<b>Delete</b>	Click to delete the column.

Name	Description
<b>Down</b>	Adjust the position of the column downwards.
<b>Up</b>	Adjust the position of the column upwards.

## Input Table Mapping dialog

The operators that precede the Sub-Flow operator as inputs must map to the datasets that are the starting point for the sub flow. Use this dialog to map these elements.

The mapping is a one-to-one relationship between the inputs and the datasets in the sub flow. Corresponding tables should have the same columns as those used in the sub flow.

In the dropdown list, specify the sub-flow table dataset that maps to the input. See [Sub-Flow](#) for information.

## Interaction Parameters dialog

When you create a linear regression, you can specify available independent variables where you suspect that the data parameters might have a combined effect on the dependent variable.

Creating interaction parameters is useful when the modeler believes the combined interaction of two independent variables is not additive.



**Note:** For Hadoop only, if the preceding operator is Variable Selection, then this option is disabled, because the selected columns are determined by the variable selection results.

Setting	Description
<b>First Column</b>	Sets the first column to be combined.
<b>Interaction</b>	Defines the interaction between First Column and Second Column. Can be one of the following.



Setting	Description
	<ul style="list-style-type: none"> <li>• <b>*</b> - indicates the two selected variables should be defined as <math>a : b + a + b</math>. In other words, the two variables' values are multiplied together (<math>a:b</math>) and also added individually(<math>a, b</math>). <ul style="list-style-type: none"> <li>◦ For example, Gender * Adult might create a new variable with the values of <i>male, female, adult, juvenile, male adult, female adult, male juvenile, and female juvenile</i>.</li> </ul> </li> <li>• <b>:</b> - indicates the interaction of the two column variables are as follows. <ul style="list-style-type: none"> <li>◦ Interactions between numeric variables generate a new column that is the product of the variables.</li> <li>◦ Interactions between categorical variables (word strings) generate a column per category pair, such as male/female.</li> <li>◦ Interactions between numeric and categorical variables generate a new variable per category for analysis.</li> </ul> </li> </ul>
<b>Second Column</b>	Sets the second column to be combined.
<b>New field</b>	Adds a new row of interacting columns.
<b>Delete</b>	Deletes the corresponding row.

## Join Properties - Database dialog

From the TIBCO Data Science - Team Studio Join operator, specify settings to join multiple datasets that are stored in a database.

*Join Properties: Selected Columns for Output Table*

Setting	Description
<b>Output Table</b>	This section reflects the number of fields specified for the join output.
<b>Selected Columns for</b>	From the list, select checkbox for the columns to include in the output table. You can select any individual, all, or no columns from an individual

Setting	Description
<b>Output Table</b>	input table.  The input tables' names are enclosed in brackets immediately before the column name.
<b>Input Tables</b>	You can filter for a specific table using the <b>all</b> and <b>none</b> controls for columns listed under <b>Selected Columns for Output Table</b> .

*Join Conditions / Create Join Condition*

Setting	Description
<b>Left Table</b>	To create or update a join condition, you must specify a left table from the dropdown list box. The available tables are those listed under <b>Input Table</b> .
<b>Join Type</b>	<p>From the dropdown list box, specify the join type. Available SQL join types are:</p> <ul style="list-style-type: none"> <li>• <b>JOIN</b></li> <li>• <b>LEFT JOIN</b></li> <li>• <b>RIGHT JOIN</b></li> <li>• <b>FULL OUTER JOIN</b></li> <li>• <b>CROSS JOIN</b></li> </ul> <p><b>Join Type</b></p> <div> <p><b>Note:</b> If you are creating a multi-condition join, leave <b>Join Type</b> blank.</p> </div>
<b>Right Table</b>	<p>From the dropdown list box, select the table to join to the <b>Left Table</b>.</p> <p>A right table should be specified only once per join operator. If you are joining on multiple conditions, the right table must be specified only once. If you are joining more than two tables, a right table can appear only once in the join conditions.</p>
<b>Column1</b>	From the dropdown list box, select the column from the <b>Left Table</b> on which the join is performed.

Setting	Description
<b>Condition</b>	From the dropdown list box, select the join condition between <b>Column1</b> and <b>Column2</b> .
<b>Column2</b>	From the dropdown list box, select the column from the right table on which the join is performed.
<b>Multi-Connection</b>	If more than one join condition is necessary, from the dropdown list box, select <b>AND</b> or <b>OR</b> to combine with the following join condition.

## Key Columns dialog

When you perform random sampling, you can set an option for the operator to always create the same set of random rows for each sample data generation, and then you can specify the primary and secondary keys. Select the columns that the operator should use.

Setting	Description
<b>All/None</b>	<ul style="list-style-type: none"> <li>• <b>All</b> selects every column displayed in the list.</li> <li>• <b>None</b> clears every selected checkbox.</li> </ul>
<b>Filter data</b>	Type the name of a column to filter on that value. As you type, the list of columns is filtered.

## Null Value Replacement Configuration dialog (DB)

Specifies the columns for which to replace the null values. You can replace either null string values or null numeric values.

Setting	Description
<b>Defaults: Strings</b>	The provided string replaces all null string values in the selected columns. By default, "".

Setting	Description
<b>Defaults: Numerics</b>	The provided function replaces all null numeric values in the selected columns. By default, 0.
<b>All/None</b>	<ul style="list-style-type: none"> <li>• <b>All</b> selects every column displayed in the list.</li> <li>• <b>None</b> clears every selected checkbox.</li> </ul>
<b>Column Name</b>	Select the checkbox for the column name for which to replace null values. Only selected columns are available for null value replacement. When you select a column, the default value of the column's data type is displayed in the <b>Value</b> text box.
<b>Value</b>	<ul style="list-style-type: none"> <li>• The string value provided in the text box replaces any null string values in the selected column.</li> <li>• The fixed numeric value provided in the text box replaces any null numeric values in the selected column.</li> </ul> <p>If you provide a value of a data type that does not match the column data type, when you click <b>OK</b>, an error is displayed.</p>

## Replacing Null DateTime Values

If your selected null value column is a datetime data type, it can be compared to any other ISO formatted DateTime value.



**Note:** ISO provides an international data exchange format framework for DateTime data types that converts all datetime values into the number of milliseconds since 1970. Therefore, the DateTime data can be treated as a numeric column.

Specify the appropriate output data type format for the replacement DateTime value following Joda-Time API formatting. For details, see [ISO DateTime Format](#).

## Null Value Replacement Configuration dialog (HD)

Specifies the columns for which to replace the null values. You can replace either null string values or null numeric values.

Setting	Description
<b>Defaults: Strings</b>	The provided string replaces all null string values in the selected columns. By default, "".
<b>Defaults: Numerics</b>	The provided function replaces all null numeric values in the selected columns.
<b>All/None</b>	<ul style="list-style-type: none"> <li>• <b>All</b> selects every column displayed in the list.</li> <li>• <b>None</b> clears every selected checkbox.</li> </ul>
<b>Column Name</b>	Select the checkbox for the column name for which to replace null values. Only selected columns are available for null value replacement. Column names display the data type.
<b>Value or Function</b>	<ul style="list-style-type: none"> <li>• The string value provided in the text box replaces any null string values in the selected column.</li> <li>• The function selected from the dropdown list box replaces any null numeric values in the selected column. Can be one of the following. <ul style="list-style-type: none"> <li>◦ <b>Average</b></li> <li>◦ <b>Approximate Median</b></li> <li>◦ <b>Minimum</b></li> <li>◦ <b>Maximum</b></li> <li>◦ <b>Fixed value</b></li> </ul> </li> </ul> <p>The <b>Fixed value</b> option is not aggregated. Aggregated values can be calculated either for the entire dataset or for groups by selecting one or more columns for grouping.</p>
<b>Compute aggregates by</b>	Click <b>Select Columns</b> to specify which column(s) to aggregate by for null numeric values for the entire data set. See <a href="#">Select Columns dialog</a> for more information.

## Replacing Null DateTime Values

If your selected null value column is a datetime data type, it can be compared to any other ISO formatted DateTime value.

**i Note:** ISO provides an international data exchange format framework for DateTime data types that converts all datetime values into the number of milliseconds since 1970. Therefore, the DateTime data can be treated as a numeric column, and any of the aggregate value functions can be applied to replace a null DateTime value (Average Date).

Specify the appropriate output data type format for the replacement DateTime value following Joda-Time API formatting. For details, see [ISO DateTime Format](#).

## Null Value Replacement Configuration dialog

Specifies the columns to replace the null values. You can replace either null string values or null numeric values.

Setting	Description
<b>Defaults: Strings</b>	The provided string replaces all null string values in the selected columns. Default is "".
<b>Defaults: Numerics</b>	The provided function replaces all null numeric values in the selected columns. Default is 0.
<b>All/None</b>	<ul style="list-style-type: none"> <li>• <b>All:</b> Selects all the columns displayed in the list.</li> <li>• <b>None:</b> Clears the selected checkboxes.</li> </ul>
<b>Column Name</b>	Select the checkboxes for the column name for which to replace null values. Only selected columns are available for null value replacement. When you select a column, the default value of the column's data type is displayed in the <b>Value</b> text box.
<b>Value</b>	<ul style="list-style-type: none"> <li>• The string value provided in the text box replaces any null string values in the selected column.</li> <li>• The fixed numeric value provided in the text box replaces any null numeric values in the selected column.</li> </ul> <p>If you provide a value of a data type that does not match the column data type, an error is displayed when you click <b>OK</b>.</p>

## Replacing Null DateTime Values

If your selected null value column is a datetime data type, it can be compared to any other ISO formatted DateTime value.



**Note:** ISO provides an international data exchange format framework for DateTime data types that converts all datetime values into the number of milliseconds since 1970. Therefore, the DateTime data can be treated as a numeric column.

Specify the appropriate output data type format for the replacement DateTime value following Joda-Time API formatting. For details, see [ISO DateTime Format](#).

## Ordered Columns dialog

Specifies the new order of columns, and optionally their new column names, using the Reorder Columns operator.

Setting	Description
Columns	From the dropdown list box, select the column names to reorder.
Create	Creates a new row in the dialog so you can specify additional columns to reorder.
New Name	Optionally, change the name of the column.

Click the up and down arrows to change the order of its corresponding column. Click the X to delete the column from the reorder list.

## Results File Structure dialog

The Results File Structure dialog enables you to define the structure of the Hadoop data file(s) that are output.

If the file already exists (from a previous run, for example) the Output Hadoop File Structure dialog is prepopulated with the structure of this file.

**Important:** The Results File Structure dialog is related to the Hadoop File operator [Configure Columns dialog](#), which you access by clicking **Hadoop File Structure** in the Hadoop File operator properties.

## Select Columns dialog

When invoked from an operator properties dialog, the Select Columns dialog displays a list of columns available in the attached dataset. Select the columns that the operator should use.

**Note:** Some operators have restrictions on what columns are shown; some show all columns. Refer to the operator's documentation for more information. Selected columns are used by the operator.

Setting	Description
<b>All/None</b>	<ul style="list-style-type: none"> <li><b>All</b> selects every column displayed in the list.</li> <li><b>None</b> clears every selected checkbox.</li> </ul>
<b>Filter data</b>	Type the name of a column to filter on that value. As you type, the list of columns is filtered.

## Storage Parameters dialog

For operators that can generate an output table, the Storage Parameters dialog allows the user to specify several additional parameters regarding storage method and compression.

This helps avoid proliferation of excessively large quantities of data, and is a convenience feature that allows for re-use of similar workflows without changing constant values in numerous places.

**Important:** Currently, this feature is unique to Greenplum databases.



## Parameters

<b>Append Only?</b>	<p>Allows the operator's output data to be simply appended to the existing data source.</p> <p>Default value: No.</p>
<b>Columnar Storage?</b>	<p>Allows storage of the data in columns. This option is enabled only if <b>Append Only</b> is enabled.</p> <p>Default value: No.</p>
<b>Compression?</b>	<p>If the user selects Yes, he or she must set the level of compression as a number between 1 and 9. This option is enabled only if <b>Append Only</b> is enabled.</p> <p>Default value: No.</p> <p>When enabled, the default compression value is 1.</p>
<b>Distribution?</b>	<p>Determines how the dataset rows are sorted for the output column.</p> <p>Options:</p> <ul style="list-style-type: none"> <li>• <b>Distributed randomly</b></li> <li>• <b>Distributed by these columns</b></li> </ul> <p>If set to <b>Distributed by these columns</b>, a list of columns separated by commas must be provided for the distribution clause. In this case, you must enter a distribution clause as text, in the form of a list of columns separated by commas (the columns specified cannot be of type array).</p> <p>Default value: <b>Distributed randomly</b>.</p>
<b>Partition By</b>	<p>This option is available for Greenplum databases, allowing the data to be distributed on disk by table partitioning.</p> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p><b>Note:</b> Table partitioning addresses the problem of supporting very large tables, such as fact tables, by allowing you to divide them into smaller and more manageable pieces.</p> </div> <p>Specify the desired method for distributing the data on disk.</p>

## Store Intermediate Results

Many operators can be configured to store intermediate results in a database table or view, or in a Hadoop file.

### Database Operators

For database operators, the user can select whether to create intermediate results as a table or view, and where the results are created. The default values for the results locations use workflow variables to enable the user to make changes across the entire workflow without having to edit each individual operator.

**i Note:** Using views avoids creating large quantities of extra data, but the composition of many views in sequence might make it hard for the database to optimize the resulting query.

**Output Type**

☐ TABLE ☒ VIEW

**Output Schema**

@default\_schema ▼

**Output Table**

@default\_prefix\_agg\_0

**Storage Parameters**

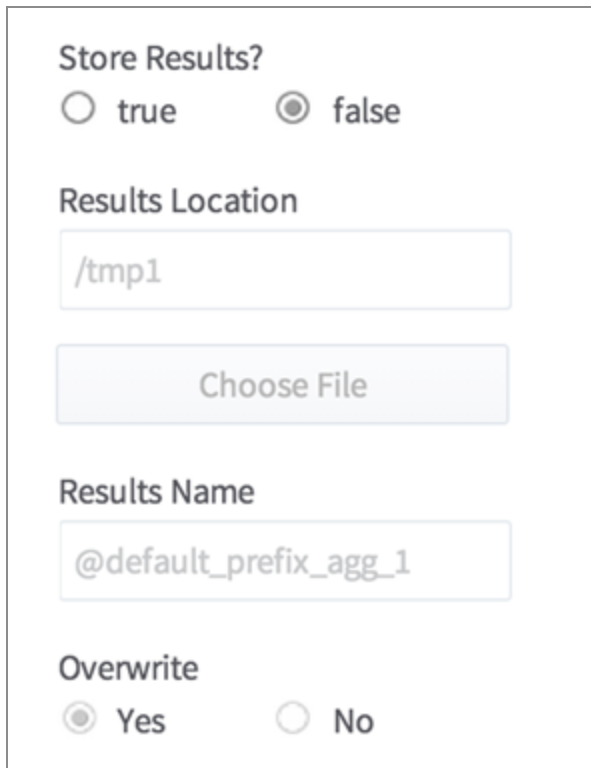
Storage Parameters

**Drop If Exists**

☒ Yes ☐ No

## Hadoop Operators

For Hadoop operators, the user is often able to choose whether to store intermediate results. The default values for the results location use workflow variables to enable the user to make changes across the entire workflow without having to edit each individual operator. The directory is based on the value of the `@default_tmpdir` workflow variable.



The screenshot shows a configuration dialog for a Hadoop operator. It contains the following elements:

- Store Results?**: Two radio buttons, 'true' and 'false'. The 'false' button is selected.
- Results Location**: A text input field containing the value '/tmp1'.
- Choose File**: A button located below the Results Location field.
- Results Name**: A text input field containing the value '@default\_prefix\_agg\_1'.
- Overwrite**: Two radio buttons, 'Yes' and 'No'. The 'Yes' button is selected.

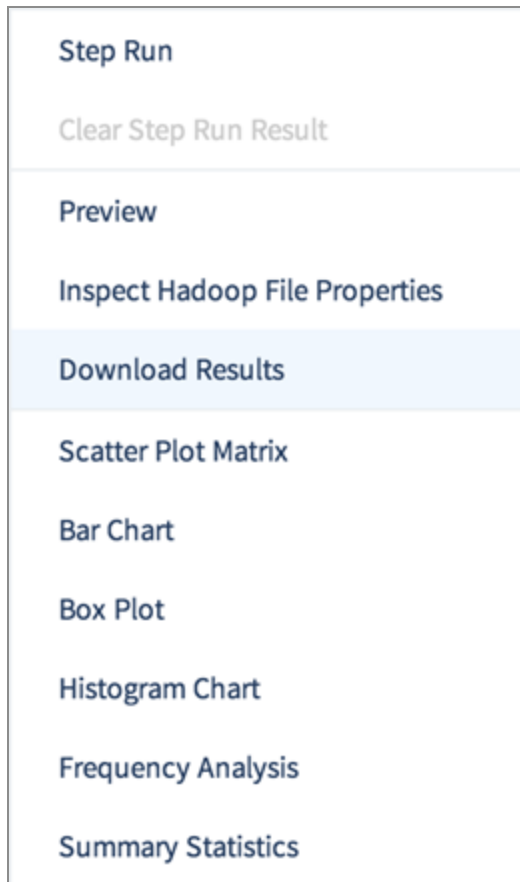
In certain cases, an operator can only be connected to another operator if its **Store Results?** parameter is set to True. This is because certain operators (for example, most modeling operators) must work on files rather than an intermediate result set. TIBCO Data Science - Team Studio automatically sets **Store Results?** to be True when this is required.

Store results activated for operator:  
Aggregation-5

## Downloading Stored Results

When **Store Results?** is set to True for a particular operator, the user is able to select that operator and download the data results at any time, provided the flow has already been run.

Right-click on the operator and select **Download Results**:



**i Note:** When downloading, you can choose whether to include the header row in the download file.

Download Results	
Name	alp_agg_1
include header row	<input checked="" type="checkbox"/>
Start at line	1
Number of lines	

[Close](#)
[Download](#)

## Sub-Flow Variable dialog

Use the sub-flow variables to define the workflow variable values used in the sub-flow workflow.

The Sub-Flow Variable dialog provides a list of the variables defined in the sub-flow, including the standard variables, **@default\_tempdir**, **@default\_schema**, and **@default\_prefix**.

This dialog is identical to the Workflow Variables dialog, except that you cannot add or remove variables. You can only provide values.

The existing values defined for the variables in the sub-flow are shown as defaults. Changes to sub-flow variable values are stored in the Parent workflow XML, which allows default values to be stored in the originating sub-flow XML.

## Window Column Configuration dialog

The Window Column Configuration dialog enables you to configure window columns for an aggregate calculation.

Enter the name of the window column to create in the **Result Column** field, the window function expression in the **Window Function** field, and the window OVER specification for the window column in the **Window Specification** field.

You can choose the data type of the window function column in the **Data Type** dropdown list.

Data type options:

- BIGINT
- BOOLEAN
- BIT
- BIT VARYING
- CHAR
- DATE
- DOUBLE PRECISION
- NUMERIC
- INTEGER
- VARCHAR

Click **New field** to add another window column. Click **Delete** to remove a window column.

For more information about PostgreSQL window function calls, see [PostgreSQL Window Functions](#).



**Important:** The windowing functionality is not supported for generic JDBC data sources.

## Operator Compatibility

You can check which operators are compatible with the supported data source types and review operators that have been deprecated.

# AWS EMR Data Source and Operator Compatibility

The following list provides information and links for the Legacy operators in the TIBCO Data Science - Team Studio that you can use with a AWS EMR data source.

## Data Extraction Operators

[Copy To Database](#)

[Hadoop File](#)

## Exploration Operators

[Bar Chart](#)

[Box Plot](#)

[Correlation \(DB\)](#)

[Frequency](#)

[Histogram](#)

[Line Chart](#)

[Scatter Plot Matrix](#)

[Summary Statistics \(DB\)](#)

[Variable Selection \(DB\)](#)

## Transformation Operators

[Aggregation \(DB\)](#)

[Batch Aggregation](#)

[Collapse](#)

[Column Filter \(DB\)](#)

[Correlation Filter \(HD\)](#)

[Distinct \(HD\)](#)

[Fuzzy Join](#)

[Join \(DB\)](#)

[Normalization \(DB\)](#)

[Null Value Replacement \(DB\)](#)

[Numeric to Text \(DB\)](#)

[One-Hot Encoding](#)

[Pivot \(DB\)](#)

[Reorder Columns \(HD\)](#)

[Replace Outliers \(DB\)](#)

[Row Filter \(DB\)](#)

[Sessionization](#)

[Set Operations \(DB\)](#)

[Sort By Multiple Columns](#)

[Transpose](#)

[Unpivot \(DB\)](#)

[Unstack](#)

[Variable \(DB\)](#)

[Wide Data Variable Selector - Chi Square / Anova](#)

[Wide Data Variable Selector - Correlations](#)

[Window Functions - Aggregate](#)

[Window Functions - Lag/Lead](#)

[Window Functions - Rank](#)

## **Sampling Operators**

[Random Sampling \(DB\)](#)

[Resampling](#)

[Sample Selector](#)

## **Modeling Operators**

[Alpine Forest Classification](#)

[Alpine Forest Regression](#)



[ARIMA Time Series \(DB\)](#)  
[Association Rules](#)  
[Collaborative Filter Trainer](#)  
[Decision Tree](#)  
[Gradient Boosting Classification](#)  
[Gradient Boosting Regression](#)  
[K-Means \(HD\)](#)  
[Linear Regression \(HD\)](#)  
[Logistic Regression \(HD\)](#)  
[Naive Bayes \(HD\)](#)  
[Neural Network](#)  
[PCA \(HD\)](#)  
[SVM Classification](#)

### **Prediction Operators**

[Chi Square, Goodness of Fit](#)  
[Chi Square, Independence Test](#)  
[Classifier \(DB\)](#)  
[Collaborative Filter Predictor](#)  
[Collaborative Filter Recommender](#)  
[N-gram Dictionary Loader](#)  
[PCA Apply<sup>1</sup>](#)  
[Predictor \(DB\)](#)

### **Model Validation Operators**

[Alpine Forest Evaluator](#)  
[Classification Threshold Metrics](#)

---

<sup>1</sup>This operator is deprecated but not yet removed or replaced.

[Confusion Matrix](#)

[Goodness of Fit](#)

[Lift \(DB\)](#)

[Regression Evaluator \(DB\)](#)

[ROC](#)

[T-Test - Independent Samples](#)

[T-Test - Paired Samples](#)

[T-Test - Single Sample](#)

## **Tool Operators**

[Convert](#)

[Export](#)

[Export to Excel \(DB\)](#)

[Export to SBDF \(DB\)](#)

[Flow Control](#)

[HQL Execute](#)

[Load Model](#)

[Note](#)

[Pig Execute](#)

[Python Execute \(DB\)](#)

[R Execute](#)

[Sub-Flow](#)

## **Natural Language Processing (NLP) Operators**

[LDA Predictor](#)

[LDA Trainer](#)

[N-gram Dictionary Builder](#)


[Text Extractor](#)

[Text Featurizer](#)

## Deprecated, Removed, or Replaced Operators

Some operators are deprecated, removed, or replaced with improved operators. These older operators are no longer officially supported.

If you have a workflow with one of these removed operators and try to configure the operator, the following error message is displayed.

 **Important:** This operator is no longer supported. Please remove it from your workflow.

The tables below provide compatibility information and upgrade paths.

### Database

Operator name	Version deprecated	Version removed	Alternatives	Replacement status
Adaboost	5.8	6.1	Decision Tree	Future Roadmap
Alpine Forest	5.8	6.1	Alpine Forest Classification or Alpine Forest Regression on Hadoop	Complete <a href="#">Alpine Forest - MADlib</a> <a href="#">Alpine Forest Predictor - MADlib</a> (6.2.1)
Association Rules	5.8	6.1	K-Means or LDA	Complete <a href="#">Association Rules</a> (6.2.2)

Operator name	Version deprecated	Version removed	Alternatives	Replacement status
CART	–	6.1	Alpine Forest Classification or Alpine Forest Regression on Hadoop	Complete <a href="#">Decision Tree Regression - CART</a> <a href="#">Decision Tree Classification - CART</a> (6.2)
Classification Threshold Tuning	-	6.5	Classification Threshold Metrics	<a href="#">Classification Threshold Metrics</a>
Decision Tree	–	6.1	Alpine Forest Classification or Alpine Forest Regression on Hadoop	Complete <a href="#">Decision Tree Regression - CART</a> <a href="#">Decision Tree Classification - CART</a> (6.2)
EM Cluster	5.8	6.2		Future Roadmap
EM Cluster Predictor	5.8	6.2		Future Roadmap
LDA	5.8	6.2	K-Means	Available In NLP Package <a href="#">LDA Trainer</a>

Operator name	Version deprecated	Version removed	Alternatives	Replacement status
				(6.2)
LDA Predictor	5.8	6.2		Available In NLP Package <a href="#">LDA Trainer</a> (6.2)
Naive Bayes	–	6.1	Logistic Regression, SVM Classification	Complete <a href="#">Classification Modeling with Naive Bayes</a> (6.1)
Neural Network	5.8	6.1	Naive Bayes, Logistic Regression	Complete <a href="#">Neural Network</a> (6.3)
Time Series	–	6.1		Complete <a href="#">ARIMA Time Series (DB)</a> (6.3)

## Hadoop

Operator name	Version deprecated	Version removed	Alternatives	Replacement status
PCA Apply	6.6		<a href="#">Predictor (HD)</a>	Predictor
Time Series	–	6.1		<a href="#">ARIMA Time Series (DB)</a> - HD

## Processing Tools for Hadoop-Enabled Operators

The tables in this section outline which data processing tool is supported by each of the various HDFS-supported operators.

### Processing Tools for Data Load Operators (HDFS)

These HDFS-supported data load operators support the specified data processing tools.

Operator name	Pig	MapReduce	Spark	Sqoop
<a href="#">Copy To Database</a>	?	?		x (parallel mode)
<a href="#">Copy To Hadoop</a>				x
<a href="#">Hadoop File</a>		x		

### Processing Tools for Exploration Operators (HDFS)

These HDFS-supported data exploration operators support the specified data processing tools.

Operator name	Pig	MapReduce	Spark	Sqoop
<a href="#">Bar Chart</a>	x			
<a href="#">Box Plot</a>	x			
<a href="#">Correlation (DB)</a>		x		
<a href="#">Frequency</a>	x			
<a href="#">Histogram</a>	x			
<a href="#">Scatter Plot Matrix</a>	x			

Operator name	Pig	MapReduce	Spark	Sqoop
<a href="#">Summary Statistics (DB)</a>	x		x	
<a href="#">Variable Selection (DB)</a>		x		

## Processing Tools for Transformation Operators (HDFS)

These HDFS-supported data transformation operators support the specified data processing tools.

Operator name	Pig	MapReduce	Spark	Sqoop
<a href="#">Aggregation (DB)</a>	x			
<a href="#">Collapse</a>		x		
<a href="#">Column Filter (DB)</a>	x			
<a href="#">Distinct (HD)</a>		x		
<a href="#">Join (DB)</a>	x	x		
<a href="#">Normalization (DB)</a>	x			
<a href="#">Null Value Replacement (DB)</a>	x			
<a href="#">Pivot (DB)</a>		x		
<a href="#">Row Filter (DB)</a>	x			
<a href="#">Set Operations (DB)</a>		x		
<a href="#">Variable (DB)</a>	x			

## Processing Tools for Sample Operators (HDFS)

These HDFS-supported sample operators support the specified data processing tools.



**Note:** Operators marked with n/a do not use HDFS processing tools.

Operator name	Pig	MapReduce	Spark	Sqoop
<a href="#">Random Sampling (DB)</a>		x		

## Processing Tools for Modeling Operators (HDFS)

These HDFS-supported model operators support the specified data processing tools.

Operator name	Pig	MapReduce	Spark	Sqoop
<a href="#">Alpine Forest Classification</a>		x	x	
<a href="#">Alpine Forest Regression</a>		x	x	
<a href="#">Decision Tree</a>		x		
<a href="#">Gradient Boosting Classification</a>			x	
<a href="#">Gradient Boosting Regression</a>			x	
<a href="#">K-Means (HD)</a>		x	x	
<a href="#">Linear Regression (HD)</a>		x	x	
<a href="#">Logistic Regression (HD)</a>		x	x	
<a href="#">Naive Bayes (HD)</a>		x	x	
<a href="#">PCA (HD)</a>		x		
<a href="#">SVM Classification</a>		x		



## Processing Tools for NLP Operators (HDFS)

These HDFS-supported NLP operators support the specified data processing tools.

Operator name	Pig	MapReduce	Spark	Sqoop
<a href="#">LDA Predictor</a>			x	
<a href="#">LDA Trainer</a>			x	
<a href="#">N-gram Dictionary Builder</a>			x	
<a href="#">Text Extractor</a>			x	
<a href="#">Text Featurizer</a>			x	

## Processing Tools for Prediction Operators (HDFS)

These HDFS-supported prediction operators support the specified data processing tools.



**Note:** Operators marked with n/a do not use HDFS processing tools.

Operator name	Pig	MapReduce	Spark	Sqoop
<a href="#">Classifier (DB)</a>		x		
<a href="#">PCA Apply</a>		x		
<a href="#">Predictor (DB)</a>		x		

## Processing Tools for Model Validation Operators (HDFS)

These HDFS-supported tool operators support the specified data processing tools.

Operator name	Pig	MapReduce	Spark	Sqoop
<a href="#">Alpine Forest Evaluator</a>		x		
<a href="#">Confusion Matrix</a>		x		
<a href="#">Goodness of Fit</a>		x		
<a href="#">Lift (DB)</a>		x		
<a href="#">ROC</a>		x		

## Processing Tools for Tool Operators (HDFS)

These HDFS-supported tool operators support the specified data processing tools.

**i** **Note:** Operators marked with n/a do not use HDFS processing tools.

Operator name	Pig	MapReduce	Spark	Sqoop
<a href="#">Export</a>	n/a	n/a	n/a	n/a
Custom Operator*				
<a href="#">Flow Control</a>	x			
<a href="#">Pig Execute</a>	x			
<a href="#">Sub-Flow*</a>				

**i** **Note:** \*For the Flow Control and Sub-Flow Operators, the HDFS Data Processing tool used depends on the customer-specific implementation.

# Glossary

---

## A

---

### **Alpine model**

See TIBCO Data Science – Team Studio analytics model.

### **Analytic workflow**

A flow of connected operators. Created and developed in the Workflow Editor.

### **Associate a Dataset**

Associating a dataset makes that data available to the workspace under the Data tab. These are datasets that are not a part of the sandbox schema. You can use these datasets to import data that is not contained within the main data source of the workflow.

## C

---

### **Comment**

User can comment on anything in the Activity pane in the workspace overview.

### **Connect view**

It is a generated table created by a join or select statement on a database data source. It is stored locally on the application database.

### **Custom operator**

One can integrate their own algorithms and processes into the TIBCO Data Science – Team Studio analytics engine using the Custom Operator Framework. Custom operators are written using Scala or Java and can harness the power of Spark for advanced machine learning and transformations.

## D

---

### **Dashboard widget**

On the homepage of TIBCO Data Science – Team Studio after logging in, a user can customize their view to create a dashboard of important information. Click the gear

icon on the home page to customize widgets.

**Data source**

An external data provider, either a Hadoop database or a relational database.

**Data source states | Disabled**

This has been disabled by an administrator and cannot be used until it is turned back on. Users may not open workflows or run jobs using this data source.

**Data source states | Incomplete**

A user has begun to add this data source but has not completed the connection parameters. Users can save some data sources as incomplete. The data source is not usable until the rest of the data is provided.

**Data source states | Offline**

This indicates that this data source is having trouble connecting properly. Verify connection parameters and try again.

**Data source states | Online**

This has correct connection information and can be used in TIBCO Data Science - Team Studio.

**Dataset**

Datasets come from databases, HDFS, or uploaded files such as CSVs. Datasets are part of a workspace, and they are used within workflows, Touchpoints, and sandboxes in order to perform analyses.

**Design time**

The actions a person takes before running an operator or workflow. They are designing/customizing the operator's parameters. This is an important concept to understand when learning about custom operators.

---

**E**

---

**Exit operator**

An operator that is at the end of a subflow workflow and is used to connect output to the parent workflow.

---

**I**

---

**Insight**

Insights are pieces of information that are deemed important to a particular workspace or a workflow. User can add an insight directly or promote a note or a comment to an insight. One can also attach workflow results, files, datasets, and other files uploaded from the desktop to support the importance of the finding.

---

**J**

---

**Job**

A scheduled task that is created in a workspace. A job can be run on a regular time interval, or on demand. Jobs are useful for updating data automatically over time or run specific tasks overnight.

---

**L**

---

**Legacy Workflow**

Creates a workflow using the architecture of TIBCO Data Science – Team Studio version 6.6.0, including access to all custom operators built for version 6.6.0.

---

**M**

---

**Milestone**

A section of work that a team member or group of members works on. Milestones include a due date that enables the user to see at a glance the progress of a particular analytic project. Milestones are shown on the workspace page under the Milestones tab, and also on the workspace overview. One can also change the status of the project to one of the three available options: On Track, Needs Attention, or At Risk.

---

**N**

---

**New Workflow**

Creates a workflow using the architecture of TIBCO Data Science – Team Studio version 7.0.1, including access to all new operators developed to run in Apache Spark 3.2.

**Note**

User can make notes on a workspace or any workfile within that space. Notes show under the Activity pane in the workspace overview and are viewable to those with access to the workspace. Notes can be promoted to Insights and commented on by other user.

**Notes**

It can be promoted to Insights and commented on by other user.

**Notification**

Notifications alert the user of important changes in the application, such as job results or collaboration information. When someone adds another person to the workspace, that person gets a notification. Many types of activities can have notifications, which can be configured individually.

**O**

---

**Operator**

An operator encapsulates some algorithm or transformation within a workflow. Operators show up as a list on the sidebar of the application within the workflow editor, and can be dragged to the canvas and connected to other operators or data sources. They are one of the main building blocks for workflows (the other being data sources). User can filter the operators based on the intended operation such as Load, Explore, Transform, Model, Predict, and Tools.

**P**

---

**Parameter**

An option that you set to control an algorithm.

**R**

---

**Run**

Running a workflow runs the entire workflow, running all operators and data sources. Users cannot run a whole workflow with invalid operators. Alternatively, they can use a Step Run to run operators that have not run before. To rerun everything from scratch, select Clear Step Run Results from the contextual menu.

## S

---

### **Sandbox**

A place on the Data tab where a user can bring in their training schema and perform simple explorations with the help of visualizations. One can also create external views from the sandbox datasets. Sandboxes are only available for database data sources.

### **Shared account**

An account that can be used to provide access to a database data source for multiple members of an organization. This means that the user shares a single set of credentials for the data source.

### **Source operator**

An operator that starts a workflow. Contrast with terminal operator.

### **Step run**

User can run only the operators and data sources needed to get up to the selected operator for step run in a workflow. This allows for faster iteration so that the user does not need to run the entire workflow again.

### **Sub-flow**

User can use the sub-flow operator to run another workflow from the current workflow.

## T

---

### **Tag**

Tags can be used to categorize datasets or results within the application. Tags can be added to any work file. Selecting a tag brings up a view of all work files with that tag.

### **Team Studio analytics model**

A model that is generated from a Legacy Workflow and is saved as an .am file. This allows the portability of models from one workflow to the other without having to build the whole model again. The Alpine models are saved under the Work Files tab of a workspace.

**Terminal operator**

An operator that no other operator can directly follow in the workflow. Contrast with source operator.

**Touchpoint Catalog**

Displays the list of Touchpoints published by the team members.

**Publish a Touchpoint**

Adds a Touchpoint to the catalog so that it is available to all the members of the application. Depending on the Touchpoint's settings, it can be run either as the creator or the person.

**Unpublish a Touchpoint**

Removes a Touchpoint from the catalog so that it is no longer available publicly for people in the application to view or run them. It is restricted to the members of the workspace from where the Touchpoint originated.

**Touchpoint**

Touchpoint wrap the functionality of complex workflows in an interactive application that can be consumed by the business analyst.

**W**

---

**Workfile**

Any file that is within the Work Files section within the workspace. Work files can be workflows, SQL files, CSVs, Touchpoints, Alpine models, or result files saved to the workspace. In addition, a user can upload files (such as images or .zip files) to make them part of the workspace.

**Workflow**

A collection of datasets and operators that perform analytic tasks. A workflow is where data scientists build out models using the available operators and algorithms in TIBCO Data Science – Team Studio.

**Workflow variable**

User can override default parameters and define their own workflow-wide variables using a workflow variable. Workflow variables can be used to set default paths and parameters for HDFS, configure operator parameters, and enable input via



Touchpoints. To set the workflow variables, click Actions > Workflow Variables from the workflow user interface. All workflow variables start with the character @.

**Workspace**

Workspaces allow team members to collaborate on a data science project.

Workspaces hold work files and can also have scheduled jobs, milestones, and associated database and Hadoop datasets under the Data tab to keep a track of progress on a project. A workspace can be either public or private. User can create a workspace from the workspace page.

# TIBCO Documentation and Support Services

---

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

## How to Access TIBCO Documentation

Documentation for TIBCO products is available on the [Product Documentation website](#), mainly in HTML and PDF formats.

The [Product Documentation website](#) is updated frequently and is more current than any other documentation included with the product.

## Product-Specific Documentation

The following documentation for TIBCO® Data Science - Team Studio is available on the [TIBCO® Data Science - Team Studio Product Documentation](#) page:

- *TIBCO® Data Science Team Studio Release Notes*
- *TIBCO® Data Science Team Studio Installation and Administration*
- *TIBCO® Data Science Team Studio User Guide*
- *TIBCO® Data Science Team Studio Security Guidelines*
- *TIBCO® Data Science Team Studio API Documentation*
- *TIBCO® Data Science Team Studio Web Help*

## How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our [product Support website](#).
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the our [product Support website](#). If you do not have a username, you can

request one by clicking **Register** on the website.

## How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, go to [TIBCO Community](#).

# Legal and Third-Party Notices

---

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of Cloud Software Group, Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, Spotfire, and Alpine Data Labs are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (<https://www.cloud.com/legal>) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: <https://scripts.sil.org/OFL>

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "README" FILES.

This and other products of Cloud SG may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at <https://www.tibco.com/patents>.

Copyright © 2017-2023. Cloud Software Group, Inc. All Rights Reserved.