



Spectrum Technology Platform

Spectrum Machine Learning Guide

Version 2020.1.0



Table of Contents

1 - Introduction

Spectrum Machine Learning.....	4
A Spectrum Machine Learning Workflow.....	5

2 - Machine Learning Stages

Enterprise Designer Stages.....	7
Flow Designer Stages.....	37

3 - Machine Learning Model Management

Accessing Machine Learning Model Management.....	49
Model Assessment.....	50
Binning Management.....	57
Configuration Settings.....	58

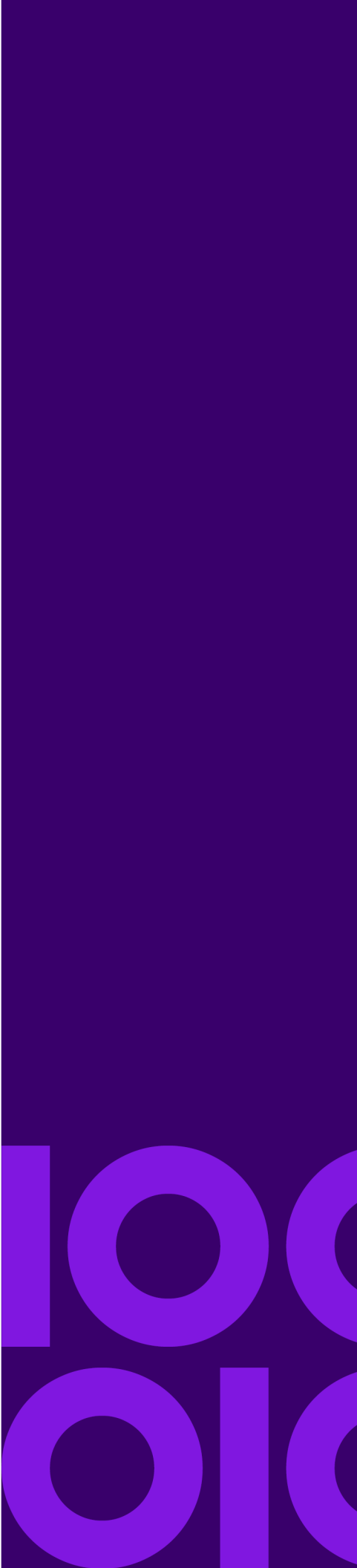
4 - Data Science Demonstration Flows

Introduction.....	62
Supervised Learning: Loan Default Prediction....	62
Unsupervised Learning: Segmentation.....	63

1 - Introduction

In this section

Spectrum Machine Learning.....	4
A Spectrum Machine Learning Workflow.....	5



Spectrum Machine Learning

Spectrum Technology Platform Machine Learning provides the ability to group (bin) numeric data and fit supervised and unsupervised machine learning model data in those models.

Note: Spectrum Machine Learning is supported only on Windows and Linux operating systems.

Note: Spectrum Machine Learning uses an underlying H2O.ai library for modeling algorithms in K-Means Clustering, Linear Regression, Logistic Regression, Principal Component Analysis, Random Forest Classification, and Random Forest Regression.

Binning

Binning divides records into groups (bins) for a continuous variable without taking into account objective information. You can perform unsupervised binning in one of two ways: using equal-width bins or equal-frequency bins.

K-Means Clustering

K-Means Clustering creates models based on analytical clustering, which segments a set of records into clusters of similar records based on data values.

Linear Regression

Linear Regression performs machine learning by creating models from datasets that use continuous objectives with input variables.

Logistic Regression

Logistic Regression creates models from datasets that use binary objectives with input variables.

Principal Component Analysis

Principal Component Analysis is a statistical process that converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables known as principal components.

Random Forest Classification

Random Forest Classification performs machine learning by creating models from datasets that use continuous objectives with input variables.

Random Forest Regression

Random Forest Regression performs machine learning by creating models from datasets that use binary objectives with input variables.

Machine Learning Model Management

Machine Learning Model Management includes Model Assessment, which enables you to manage all machine learning models on your Spectrum Technology Platform server, and Binning Management, which enables you to manage all binning on your Spectrum Technology Platform server.

A Spectrum Machine Learning Workflow

A typical machine learning workflow includes the following steps that take place in one or more dataflows:

1. Access the data using other Spectrum modules, such as Spectrum Data Federation.
2. Prepare the data using stages from other Spectrum modules such as those in Spectrum Data Federation, Data Quality, and the core modules.
3. Fit a machine learning model, run the dataflow, and then review the contents of the Model Output tab in the model stage. You can then tweak the model if necessary and rerun the dataflow. Following that, you need to review the full set of model assessment output in Machine Learning Model Management tool. You can review one model at a time or compare two models.
4. Optional: If the model will be used to score data, expose the model in the Machine Learning Model Management tool, which makes the model available to the Java Model Scoring stage.
 - a) Create a Spectrum Technology Platform data flow with steps [1](#) on page [5–2](#) on page [5](#) above, then replace step 3 with the Java Model Scoring stage. Set up this dataflow to run in batch mode to populate a file with model scores applied to refreshed data (the fields used as Xs or inputs are refreshed in step [1](#) on page [5–2](#) on page [5](#) as a natural part of doing business).
 - b) Alternatively, use a web service in Spectrum Technology Platform to score data on demand. For example, access the website, get the customer ID and model inputs, score those and return the score to a process that customizes web content for your customer.
5. Optional: You can also deploy model scores into a Context Graph graph database as an entity property, onto maps, or into CES applications.

2 - Machine Learning Stages

In this section

Enterprise Designer Stages.....	7
Flow Designer Stages.....	37



Enterprise Designer Stages

Binning

Introduction

The Binning stage performs what is known as unsupervised binning, which divides a continuous variable into groups (bins) without taking into account objective information. The data captured includes ranges, quantities, and percentage of values within each range.

Advantages to performing binning include the following:

- It allows records with missing data to be included in the model.
- It controls or mitigates the impact of outliers over the model.
- It solves the issue of having different scales among the characteristics, making the weights of the coefficients in the final model comparable.

In Spectrum Technology Platform unsupervised binning, you can use equal-width bins, where the data is divided into bins of equal size, or equal-frequency bins, where the data is divided into groups containing approximately the same number of records. In the Binning stage, equal-width bins are referred to as Equal Range bins and equal-frequency bins are referred to as Equal Population bins.

You can perform more binning functions using the Machine Learning Model Management **Binning Management** tool.

You can also view a list of binning and delete binning using command line instructions. See "Binning" in the "Administration Utility" section of the *Administration Guide*.

Defining Binning Properties

1. Under **Primary Stages > Deployed Stages > Machine Learning**, click the **Binning** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages.

Note: The input stage must be the data source that contains both the objective and input variable fields for your model. An output stage is not required unless you select the **Score input data** option on the **Basic Options** tab. You may also connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.

2. Double-click the Binning stage to show the **Binning Options** dialog box.

3. Enter a **Binning name** if you do not want to use the default name.
4. Check the **Overwrite** box to overwrite the existing model with new data.
5. Enter a **Description** of the model.
6. Click **Include** for each field whose data you want included in binning. Note that only numeric fields will appear in this list.
7. Click **OK** to save your settings.

Configuring Basic Options

1. Select whether you want to perform an equal-range or equal-population **Binning style**.
2. Select in **Null value bin** how you want to handle empty bin fields, which represent unknown values due to missing data.
 - Select **Highest** to assign null values to the highest bin.
 - Select **Lowest** to assign null values to the lowest bin.

The lowest bin is always bin 1.

3. Click **Target internal bins** and enter the number of bins you want to fill between the end bins. If you are performing equal-range binning, you may select this type of processing or **Bin width**, but not both. If you are performing equal-population binning, you may only perform internal-bin processing.
4. If you are performing equal-range binning and want to select this type of processing rather than internal-bin processing, click **Bin width** and enter the number of units you want in each bin.
5. Click **Include** for each field whose data you want included in binning.

Note: Only numeric fields will appear in this list.

6. Click **OK** to save your settings.

Binning Output

The Binning stage has two output ports. The first port will output all input fields plus a binned field for each selected input field. For example, if the input contains Name, Age, and Income fields and you perform binning on Age and Income, the output from the first port will contain the following fields:

- Name
- Age
- Binned_Age
- Income
- Binned_Income

The second port outputs four types of information for each selected input field. For example, if you perform binning on Age, the output from the second port will contain the following fields:

- Age_Bins
- Age_BinValue
- Age_Count
- Age_Percentage

K-Means Clustering

Introduction

K-Means Clustering creates models based on analytical clustering, which segments a set of records into clusters of similar records based on data values.

To create your model, you must first complete the **Model Properties** tab. The **Basic Options** and **Advanced Options** tabs provide sufficient default settings to complete a job, but you can alter those settings to meet your needs. You then run your job and a limited version of the resulting model output details appears on the **Model Output** tab. The model is stored on the Spectrum Technology Platform server and the complete output is available in the Machine Learning Model Management tool.

Defining Model Properties

1. Under **Primary Stages > Deployed Stages > Machine Learning**, click the **K-Means Clustering** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages.

Note: The input stage must be the data source that contains input variable fields for your model. An output stage is not required unless you select the **Score input data** option on the **Basic Options** tab. You may also connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.

2. Double-click the K-Means stage to show the **K-Means Clustering Options** dialog box.
3. Enter a **Model name** if you do not want to use the default name.
4. Optional: Check the **Overwrite** box to overwrite the existing model with new data.
5. Enter the **Number of clusters** you want in your model if you do not want the default number (5).
6. Optional: Enter a **Description** of the model.
7. Click **Include** for each field whose data you want added to the model.
8. Use the **Model Data Type** drop-down to specify whether the input field is to be used as a numeric, categorical, or datetime field.
9. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Basic Options

1. Leave **Standardize input fields** checked to standardize the numeric columns to have zero mean and unit variance.
If you do not use standardization, the results may include components dominated by variables appearing to have larger variances relative to other attributes as a matter of scale rather than true contribution.
2. Check **Estimate number of clusters** to have the K-Means algorithm attempt to determine the number of clusters that your model will contain. Even though you designate the number of desired clusters on the Model Properties tab, the routine may discover in its processing that a different number of clusters is more appropriate given the data.
3. Specify a value between 1 and 100 as the **Percentage for training data** when the input data is randomly split into training and test data samples.
4. Enter the value of 100 minus the amount you entered in step 3 on page 10 as the **Percentage for test data**.
5. Enter a number as the **Seed for sampling** to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
6. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Advanced Options

1. Leave **Ignore constant fields** checked to skip fields that have the same value for each record.
2. Leave **Seed for algorithm** checked and enter a seed number to ensure that when the data is split into test and training data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
3. Select the correct initialization mode in the **Init** dropdown.

Initialization mode	Description
Furthest	Initializes the first centroid randomly, but then initializes the second centroid to be the data point farthest away from it. Initializes the centroids to be well spread-out from each other.
Plus-Plus	Initializes the cluster centers before proceeding with the standard k -means optimization iterations. With the k -means++ initialization, the algorithm is guaranteed to find a solution that is $O(\log k)$ competitive to the optimal k -means solution.
Random	Chooses K clusters from the set of N observations at random so that each observation has an equal chance of being chosen. This is the default initialization mode.

4. Leave **Seed for N fold** checked and enter a seed number to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
5. Check **N fold** and enter the number of folds if you are performing cross-validation.
6. Check **Fold assignment** and select from the drop-down list if you are performing cross-validation.

Fold assignment	Description
Auto	Allows the algorithm to automatically choose an option; currently it uses Random. This is the default.
Modulo	Evenly splits the dataset into the folds and does not depend on the seed.

Note: This field is applicable only if you entered a value in **N fold**.

7. Check **Maximum iterations** and enter the number of training iterations that should take place.
8. Click **OK** to save the model and configuration or continue to the next tab.

Model Output

This tab shows the metrics you are using to assess the fitted model. These fields cannot be edited. The **Training** column will always contain data. If you selected a train/test split on the **Basic Options** tab, the **Test** column will also be filled, unless you have selected an N Fold validation on the **Advanced Options** tab, in which case the **N Fold** column will be filled. Click the **Output** button to regenerate the output, and click **Model details** to view the entire output in the Machine Learning Model Management tool.

Output Port


The K-Means Clustering stage contains one optional output port: the Model Metrics Port. This port's functionality is determined by your selections and input when completing the stage's basic and advanced options. For example, if you choose to conduct N Fold validation by checking the **N Fold** field on the **Advanced Options** tab, the **N Fold** column in the output metrics will be populated with data. Alternatively, if you choose not to conduct N Fold validation, the **N Fold** column will be blank.

Model Metrics Port

Perform this procedure to use the Model Metrics Port.

The **Model Metrics Port** lets you output the model assessment metrics to a data file. This will help you compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics.

1. Open a dataflow that uses the K-Means Clustering stage.

2. Attach a Write to File stage or another data output stage to the second output port.
3. Run the job.
4. Alternative to step 3 on page 12: Add an inspection point to the channel that connects the K-Means Clustering stage to the sink stage you added in step 2 on page 12 by right-clicking the channel and selecting "Add inspection point." Then click the Inspect Current Flow button  on the Enterprise Designer toolbar.

Inspection will run and you should see results similar to those shown below.

Creation Time	Flow Name	Metrics	Model Name	Model Type	N Fold	Test	Training
10/11/2018 2:37:00 AM	LinearRegressionTest1...	MSE	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10
10/11/2018 2:37:00 AM	LinearRegressionTest1...	RMSE	LinearRegressionTest1...	Linear Regression	0.0707247064967455	0.0567287296394643	1.14687396359675E-05
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Number of observations	LinearRegressionTest1...	Linear Regression	33	17	33
10/11/2018 2:37:00 AM	LinearRegressionTest1...	R2	LinearRegressionTest1...	Linear Regression	-0.366538763835926	0.0268719304981138	0.999999964065547
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Mean residual deviance	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10

Linear Regression

Introduction

Linear Regression enables you to perform machine learning by creating models from datasets that use continuous objectives with input variables.

To create your model, you must first complete the **Model Properties** tab. The **Basic Options** and **Advanced Options** tabs provide sufficient default settings to complete a job, but you can change those settings to meet your needs. You then run your job and a limited version of the resulting model appears on the **Model Output** tab; the complete output is available in the Machine Learning Model Management tool.

Defining Model Properties

1. Under **Primary Stages > Deployed Stages > Machine Learning**, click the **Linear Regression** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages. Note that the input stage must be the data source that contains both the objective and input variable fields for your model; an output stage is not required unless you select the Score input data option on the Basic Options tab. You may also connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.
2. Double-click the Linear Regression stage to show the **Linear Regression Options** dialog box.
3. Enter a **Model name** if you do not want to use the default name.
4. Check the **Overwrite** box to overwrite the existing model with new data.
5. Click the **Objective field** drop-down and select a numerical field.
6. Enter a **Description** of the model.
7. Click **Include** for each field whose data you want added to the model; be sure to include the field you selected as the Objective field.

8. Use the **Model Data Type** drop-down to specify whether each input field is to be used as a numeric, categorical, or datetime field.
9. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Basic Options

1. Leave **Standardize input fields** checked to standardize the numeric columns to have zero mean and unit variance.
If you do not use standardization, the results may include components dominated by variables appearing to have larger variances relative to other attributes as a matter of scale rather than true contribution.
2. Check **Score input data** to add a column for the model prediction (score) to the input data.
3. Select a **Link function** from the drop-down list. This specifies the link between random and systematic components. It says how the expected value of the response relates to the linear predictor of explanatory variables.

Link function	Description
Identity	Predicts nonsense "probabilities" less than zero or greater than one; sometimes used for binomial data to yield a linear probability model. $g(p) = p$
Inverse	Computes the inverse of link functions for real estimates. $g(\mu_i) = \mu_i$
Log	Counts occurrences in a fixed amount of time and space. $g(\mu_i) = \log(\mu_i)$

4. Specify how to handle missing data by checking **Skip** or **Impute means**, which will add the mean value for any missing data.
5. Specify a value between 1 and 100 as the **Percentage for training data** when the input data is randomly split into training and test data samples.
6. Enter the value of 100 minus the amount you entered in step 5 on page 13 as the **Percentage for test data**.
7. Enter a number as the **Seed for sampling** to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
8. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Advanced Options

1. Leave **Ignore constant fields** checked to skip fields that have the same value for each record.
2. Check **Compute p values** to calculate p values for the parameter estimates.
3. Check **Remove collinear column** to automatically remove collinear columns during model building.

This option must be checked if **Compute p values** is also checked.

This will result in a 0 coefficient in the returned model.

4. Leave **Include constant term (Intercept)** checked to include a constant term (intercept) in the model.

This field must be checked if **Remove collinear column** is also checked.

5. Select a **Solver** from the dropdown list.

Solver	Description
Auto	Solver will be determined based on input data and parameters.
CoordinateDescent	IRLSM with the covariance updates version of cyclical coordinate descent in the innermost loop.
CoordinateDescentNaive	IRLSM with the naive updates version of cyclical coordinate descent in the innermost loop.
IRLSM	Ideal for problems with a small number of predictors or for Lambda searches with L1 penalty.

Note: **CoordinateDescent** and **CoordinateDescentNaive** are currently experimental.

6. Leave **Seed for N fold** checked and enter a seed number to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck in this field to get a random split each time you run the flow.
7. Check **N fold** and enter the number of folds if you are performing cross-validation.
8. Click **Fold assignment** and select from the drop-down list if you are performing cross-validation. This field is applicable only if you entered a value in **N fold** and **Fold field** is not specified.

Option	Description
Auto	Allows the algorithm to automatically choose an option; currently it uses Random.

Option	Description
Modulo	Evenly splits the dataset into the folds and does not depend on the seed.
Random	Randomly splits the data into nfolds pieces; best for large datasets.

9. If you are performing cross-validation, check **Fold field** and select the field that contains the cross-validation fold index assignment from the drop-down list.

This field is applicable only if you did not enter a value in **N fold** and **Fold assignment**.

10. Check **Maximum iterations** and enter the number of training iterations that should take place.
11. Check **Objective epsilon** and enter the threshold for convergence; this must be a value between 0 and 1.

If the objective value is less than this threshold, the model will be converged.

12. Check **Beta epsilon** and enter the threshold for convergence; this must be a value between 0 and 1.

If the objective value is less than this threshold, the model will be converged. If the L1 normalization of the current beta change is below this threshold, consider using convergence.

13. Select the **Regularization type** you want to use.

Regularization type	Description
LASSO (Least Absolute Shrinkage and Selection Operator)	Selects a small subset of variables with a value of lambda high enough to be considered crucial. May not perform well when there are correlated predictor variables, as it will select one variable of the correlated group and remove all others. Also limited by high dimensionality; when a model contains more variables than records, LASSO is limited in how many variables it can select. Ridge Regression does not have this limitation. When the number of variables included in the model is large, or if the solution is known to be sparse, LASSO is recommended.
Ridge Regression	Retains all predictor variables and shrinks their coefficients proportionally. When correlated predictor variables exist, Ridge Regression reduces the coefficients of the entire group of correlated variables towards equaling one another. If you do not want correlated predictor variables removed from your model, use Ridge Regression.
Elastic Net	Combines LASSO and Ridge Regression by acting as a variable selector while also preserving the grouping effect for correlated variables (shrinking coefficients of correlated variables simultaneously). Elastic Net is not

Regularization type	Description
	limited by high dimensionality and can evaluate all variables when a model contains more variables than records.

A common concern in predictive modeling is overfitting, when an analytical model corresponds too closely (or exactly) to a specific dataset and therefore may fail when applied to additional data or future observations. Regularization is one method used to mitigate overfitting.

14. Check **Value of alpha** and change the value if you do not want to use the default of .5.

The alpha parameter controls the distribution between the ℓ_1 and ℓ_2 penalties. Valid values range between 0 and 1; a value of 1.0 represents LASSO, and a value of 0.0 produces ridge regression. The table below illustrates how alpha and lambda affect regularization.

lambda value	alpha value	Result
lambda == 0	alpha = any value	No regularization. alpha is ignored.
lambda > 0	alpha == 0	Ridge Regression
lambda > 0	alpha == 1	LASSO
lambda > 0	0 < alpha < 1	Elastic Net Penalty

Note: The single equals sign is an assignment operator meaning "is," while the double equals sign is an equality operator meaning "equal to."

15. Check **Value of lambda** and specify a value if you do not want Linear Regression to use the default method of calculating the lambda value, which is a heuristic based on training data.

The lambda parameter controls the amount of regularization applied. For instance, if lambda is 0.0, no regularization is applied and the alpha parameter is ignored.

16. Check **Search for optimal value of lambda** to have Linear Regression compute models for full regularization path.

This starts at lambda max (the highest lambda value that makes sense—that is, the lowest value driving all coefficients to zero) and goes down to lambda min on the log scale, decreasing regularization strength at each step. The returned model will have coefficients corresponding to the optimal lambda value as decided during training.

17. Check **Stop early** to end processing when there is no more relative improvement on the training or validation set.
18. Check **Maximum lambdas to search** and enter the maximum number of lambdas to use during the process of lambda search.
19. Check **Maximum active predictors** and enter the maximum number of predictors to use during computation.

This value is used as a stopping criterion to prevent expensive model building with many predictors.

20. Click **OK** to save the model and configuration or continue to the next tab.

Model Output

This tab shows the metrics you are using to assess the fitted model. These fields cannot be edited. The Training column will always contain data. If you selected a train/test split on the Basic Options tab, the Test column will also be filled, unless you have selected an N Fold validation on the Advanced Options tab, in which case the N Fold column will be filled.

After you run your job, the resulting model is stored on the Spectrum Technology Platform server. Click the **Output** button to regenerate the output and click **Model details** to view the entire output in the Machine Learning Model Management tool.

Output Ports

The Linear Regression stage contains two optional output ports: the Model Score Port and the Model Metrics Port. The functionality of these ports is determined by your selections and input when completing the stage's basic and advanced options. For example, if you choose to conduct N Fold validation by checking the **N Fold** field on the Advanced Options tab, the N Fold column in the output metrics generated by the Model Metrics Port will be populated with data. Alternatively, if you choose not to conduct N Fold validation, the N Fold column will be blank. Likewise, The Model Score Port is activated when you check the **Score input data** field on the Basic Options tab.


Model Score Port

When you check the **Score input data** field on the Basic Options tab, this tells Linear Regression to calculate predicted values when creating the model, which in turn adds the **Predicted_Value** column for that score in the output data. You can attach any kind of sink to this port: a Write to File stage, a Write to Null stage, and so on.

Model Metrics Port

Follow steps in this procedure to use the Model Metrics Port.

The **Model Metrics Port** lets you output the model assessment metrics to a data file. This will help you compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics.

1. Open a dataflow that uses the Linear Regression stage.
2. Attach a Write to File stage or another data output stage to the second output port.
3. Run the job.
4. Alternative to step 3 on page 17: Add an inspection point to the channel that connects the Linear Regression stage to the sink stage you added in step 2 by right-clicking the channel and selecting "Add inspection point." Then click the Inspect Current Flow button  on the Enterprise Designer toolbar. Inspection will run and you should see results similar to the ones shown below.

Creation Time	Flow Name	Metrics	Model Name	Model Type	N Fold	Test	Training
10/11/2018 2:37:00 AM	LinearRegressionTest1...	MSE	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10
10/11/2018 2:37:00 AM	LinearRegressionTest1...	RMSE	LinearRegressionTest1...	Linear Regression	0.0707247064967455	0.0567287296394643	1.14687396359675E-05
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Number of observations	LinearRegressionTest1...	Linear Regression	33	17	33
10/11/2018 2:37:00 AM	LinearRegressionTest1...	R2	LinearRegressionTest1...	Linear Regression	-0.366538763835926	0.0268719304981138	0.999999964065547
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Mean residual deviance	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10

Logistic Regression

Introduction

Logistic Regression enables you to perform machine learning by creating models from datasets that use binary objectives with input variables.

To create your model, you must first complete the **Model Properties** tab. The **Basic Options** and **Advanced Options** tabs provide sufficient default settings to complete a job, but you can change those settings to meet your needs. You then run your job and a limited version of the resulting model appears on the **Model Output** tab. The complete output is available in the Machine Learning Model Management tool.

Defining Model Properties

1. Under **Primary Stages > Deployed Stages > Machine Learning**, click the **Logistic Regression** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages.

Note: The input stage must be the data source that contains both the objective and input variable fields for your model; an output stage is not required unless you select the **Score input data** option on the **Basic Options** tab. You may also connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.

2. Double-click the Logistic Regression stage to show the **Logistic Regression Options** dialog box.
3. Enter a **Model name** if you do not want to use the default name.
4. Check the **Overwrite** box to overwrite the existing model with new data.
5. Click the **Objective field** drop-down and select "Categorical."
6. Enter a **Description** of the model.
7. Click **Include** for each field whose data you want added to the model.
Be sure to include the field you selected as the Objective field.
8. Use the **Model Data Type** drop-down to specify whether each input field is to be used as a numeric, categorical, or datetime field.
9. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Basic Options

1. Leave **Standardize input fields** checked to standardize the numeric columns to have zero mean and unit variance.

If you do not use standardization, the results may include components dominated by variables appearing to have larger variances relative to other attributes as a matter of scale rather than true contribution.

2. Check **Score input data** to add a column for the model prediction (score) to the input data.
3. Check **Prior** if the data has been sampled and the mean of response does not reflect reality; then enter the prior probability for $p(y=1)$ in the text field.
4. Specify how to handle missing data by checking **Skip** or **Impute means**, which will add the mean value for any missing data.
5. Specify a value between 1 and 100 as the **Percentage for training data** when the input data is randomly split into training and test data samples.
6. Enter the value of 100 minus the amount you entered in Step 5 as the **Percentage for test data**.
7. Enter a number as the **Seed for sampling** to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
8. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Advanced Options

1. Leave **Ignore constant fields** checked to skip fields that have the same value for each record.
2. Leave **Compute p values** checked to calculate p values for the parameter estimates.
3. Leave **Remove collinear column** checked to automatically remove collinear columns during model building.

This option must be checked if **Compute p values** is also checked.

This will result in a 0 coefficient in the returned model.

4. Leave **Include constant term (Intercept)** checked to include a constant term (intercept) in the model.
This field must be checked if **Remove collinear column** is also checked.
5. Select a **Solver** from the dropdown list.

Solver	Description
Auto	Solver will be determined based on input data and parameters.
CoordinateDescent	IRLSM with the covariance updates version of cyclical coordinate descent in the innermost loop.
CoordinateDescentNaive	IRLSM with the naive updates version of cyclical coordinate descent in the innermost loop.
IRLSM	Ideal for problems with a small number of predictors or for Lambda searches with L1 penalty.

Solver	Description
L_BFGS	Ideal for datasets with many columns.

Note: **CoordinateDescentNaive** and **CoordinateDescentNaive** are currently experimental.

- Leave **Seed for N fold** checked and enter a seed number to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
- Check **N fold** and enter the number of folds if you are performing cross-validation.
- Check **Fold assignment** and select from the drop-down list if you are performing cross-validation.

Fold assignment	Description
Auto	Allows the algorithm to automatically choose an option; currently it uses Random.
Modulo	Evenly splits the dataset into the folds and does not depend on the seed.
Random	Randomly splits the data into nfolds pieces; best for large datasets.
Stratified	Stratifies the folds based on the response variable for classification problems. Evenly distributes observations from the different classes to all sets when splitting a dataset into train and test data. This can be useful if there are many classes and the dataset is relatively small.

This field is applicable only if you entered a value in **N fold** and **Fold field** is not specified.

- If you are performing cross-validation, check **Fold field** and select the field that contains the cross-validation fold index assignment from the drop-down list.

This field is applicable only if you did not enter a value in **N fold** and **Fold assignment**.

- Check **Maximum iterations** and enter the number of training iterations that should take place.
- Check **Objective epsilon** and enter the threshold for convergence; this must be a value between 0 and 1.
If the objective value is less than this threshold, the model will be converged.
- Check **Beta epsilon** and enter the threshold for convergence; this must be a value between 0 and 1.
If the objective value is less than this threshold, the model will be converged. If the L1 normalization of the current beta change is below this threshold, consider using convergence.
- Select the **Regularization type** you want to use.

Regularization type	Description
LASSO (Least Absolute Shrinkage and Selection Operator)	Selects a small subset of variables with a value of lambda high enough to be considered crucial. May not perform well when there are correlated predictor variables, as it will select one variable of the correlated group and remove all others. Also limited by high dimensionality; when a model contains more variables than records, LASSO is limited in how many variables it can select. Ridge Regression does not have this limitation. When the number of variables included in the model is large, or if the solution is known to be sparse, LASSO is recommended.
Ridge Regression	Retains all predictor variables and shrinks their coefficients proportionally. When correlated predictor variables exist, Ridge Regression reduces the coefficients of the entire group of correlated variables towards equaling one another. If you do not want correlated predictor variables removed from your model, use Ridge Regression.
Elastic Net	Combines LASSO and Ridge Regression by acting as a variable selector while also preserving the grouping effect for correlated variables (shrinking coefficients of correlated variables simultaneously). Elastic Net is not limited by high dimensionality and can evaluate all variables when a model contains more variables than records.

A common concern in predictive modeling is overfitting, when an analytical model corresponds too closely (or exactly) to a specific dataset and therefore may fail when applied to additional data or future observations. Regularization is one method used to mitigate overfitting.

14. Check **Value of alpha** and change the value if you do not want to use the default of .5.

The alpha parameter controls the distribution between the ℓ_1 and ℓ_2 penalties. Valid values range between 0 and 1; a value of 1.0 represents LASSO, and a value of 0.0 produces ridge regression. The table below illustrates how alpha and lambda affect regularization.

lambda value	alpha value	Result
lambda == 0	alpha = any value	No regularization. alpha is ignored.
lambda > 0	alpha == 0	Ridge Regression
lambda > 0	alpha == 1	LASSO
lambda > 0	0 < alpha < 1	Elastic Net Penalty

Note: The single equals sign is an assignment operator meaning "is," while the double equals sign is an equality operator meaning "equal to."

15. Check **Value of lambda** and specify a value if you do not want Logistic Regression to use the default method of calculating the lambda value, which is a heuristic based on training data.

The lambda parameter controls the amount of regularization applied. For example, if lambda is 0.0, no regularization is applied and the alpha parameter is ignored.

16. Check **Search for optimal value of lambda** to have Logistic Regression compute models for full regularization path.
This starts at lambda max (the highest lambda value that makes sense—that is, the lowest value driving all coefficients to zero) and goes down to lambda min on the log scale, decreasing regularization strength at each step.
The returned model will have coefficients corresponding to the optimal lambda value as decided during training.
17. Check **Stop early** to end processing when there is no more relative improvement on the training or validation set.
18. Check **Maximum lambdas to search** and enter the maximum number of lambdas to use during the process of lambda search.
19. Check **Maximum active predictors** and enter the maximum number of predictors to use during computation.
This value is used as a stopping criterion to prevent expensive model building with many predictors.
20. Click **OK** to save the model and configuration or continue to the next tab.

Model Output

This tab shows the metrics you are using to assess the fitted model. These fields cannot be edited. The Training column will always contain data. If you selected a train/test split on the Basic Options tab, the Test column will also be filled, unless you have selected an N Fold validation on the Advanced Options tab, in which case the N Fold column will be filled.

After you run your job, the resulting model is stored on the Spectrum Technology Platform server. Click the **Output** button to regenerate the output and click **Model details** to view the entire output in the Machine Learning Model Management tool.

Output Ports

The Logistic Regression stage contains two optional output ports: the Model Score Port and the Model Metrics Port. The functionality of these ports is determined by your selections and input when completing the stage's basic and advanced options. For example, if you choose to conduct N Fold validation by checking the **N Fold** field on the **Advanced Options** tab, the N Fold column in the output metrics generated by the Model Metrics Port will be populated with data. Alternatively, if you choose not to conduct N Fold validation, the N Fold column will be blank. Likewise, The Model Score Port is activated when you check the **Score input data** field on the **Basic Options** tab.

Model Score Port


When you check the **Score input data** field on the **Basic Options** tab, this tells Logistic Regression to calculate predicted values when creating the model, which in turn adds the **Predicted_Value**,

Probability_of_class_A, and **Probability_of_class_B** columns for that score in the output data. You can attach any kind of sink to this port: a Write to File stage, a Write to Null stage, and so on.

Model Metrics Port

Perform this procedure to use the Model Metrics Port.

The **Model Metrics Port** lets you output the model assessment metrics to a data file. This will help you compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics.

1. Open a dataflow that uses the Logistic Regression stage.
2. Attach a Write to File stage or another data output stage to the second output port.
3. Run the job.
4. Alternative to step 3 on page 23: Add an inspection point to the channel that connects the Logistic Regression stage to the sink stage you added in step 2 on page 23 by right-clicking the channel and selecting "Add inspection point." Then click the Inspect Current Flow button  on the Enterprise Designer toolbar. Inspection will run and you should see results similar to the ones shown below.

Creation Time	Flow Name	Metrics	Model Name	Model Type	N Fold	Test	Training
10/11/2018 2:37:00 AM	LinearRegressionTest1...	MSE	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10
10/11/2018 2:37:00 AM	LinearRegressionTest1...	RMSE	LinearRegressionTest1...	Linear Regression	0.0707247064967455	0.0567287296394643	1.14687396359675E-05
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Number of observations	LinearRegressionTest1...	Linear Regression	33	17	33
10/11/2018 2:37:00 AM	LinearRegressionTest1...	R2	LinearRegressionTest1...	Linear Regression	-0.366538763835926	0.0268719304981138	0.999999964065547
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Mean residual deviance	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10

Principal Component Analysis

Introduction

Principal Component Analysis (PCA) is a statistical process that converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables known as principal components.

To create your model, you must first complete the **Model Properties** tab. The **Basic Options** and **Advanced Options** tabs provide sufficient default settings to complete a job, but you can change those settings to meet your needs. You then run your job and a limited version of the resulting model appears on the **Model Output** tab; the complete output is available in the Machine Learning Model Management tool. If you are satisfied with the output of your model, you can then expose it and use it in a scoring dataflow.

Defining Model Properties

1. Under **Primary Stages > Deployed Stages > Machine Learning**, click the **PCA Options** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages.

Note: The input stage must be the data source that contains the principal components for your model. An output stage is not required but you may connect one if you wish to capture your output independent of the Machine Learning Model Management tool.

2. Double-click the PCA Options stage to show the **PCA Options** dialog box.
3. Enter a **Model name** if you do not want to use the default name.
4. Optional: Check the **Overwrite** box to overwrite the existing model with new data.
5. Enter the number of **Principal components** you want your model to contain.
6. Optional: Enter a **Description** of the model.
7. In the **Inputs** table click "Include" for each field whose data you want added to the model.
8. Use the **Model Data Type** drop-down to specify whether the input field is to be used as a categorical, datetime, numeric, string, or uniqueid field.
9. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Basic Options

1. Configure **Use all factor level**.
 - Leave this option unchecked to skip the first principal component, which has the largest variance in the data.
 - Check this box to retain the first principal component.
2. Select the appropriate **Transform** for the training data.

Transform	Description
Demmean	Subtracts the mean of each column.
Descale	Divides by the standard deviation of each column.
None	No transform.
Normalize	Demeans and divides each column by its range (maximum minus minimum).
Standardize	Uses zero mean and unit variance. This is the default transform.

3. Specify how to handle **Missing data** by checking **Skip** or **Impute means**, which will add the mean value for any missing data.
4. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Advanced Options

1. Leave **Ignore constant fields** checked to skip fields that have the same value for each record.
2. Select a **PCA method** from the dropdown list. Note that GLRM and Power are currently experimental.

PCA method	Description
GLRM	Fits a generalized low-rank model with L2 loss function and no regularization; solves for the SVD using local matrix algebra. This option is enabled only if you checked Use all factor level on the Basic Options tab.
GramSVD	Uses a distributed computation of the Gram matrix, followed by a local SVD using the JAMA package.
Power	Computes the SVD using the power iteration method.
Randomized	Uses the randomized subspace iteration method.

3. Leave **Maximum iterations** unchecked to have an unlimited number of training iterations (default). Check the box and enter a number to limit the amount of training iterations.
4. Click **OK** to save the model and configuration or continue to the next tab.

Model Output

This tab shows the metrics you are using to assess the fitted model. These fields cannot be edited.

After you run your job, the resulting model is stored on the Spectrum Technology Platform server. Click the **Output** button to regenerate the output and click **Model details** to view the entire output in the Machine Learning Model Management tool.


Output Port

The Principal Component Analysis stage contains one optional output port: the Model Metrics Port. This port's functionality is determined by your selections and input when completing the stage's basic and advanced options.

Model Metrics Port

Perform this procedure to use the Model Metrics Port.

The **Model Metrics Port** lets you output the model assessment metrics to a data file. This will help you compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics.

1. Open a dataflow that uses the PCA stage.
2. Attach a Write to File stage or another data output stage to the second output port.
3. Run the job.
4. Alternative to step 3 on page 26: Add an inspection point to the channel that connects the PCA stage to the sink stage you added in step 2 on page 26 by right-clicking the channel and selecting "Add inspection point." Then click the Inspect Current Flow button  on the Enterprise Designer toolbar. Inspection will run and you should see results similar to the ones shown below.

Creation Time	Cumulative Proportion	Flow Name	Model Name	Model Type	Principal Component	Proportion of Variance	Standard Deviation
10/11/2018 8:36:00 PM	0.120990707073471	PCA_MODEL_ASSESSMENT	PCA Model Assessment	PCA	PC1	0.120990707073471	1.73278529942543
10/11/2018 8:36:00 PM	0.163608702477588	PCA_MODEL_ASSESSMENT	PCA Model Assessment	PCA	PC2	0.0426179954041175	1.02840755055022
10/11/2018 8:36:00 PM	0.20114715020656	PCA_MODEL_ASSESSMENT	PCA Model Assessment	PCA	PC3	0.0375384477289716	0.965176862701583
10/11/2018 8:36:00 PM	0.236650281720107	PCA_MODEL_ASSESSMENT	PCA Model Assessment	PCA	PC4	0.0355031315135469	0.93864652798561
10/11/2018 8:36:00 PM	0.269754397170741	PCA_MODEL_ASSESSMENT	PCA Model Assessment	PCA	PC5	0.0331041154506347	0.90637880520786

Random Forest Classification

Introduction

Random Forest Classification enables you to perform machine learning by creating models from datasets that use continuous objectives with input variables.

To create your model, you must first complete the Model Properties tab. The Basic Options and Advanced Options tabs provide sufficient default settings to complete a job, but you can change those settings to meet your needs. You then run your job and a limited version of the resulting model appears on the Model Output tab; the complete output is available in the Machine Learning Model Management tool.

Note: For additional information, refer to this article about [Distributed Random Forest \(DRF\)](#) for additional information regarding Random Forest Classification and its options.

Defining Model Properties

1. Under **Primary Stages > Deployed Stages > Machine Learning**, click the **Random Forest Classification** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages.

Note: The input stage must be the data source that contains both the objective and input variable fields for your model; an output stage is not required unless you select the Score input data option on the Basic Options tab. You may also connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.

2. Double-click the Random Forest Classification stage to show the **Random Forest Classification Options** dialog box.

3. Enter a **Model name** if you do not want to use the default name.
4. Optional: Check the **Overwrite** box to overwrite the existing model with new data.
5. Click the **Objective field** drop-down and select a numeric field.
6. Click **Multinomial levels** and enter the maximum number of categories into which the objective field can be grouped. Note that checking this option will disable the **Score input data** option on the Basic Options tab.
7. Optional: Enter a **Description** of the model.
8. Click **Include** for each field whose data you want added to the model.
Be sure to include the field you selected as the Objective field.
9. Use the **Model Data Type** drop-down to specify whether each input field is to be used as a numeric, categorical, or datetime field.
10. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Basic Options

1. Enter the maximum **Number of trees** in your model.
2. Enter the **Maximum depth**.
This is the maximum number of levels you want your model to contain.
3. Enter the **Minimum rows**.
This is the minimum number of rows (or records) you want your model to contain.
4. Enter the **Number of bins numeric**.
This is the number of bins you want the histogram to build and then split at the best point.
5. Enter the **Number of bins top level**.
This is the minimum number of bins you want at the root level.
6. Enter the **Number of bins categorical**.
This is the maximum number of bins you want the histogram to build and then split at the best point.
7. Check **Sample rate** and enter the percentage of the rows to be used as a sample in each tree.
This can be a value from 0.0 to 1.0.
8. Check **Column sample rate per tree** and enter the column sampling rate for each tree.
This can be a value from 0.0 to 1.0.
9. Check **Columns at each level** and enter the relative change of the column sampling rate for every level.
Valid values range from 1.0 to the number of the selected input predictor. Default is 1.0.
10. Check **Score input data** to add a column for the model prediction (score) to the input data.

Note: This option is disabled if you checked **Multinomial levels** on the Model Properties tab.

11. Specify a value between 1 and 100 as the **Percentage for training data** when the input data is randomly split into training and test data samples.
12. Enter the value of 100 minus the amount you entered in step 11 on page 28 as the **Percentage for test data**.
13. **Seed for sampling** to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
14. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Advanced Options

1. Leave **Ignore constant fields** checked to skip fields that have the same value for each record.
2. Check **Balance classes** to balance the class distribution and either under sample the majority classes or over sample the minority classes.
3. Select a **Histogram type**.

Histogram type	Description
Auto	Buckets are binned from minimum to maximum in steps of $(\text{max}-\text{min})/N$. Use this option to specify the type of histogram for finding optimal split points.
QuantilesGlobal	Buckets have equal population. This computes <code>nbins</code> quantiles for each numeric (non-binary) column, then refines/pads each bucket (between two quantiles) uniformly (and randomly for remainders) into a total of <code>nbins_top_level</code> bins.
Random	The algorithm will sample $N-1$ points from minimum to maximum and use the sorted list of those to find the best split.
RoundRobin	The algorithm will cycle through all histogram types (one per tree).
UniformAdaptive	Each feature is binned into buckets of equal step size (not population). This is the quickest method but can lead to less accurate splits if the distribution is highly skewed.

4. Select a **Categorical encoding**.

Categorical encoding	Description
Auto	Automatically performs <code>enum</code> encoding.

Categorical encoding	Description
Binary	Converts categories to integers, then to binary, and assigns each digit a separate column. Encodes the data in fewer dimensions but with some distortion of the distances. Note: No more than 32 columns can exist per categorical feature.
Eigen	k columns per categorical feature, keeping projections of one-hot-encoded matrix onto k -dim eigen space only.
Enum	Cycles through all histogram types (one per tree).
OneHotExplicit	One column exists per category, with "1" or "0" in each cell representing whether the row contains that column's category.

5. Leave **Seed for algorithm** and **N fold** checked and enter a seed number to ensure that when the data is split into test and training data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
6. If you are performing cross-validation, check **N fold** and enter the number of folds.
7. If you are performing cross-validation, check **Fold assignment** and select from the dropdown list.

Fold assignment	Description
Auto	Allows the algorithm to automatically choose an option; currently it uses Random.
Modulo	Evenly splits the dataset into the folds and does not depend on the seed.
Random	Randomly splits the data into nfolds pieces; best for large datasets.
Stratified	Stratifies the folds based on the response variable for classification problems. Evenly distributes observations from the different classes to all sets when splitting a dataset into train and test data. This can be useful if there are many classes and the dataset is relatively small.

This field is applicable only if you entered a value in **N fold** and **Fold field** is not specified.

8. If you are performing cross-validation, check **Fold field** and select the field that contains the cross-validation fold index assignment from the drop-down list.

This field is applicable only if you did not enter a value in **N fold** and **Fold assignment**.

9. Check **Stopping rounds** to end training when the Stopping_metric option does not improve for the specified number of training rounds and enter the number of unsuccessful training rounds to occur before stopping. To disable this feature, specify 0.

The metric is computed on the validation data (if provided); otherwise, training data is used.

10. Select a **Stopping metric** to determine when to quit creating new trees.

Stopping metric	Description
AUC	Area under ROC curve. Note: Applicable only to binomial models.
Auto	Defaults to deviance.
Lifftopgroup	Top 1%.
Logloss	Logarithmic loss.
Meanperclasserror	The average misclassification rate.
Misclassification	The value of $(1 - (\text{correct predictions}/\text{total predictions})) * 100$.
MSE	Mean squared error; incorporates both the variance and the bias of the predictor.
RMSE	Root mean square error; measures the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. Also the square root of MSE.

11. Check **Stopping tolerance** and enter a value to specify the relative tolerance for the metric-based stopping to end training if the improvement is less than this value.

This field is enabled only if you checked **Stopping rounds**.

12. Check **Minimum split improvement** and enter a value to specify the minimum relative improvement in squared error reduction in order for a split to happen.

When properly executed, this option can help reduce overfitting. Optimal values would be in the 1e-10...1e-3 range. This field is enabled only if you checked **Stopping rounds**.

13. Click **OK** to save the model and configuration or continue to the next tab.

Model Output

This tab shows the metrics you are using to assess the fitted model. These fields cannot be edited. The Training column will always contain data. If you selected a training/test split on the **Basic Options** tab, the **Test** column will also be filled, unless you have selected an N Fold validation on the **Advanced Options** tab, in which case the **N Fold** column will be filled.

After you run your job, the resulting model is stored on the Spectrum Technology Platform server. Click the **Output** button to regenerate the output and click **Model details** to view the entire output in the Machine Learning Model Management tool.

Output Ports

The Random Forest Classification stage contains two optional output ports: the Model Score Port and the Model Metrics Port. The functionality of these ports is determined by your selections and input when completing the stage's basic and advanced options. For example, if you choose to conduct N Fold validation by checking the **N Fold** field on the **Advanced Options** tab, the N Fold column in the output metrics generated by the Model Metrics Port will be populated with data. Alternatively, if you choose not to conduct N Fold validation, the N Fold column will be blank. Likewise, The Model Score Port is activated when you check the **Score input data** field on the **Basic Options** tab.

Model Score Port

When you check the **Score input data** field on the **Basic Options** tab, this tells Random Forest Classification to calculate predicted values when creating the model, which in turn adds the **Predicted_Value**, **Probability_of_class_A**, and **Probability_of_class_B** columns for that score in the output data. You can attach any kind of sink to this port: a Write to File stage, a Write to Null stage, and so on.


Note: This port is not functional for Random Forest Classification multinomial models.

Model Metrics Port

Perform this procedure to use the Model Metrics Port.

The **Model Metrics Port** lets you output the model assessment metrics to a data file. This will help you compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics.

1. Open a dataflow that uses the Random Forest Classification stage.
2. Attach a Write to File stage or another data output stage to the second output port.
3. Run the job.

- Alternative to step 3 on page 31: Add an inspection point to the channel that connects the Random Forest Classification stage to the sink stage you added in step 2 on page 31 by right-clicking the channel and selecting "Add inspection point." Then click the Inspect Current Flow button  on the Enterprise Designer toolbar. Inspection will run and you should see results similar to the ones shown below.

Creation Time	Flow Name	Metrics	Model Name	Model Type	N Fold	Test	Training
10/11/2018 2:37:00 AM	LinearRegressionTest1...	MSE	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10
10/11/2018 2:37:00 AM	LinearRegressionTest1...	RMSE	LinearRegressionTest1...	Linear Regression	0.0707247064967455	0.0567287296394643	1.14687396359675E-05
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Number of observations	LinearRegressionTest1...	Linear Regression	33	17	33
10/11/2018 2:37:00 AM	LinearRegressionTest1...	R2	LinearRegressionTest1...	Linear Regression	-0.366538763835926	0.0268719304981138	0.999999964065547
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Mean residual deviance	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10

Random Forest Regression

Introduction

Random Forest Regression enables you to perform machine learning by creating models from datasets that use binary objectives with input variables.

To create your model, you must first complete the **Model Properties** tab. The **Basic Options** and **Advanced Options** tabs provide sufficient default settings to complete a job, but you can change those settings to meet your needs. You then run your job and a limited version of the resulting model appears on the **Model Output** tab; the complete output is available in the Machine Learning Model Management tool.

Note: For more information regarding Random Forest Regression and its options, see [Distributed Random Forest \(DRF\)](#).

Defining Model Properties

- Under **Primary Stages / Deployed Stages / Machine Learning**, click the **Random Forest Regression** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages.

Note: The input stage must be the data source that contains both the objective and input variable fields for your model; an output stage is not required unless you select the Score input data option on the Basic Options tab. You may also connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.

- Double-click the Random Forest Regression stage to show the **Random Forest Regression Options** dialog box.
- Enter a **Model name** if you do not want to use the default name.
- Optional: Check the **Overwrite** box to overwrite the existing model with new data.

5. Click the **Objective field** drop-down and select a numeric field.
6. Optional: Enter a **Description** of the model.
7. Click **Include** for each field whose data you want added to the model; be sure to include the field you selected as the Objective field.
8. Use the **Model Data Type** drop-down to specify whether each input field is to be used as a numeric, categorical, or datetime field.
9. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Basic Options

1. Enter the maximum **Number of trees** in your model. Default is 50.
2. Enter the **Maximum depth**.
This is the maximum number of levels you want your model to contain. The default is 5.
3. Enter the **Minimum rows**.
This is the minimum number of rows (or records) you want your model to contain. The default is 10.
4. Enter the **Number of bins numeric**.
This is the number of bins you want the histogram to build and then split at the best point. The default is 20.
5. Enter the **Number of bins top level**.
This is the minimum number of bins you want at the root level. The default is 1024.
6. Enter the **Number of bins categorical**.
This is the maximum number of bins you want the histogram to build and then split at the best point. The default is 1024.
7. Check **Sample rate** and enter the percentage of the rows to be used as a sample in each tree. This can be a value from 0.0 to 1.0.
8. Check **Column sample rate per tree** and enter the column sampling rate for each tree. This can be a value from 0.0 to 1.0.
9. Check **Columns at each level** and enter the relative change of the column sampling rate for every level.
This option defaults to 1.0 and can be a value from 0.0 to 2.0.
10. Check **Score input data** to add a column for the model prediction (score) to the input data.
11. Specify a value between 1 and 100 as the **Percentage for training data** when the input data is randomly split into training and test data samples.
12. Enter the value of 100 minus the amount you entered in step 11 on page 33 as the **Percentage for test data**.
13. **Seed for sampling** to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.

- Click **OK** to save the model and configuration or continue to the next tab.

Configuring Advanced Options

- Leave **Ignore constant fields** checked to skip fields that have the same value for each record.
- Select a **Histogram type**.

Histogram type	Description
Auto	Buckets are binned from minimum to maximum in steps of $(\text{max-min})/N$. Use this option to specify the type of histogram for finding optimal split points.
QuantilesGlobal	Buckets have equal population. This computes <code>nbins</code> quantiles for each numeric (non-binary) column, then refines/pads each bucket (between two quantiles) uniformly (and randomly for remainders) into a total of <code>nbins_top_level</code> bins.
Random	The algorithm will sample $N-1$ points from minimum to maximum and use the sorted list of those to find the best split.
RoundRobin	The algorithm will cycle through all histogram types (one per tree).
UniformAdaptive	Each feature is binned into buckets of equal step size (not population). This is the quickest method but can lead to less accurate splits if the distribution is highly skewed.

- Select a **Categorical encoding**.

Categorical encoding	Description
Auto	Automatically performs <code>enum</code> encoding.
Binary	<p>Converts categories to integers, then to binary, and assigns each digit a separate column. Encodes the data in fewer dimensions but with some distortion of the distances.</p> <p>Note: No more than 32 columns can exist per categorical feature.</p>

Categorical encoding	Description
Eigen	k columns per categorical feature, keeping projections of one-hot-encoded matrix onto k -dim eigen space only.
Enum	Cycles through all histogram types (one per tree).
OneHotExplicit	One column exists per category, with "1" or "0" in each cell representing whether the row contains that column's category.

4. Leave **Seed for algorithm and N fold** checked and enter a seed number to ensure that when the data is split into test and training data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
5. Check **N fold** and enter the number of folds if you are performing cross-validation.
6. If you are performing cross-validation, check **Fold assignment** and select from the dropdown list .

Fold assignment	Description
Auto	Allows the algorithm to automatically choose an option; currently it uses Random.
Modulo	Evenly splits the dataset into the folds and does not depend on the seed.
Random	Randomly splits the data into nfolds pieces; best for large datasets.

This field is applicable only if you entered a value in **N fold** and **Fold field** is not specified.

7. If you are performing cross-validation, check **Fold field** and select the field that contains the cross-validation fold index assignment from the drop-down list.

This field is applicable only if you did not enter a value in **N fold** and **Fold assignment**.

8. Check **Stopping rounds** to end training when the Stopping_metric option does not improve for the specified number of training rounds and enter the number of unsuccessful training rounds to occur before stopping. To disable this feature, specify 0.

The metric is computed on the validation data (if provided); otherwise, training data is used.

9. Select a **Stopping metric** to determine when to quit creating new trees.

Stopping metric	Description
Auto	Defaults to <code>deviance</code> .

Stopping metric	Description
deviance	Mean residual deviance; identical to MSE.
MAE	Mean absolute error; the difference between two continuous variables.
MSE	Mean squared error; incorporates both the variance and the bias of the predictor.
RMSE	Root mean square error; measures the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. Also the square root of MSE.
RMSLE	Root mean squared logarithmic error; measures the ratio between predicted and actual.

10. Check **Stopping tolerance** and enter a value to specify the relative tolerance for the metric-based stopping to end training if the improvement is less than this value.
11. Check **Minimum split improvement** and enter a value to specify the minimum relative improvement in squared error reduction in order for a split to happen.
When properly executed, this option can help reduce overfitting. Optimal values would be in the 1e-10...1e-3 range. This field is enabled only if you checked **Stopping rounds**.
12. Click **OK** to save the model and configuration or continue to the next tab.

Model Output

This tab shows the metrics you are using to assess the fitted model. These fields cannot be edited. The Training column will always contain data. If you selected a training/test split on the Basic Options tab, the Test column will also be filled, unless you have selected an N Fold validation on the Advanced Options tab, in which case the N Fold column will be filled.

After you run your job, the resulting model is stored on the Spectrum Technology Platform server. Click the **Output** button to regenerate the output and click **Model details** to view the entire output in the Machine Learning Model Management tool.

Output Ports

The Random Forest Regression stage contains two optional output ports: the Model Score Port and the Model Metrics Port. The functionality of these ports is determined by your selections and input when completing the stage's basic and advanced options. For example, if you choose to conduct N Fold validation by checking the **N Fold** field on the **Advanced Options** tab, the N Fold column in the output metrics generated by the Model Metrics Port will be populated with data. Alternatively, if

you choose not to conduct N Fold validation, the N Fold column will be blank. Likewise, The Model Score Port is activated when you check the **Score input data** field on the **Basic Options** tab.


Model Score Port

When you check the **Score input data** field on the **Basic Options** tab, this tells Random Forest Regression to calculate predicted values when creating the model, which in turn adds the **Predicted_Value** column for that score in the output data. You can attach any kind of sink to this port: a Write to File stage, a Write to Null stage, and so on.

Model Metrics Port

Perform this procedure to use the Model Metrics Port.

The **Model Metrics Port** lets you output the model assessment metrics to a data file. This will help you compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics.

1. Open a dataflow that uses the Random Forest Regression stage.
2. Attach a Write to File stage or another data output stage to the second output port.
3. Run the job.
4. Alternative to step 3 on page 37: Add an inspection point to the channel that connects the Random Forest Regression stage to the sink stage you added in step 2 on page 37 by right-clicking the channel and selecting "Add inspection point." Then click the Inspect Current Flow button  on the Enterprise Designer toolbar. Inspection will run and you should see results similar to the ones shown below.

Creation Time	Flow Name	Metrics	Model Name	Model Type	N Fold	Test	Training
10/11/2018 2:37:00 AM	LinearRegressionTest1...	MSE	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10
10/11/2018 2:37:00 AM	LinearRegressionTest1...	RMSE	LinearRegressionTest1...	Linear Regression	0.0707247064967455	0.0567287296394643	1.14687396359675E-05
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Number of observations	LinearRegressionTest1...	Linear Regression	33	17	33
10/11/2018 2:37:00 AM	LinearRegressionTest1...	R2	LinearRegressionTest1...	Linear Regression	-0.366538763836926	0.0268719304981138	0.999999964065547
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Mean residual deviance	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10

Flow Designer Stages

Binning

The Binning stage performs what is known as unsupervised binning, which divides a continuous variable into groups (bins) without taking into account objective information. The data captured includes ranges, quantities, and percentage of values within each range.

Advantages to performing binning include the following:

- It allows records with missing data to be included in the model.
- It controls or mitigates the impact of outliers over the model.
- It solves the issue of having different scales among the characteristics, making the weights of the coefficients in the final model comparable.

In Spectrum Technology Platform unsupervised binning, you can use equal-width bins, where the data is divided into bins of equal size, or equal-frequency bins, where the data is divided into groups containing approximately the same number of records. In the Binning stage, equal-width bins are referred to as Equal Range bins and equal-frequency bins are referred to as Equal Population bins.

You can perform more binning functions using the Machine Learning Model Management **Binning Management** tool.

You can also view a list of binning and delete binning using command line instructions. See "Binning" in the "Administration Utility" section of the *Administration Guide*.

How to

Add Binning to workflow

1. In the **Stages** panel, scroll to **Machine Learning**, and drag the **Binning** stage onto the canvas.
2. Connect the stage to other stages.

For more information, see **Binning Stage ports**.

3. Double-click the Binning stage to open the **Binning Properties**.
4. On the **Binning Properties** tab, configure the model name and the input fields to be included in the binning.

For more information about options on this tab, see **Binning Properties tab** on page 39.

5. On the **Basic Options** tab, configure the binning style, null value bin, number of target bins, and bin width.

For more information about options on this tab, see **Basic Options tab** on page 39.

6. Click **Apply** to save your changes.

Reference

Stage ports

Input

The input stage must be the data source that contains input variable fields for your model.

Output

The Binning stage has two output ports. The first port will output all input fields plus a binned field for each selected input field. For example, if the input contains Name, Age, and Income fields and you perform binning on Age and Income, the output from the first port will contain the following fields:

- Name
- Age
- Binned_Age
- Income
- Binned_Income

The second port outputs four types of information for each selected input field. For example, if you perform binning on Age, the output from the second port will contain the following fields:

- Age_Bins
- Age_BinValue
- Age_Count
- Age_Percentage

An output stage is not required. You may connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.

Binning Properties tab

Model Name	Specifies the name of a binning model.
Overwrite	Check this check box to overwrite data in an existing model.
Description	Provides space to document the purpose of a model.
Inputs	This table shows numeric input fields along with the data type. Check the Include check box to include data from a field in binning. Note: Only numeric fields appear in this list.

Basic Options tab

Binning Style	Select whether to perform an Equal-Range or Equal Population binning.
Null value bin	Specifies how to handle empty bin fields. These represent unknown values due to missing data. <ul style="list-style-type: none"> • Highest—Assigns null values to the highest bin • Lowest—Assigns null values to the lowest bin. <p>The lowest bin is always bin 1.</p>

Target internal bins Specifies the number of bins to fill between the end bins.

If you are performing equal-range binning, you may select this type of processing or **Bin width**, but not both. If you are performing equal-population binning, you may only perform internal-bin processing.

Bin width

Choose this option to perform equal-range binning.

If you are performing equal-range binning and want to select this type of processing rather than internal-bin processing, click **Bin width** and enter the number of units you want in each bin.

Logistics Regression

Logistic Regression enables you to perform machine learning by creating models from datasets that use binary objectives with input variables.

How to

Add Logistics Regression to workflow

This procedure describes how to add the Logistics Regression stage to a workflow in Flow Designer. Logistics Regression is a Machine Learning stage.

To create your model, you must first complete the **Model Properties** settings. The **Basic Options** and **Advanced Options** settings provide sufficient default settings to complete a job, but you can change those settings to meet your needs. After you run your job a limited version of the resulting model appears on the **Model Output** tab. The complete output is available in the Machine Learning Model Management tool.

1. In the **Stages** panel, scroll to **Machine Learning**, and drag the **Logistics Regression** stage onto the canvas.
2. Connect the stage to other stages.

The input stage must be the data source that contains the principal components for your model. An output stage is not required but you may connect one if you wish to capture your output independent of the Machine Learning Model Management tool.

For more information, see [Logistics Regression stage ports](#).

3. Double-click the Logistics Regression stage to open **Logistic Regression Options: Logistics Regression**.
4. On the **Model Properties** tab, configure the model name, number of principal components, and the input fields to be included in the analysis.
For more information about options on this tab, see [Model Properties tab](#) on page 41.
5. On the **Basic Options** tab, configure to use all factor level, to score input data, the transform, and how to handle missing data.
For more information about options on this tab, see [Basic Options tab](#) on page 42.

6. On the **Advanced Options** tab, configure whether to ignore constant fields, the PCA method, and convergence criteria.
For more information about options on this tab, see [Advanced Options tab](#) on page 42.
7. On the **Model Output** tab, view the metrics you are using to assess the fitted model.
For more information about this tab, see [Model Output tab](#) on page 45
8. Click **Apply** to save your changes.

Reference

Stage ports

Input port

The input stage must be the data source that contains the principal components for your model.

Output ports

Output is captured by the Machine Learning Model Management tool. The optional output ports allow you to pass output to subsequent stages in a workflow.

- | | |
|---------------------------|---|
| Model score port | Use this port to capture model scores independent of the Machine Learning Model Management tool. |
| Model metrics port | The optional model metrics port lets you output the model assessment metrics to a data file. This helps compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics. This port's functionality is determined by input and configuration of basic and advanced options in the stage settings. |

Model Properties tab

Options must be configured on this tab to perform an analysis.

- | | |
|------------------------|---|
| Model name | You can enter a custom name for the model to use as a reference. By default, Spectrum automatically generates a name. |
| Overwrite | Check this check box to overwrite an existing model with new data. |
| Objective field | The field for objective function values used for learning. |
| Description | Provides space to describe the model in a workflow. |
| Inputs | <p>This table shows input fields along with the data type and model data type. Check the Include check box to include data from a field in the model. In the Model Data Type column, specify whether an input field is a categorical, datetime, numeric, string, or uniqueid field.</p> <p>In the Model Data Type, click the drop-down list to specify whether each input field is to be used as a numeric, categorical, or datetime field.</p> |

Basic Options tab

- Standardize input fields** Check the check box to standardize the numeric columns to have zero mean and unit variance. This is the default.
If you do not use standardization, the results may include components dominated by variables appearing to have larger variances relative to other attributes as a matter of scale rather than true contribution.
- Score input data** Check this check box to add a column for the model prediction (score) to the input data.
- Prior** Check this check box if the data has been sampled and the mean of response does not reflect reality. Enter the prior probability for $p(y=1)$ in the text field. The default value is 0.5.
- Missing data** This option specifies how to handle missing data.
- **Skip**—Skips missing data.
 - **Impute means**—Adds the mean value for missing data.
- Sampling**
- **Percentage for training data**—Specify a value between 1 and 100 when the input data is randomly split into training and test data samples.
 - **Percentage for test data**—Enter the value of 100 minus the value entered in **Percentage for training data**.
 - **Seed for sampling**—Enter a number to ensure that when the data is split into test and train data in the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.

Advanced Options tab*Options*

- Ignore constant fields** Leave this check box checked to skip fields that have the same value for each record.
- Compute p values** Leave this check box checked to calculate p values for the parameter estimates
- Remove collinear column** Leave this check box checked to automatically remove collinear columns during model building. This option must be checked if **Compute p values** is also checked. This results in a 0 coefficient in the returned model.
- Include constant term (Intercept)** Leave this check box checked to include a constant term (intercept) in the model. This field must be checked if the **Remove collinear column** check box is also checked.
- Solver** Select a solver from in the drop-down list box.
- **Auto**— Solver will be determined based on input data and parameters.
 - **CoordinateDescent**—IRLSM with the covariance updates version of cyclical coordinate descent in the innermost loop.

- **CoordinateDescentNaive**—IRLSM with the naive updates version of cyclical coordinate descent in the innermost loop.
- **IRLSM**— Ideal for problems with a small number of predictors or for Lambda searches with L1 penalty.
- **LBFGS**— Ideal for datasets with many columns.

Note: **CoordinateDescent** and **CoordinateDescentNaive** are currently experimental.

Convergence Criteria

Maximum iterations	Specifies the number of training iterations that should take place.
Objective epsilon	Specifies the threshold for convergence. If the objective value is less than this threshold, the model will be converged. This must be a value between 0 and 1, exclusive. The default setting is 0.0001.
Beta epsilon	Specifies the threshold for convergence. If the objective value is less than this threshold, the model will be converged. If the L1 normalization of the current beta change is below this threshold, consider using convergence. This must be a value between 0 and 1, exclusive. The default setting is 0.0001.

Cross Validation

Seed for N fold	Leave this check box checked and enter a seed number to ensure that when the data is split into test and train data in the same manner each time you run the dataflow. Uncheck this field to get a random split each time you run the flow. The default setting is 15341.
N fold	Check this check box and enter the number of folds to perform cross validation.
Fold assignment	<p>Check this check box and select from the drop-down list if you are performing cross-validation. This field is applicable only if you entered a value in the N fold box and the Fold field is not specified.</p> <ul style="list-style-type: none"> • Auto—Allows the algorithm to automatically choose an option; currently it uses Random. • Modulo—Evenly splits the dataset into the folds and does not depend on the seed. • Random—Randomly splits the data into nfolds pieces; best for large datasets. • Stratified—Stratifies the folds based on the response variable for classification problems. Evenly distributes observations from the different classes to all sets when splitting a dataset into train and test data. This can be useful if there are many classes and the dataset is relatively small.
Fold field	If you are performing cross-validation, check this check box and select the field that contains the cross-validation fold index assignment from the drop-down list. This field is applicable only if you did not enter a value in N fold and Fold assignment .

Regularization

Regularization type Choose the appropriate regularization type. A common concern in predictive modeling is overfitting, when an analytical model corresponds too closely (or exactly) to a specific dataset and therefore may fail when applied to additional data or future observations. Regularization is one method used to mitigate overfitting.

- **Elastic Net Penalty**—Combines LASSO and Ridge Regression by acting as a variable selector while also preserving the grouping effect for correlated variables (shrinking coefficients of correlated variables simultaneously). Elastic Net is not limited by high dimensionality and can evaluate all variables when a model contains more variables than records.
- **LASSO**—(Least Absolute Shrinkage and Selection Operator) Selects a small subset of variables with a value of lambda high enough to be considered crucial. May not perform well when there are correlated predictor variables, as it will select one variable of the correlated group and remove all others. Also limited by high dimensionality; when a model contains more variables than records, LASSO is limited in how many variables it can select. Ridge Regression does not have this limitation. When the number of variables included in the model is large, or if the solution is known to be sparse, LASSO is recommended.
- **Ridge Regression**—Retains all predictor variables and shrinks their coefficients proportionally. When correlated predictor variables exist, Ridge Regression reduces the coefficients of the entire group of correlated variables towards equaling one another. If you do not want correlated predictor variables removed from your model, use Ridge Regression.

Value of alpha Check this check box and change the value if you do not want to use the default of .5. The alpha parameter controls the distribution between the ℓ_1 and ℓ_2 penalties. Valid values range between 0 and 1; a value of 1.0 represents LASSO, and a value of 0.0 produces ridge regression. The table below illustrates how alpha and lambda affect regularization.

Note: A single equals sign is an assignment operator meaning "is," while the double equals sign is an equality operator meaning "equal to."

Value of lambda Check this check box and specify a value if you do not want Logistic Regression to use the default method of calculating the lambda value, which is a heuristic based on training data. The lambda parameter controls the amount of regularization applied. For example, if lambda is 0.0, no regularization is applied and the alpha parameter is ignored.

Search for optimal value of lambda Check this check box to have Logistic Regression compute models for full regularization path. This starts at lambda max (the highest lambda value that makes sense—that is, the lowest value driving all coefficients to zero) and goes down to lambda min on the log scale, decreasing regularization strength at each step. The returned model will have coefficients corresponding to the optimal lambda value as decided during training.

- **Stop early if no relative improvement**—Check this check box to end processing when there is no more relative improvement on the training or validation set.

- **Maximum lamda search**—Check this check and enter the maximum number of lambdas to use during the process of lambda search.

Maximum active predictors Check this check box and enter the maximum number of predictors to use during computation. This value is used as a stopping criterion to prevent expensive model building with many predictors

Model Output tab

This tab shows the metrics you are using to assess the fitted model.

These fields cannot be edited. The Training column will always contain data. If you selected a train/test split on the **Basic Options** tab, the Test column will also be filled, unless you have selected an N Fold validation on the **Advanced Options** tab, in which case the N Fold column will be filled.

After you run your job, the resulting model is stored on the Spectrum Technology Platform server. Click the **Output** button to regenerate the output and click **Model details** to view the entire output in the Machine Learning Model Management tool.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical process that converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables known as principal components.

To create your model, you must first complete the **Model Properties** tab. The **Basic Options** and **Advanced Options** tabs provide default settings to complete a job, but you can change those settings to satisfy particular circumstances. You then run your job and a limited version of the resulting model appears on the **Model Output** tab. The complete output is available in the Machine Learning Model Management tool. If you are satisfied with the output of your model, you can then expose it and use it in a scoring dataflow.

How to

Add Principal Component Analysis (PCA) to workflow

1. In the **Stages** panel, scroll to **Machine Learning**, and drag the **PCA** stage onto the canvas.
2. Connect the stage to other stages.

The input stage must be the data source that contains the principal components for your model. An output stage is not required but you may connect one if you wish to capture your output independent of the Machine Learning Model Management tool.

For more information, see [PCA Stage ports](#).

3. Double-click the PCA stage to open **PCA Properties**.
4. On the **Model Properties** tab, configure the model name, number of principal components, and the input fields to be included in the analysis.

For more information about options on this tab, see [Model Properties tab](#) on page 46.

5. On the **Basic Options** tab, configure to use all factor level, to score input data, the transform, and how to handle missing data.

For more information about options on this tab, see [Basic Options tab](#) on page 47.

6. On the **Advanced Options** tab, configure whether to ignore constant fields, the PCA method, and convergence criteria.

For more information about options on this tab, see [Advanced Options tab](#) on page 47

7. Click **Apply** to save your changes.

Reference

Stage ports

Input port

The input stage must be the data source that contains the principal components for your model.

Output ports

Output is captured by the Machine Learning Model Management tool. The optional output ports allow you to pass output to subsequent stages in a workflow.

Model score port Use this port to capture model scores independent of the Machine Learning Model Management tool.

Model metrics port The optional model metrics port lets you output the model assessment metrics to a data file. This helps compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics. This port's functionality is determined by input and configuration of basic and advanced options in the stage settings.

Model Properties tab

Options must be configured on this tab to perform an analysis.

Model name You can specify a custom name for the model to use as a reference. By default, Spectrum automatically generates a name.

Overwrite Check this check box to overwrite an existing model with new data.

Principal components Enter the number of principal components that you want your model to contain.

Description Provides space to describe the model in a workflow.

Inputs This table shows input fields along with the data type and model data type. Check the **Include** check box to include data from a field in the model. In the Model Data Type column, specify whether an input field is a categorical, datetime, numeric, string, or uniqueid field.

Basic Options tab

User all factor level	<p>Specifies whether to retain the first principal component, which has the largest variance in the data.</p> <ul style="list-style-type: none"> • Check this option to retain the first principal component. • Uncheck this option to skip the first principal component. This is the default.
Transform	<p>Specify the transformation method for numeric columns in the training data.</p> <ul style="list-style-type: none"> • Demean—Subtracts the mean of each column. • Descale—Divides by the standard deviation of each column. • None—No transform. • Normalize—Demeans and divides each column by its range (maximum minus minimum). • Standardize—Uses zero mean and unit variance. This is the default transform.
Missing data	<p>Specifies whether to impute missing entries with the column mean.</p> <ul style="list-style-type: none"> • Skip—Choose this option to skip missing data. This is the default setting. • Impute mean—Choose this option to add the mean value for any missing data.

Advanced Options tab

Ignore constant fields	<p>Check this check box to skip fields that have the same value for each record, since no information can be gained from them. This option is checked by default.</p>
PCA Method	<p>Specify the algorithm to use for computing the principal components:</p> <ul style="list-style-type: none"> • GLRM—Fits a generalized low-rank model with L2 loss function and no regularization; solves for the SVD using local matrix algebra. • GramSVD—Uses a distributed computation of the Gram matrix, followed by a local SVD using the JAMA package. This is the default method. • Power—Computes the SVD using the power iteration method. • Randomized—Uses the randomized subspace iteration method.
Maximum iteration	<p>Specifies the number of training iterations. The value must be between 1 and 1e6 and the default is 1000.</p>

3 - Machine Learning Model Management

In this section

Accessing Machine Learning Model Management.....	49
Model Assessment.....	50
Binning Management.....	57
Configuration Settings.....	58



Accessing Machine Learning Model Management

You can access Machine Learning Model Management from the Welcome page or directly.

- **From the Welcome page**
- **Directly from a Web browser**

From the Spectrum Technology Platform Welcome Page

1. Open a web browser and go to the Spectrum Technology Platform Welcome Page at:

```
servername:port
```

For example, if you installed Spectrum Technology Platform on a computer named "myspectrumplatform" and it is using the default port 8080, you would go to:

```
myspectrumplatform:8080
```

2. Click **Spectrum Analytics Scoring**.
3. Click **Spectrum Machine Learning Model Management**.
4. If you are prompted, enter a valid Spectrum Technology Platform **User name** and **Password**.

Directly from a web browser

1. Open a web browser and navigate to the Spectrum Technology Platform Machine Learning Model Management page at:

```
servername:port/machinelearning
```

For example, if you installed Spectrum Technology Platform on a computer named "myspectrumplatform" and it is using the default port 8080, you would go to:

```
myspectrumplatform:8080/machinelearning
```

2. If you are prompted, enter a valid Spectrum Technology Platform **User name** and **Password**.

Model Assessment







Introduction to Model Assessment

The **Model Assessment** tab in Machine Learning Model Management shows a list of all machine learning models on your Spectrum Technology Platform server. You can filter this list by entering a string in the text box; every field in the table will be searched for that string.

Several operations can be performed on these models. You can import, export, expose, unexpose, or delete models. Exposed models are used in the Java Model Scoring stage to score new data using formulas created when you fit machine learning models. Additionally, you can view detailed information for each model; the details returned depend on the type of model whose data you are viewing. Finally, you can compare any two models of the same type. This comparison shows side-by-side the same information that is on the Model Detail tab for each of the models you are comparing.

Model Assessment Operations

Perform these operations by selecting a model and clicking the appropriate button:

	View model output detail. You can also access this information from the K-Means Clustering and Logistic Regression stages by clicking "For model details click here" on the Model Output tab.
	Compare models.
	Import a model from a specific path. Select whether to overwrite an existing model of the same name, if appropriate.
	Export a model to a specific path. Select whether to overwrite an existing model of the same name, if appropriate.
	Expose the model to make it available to the Java Model Scoring stage. If a model is not exposed, it cannot be used for scoring.
	Unexpose the model.



Delete the model.

Note: You cannot delete an exposed model. However, at this time there is no inherent security that prevents a user from deleting another user's models.

Model Assessment tab

The Model Detail screen shows the following information for all models:

Model Name	The name of the model.
Status	Whether the model is exposed or unexposed.
Model Type	The type of machine learning model.
User	The user name of the person who created the model.
Description	The description of the model if one was provided when it was created.
Dataflow Name	The name of the dataflow that produced the model.
Creation Time	The date and time the model was created.

Additional details are provided based on the model type.

K-Means Clustering Details

The Model Detail screen includes the following information for K-Means Clustering models:

Model Summary

Provides training data for the following:

- Number of Rows
- Number of Clusters
- Number of Categorical Columns
- Number of Iterations
- Within Cluster Sum of Squares
- Total Sum of Squares
- Between Cluster Sum of Squares

Metrics

Provides training, test, and n-fold data for the following:

- Total within cluster sum of squares

- Total sum of squares
- Between cluster sum of squares

Centroid Statistics

Provides the following training, test, and n-fold data for each centroid:

- Size
- Within cluster sum of squares

Cluster Means

Provides detailed information for each centroid. Content varies based on input data. A cluster is a group of observations from a data set identified as similar according to a particular clustering algorithm

Standardized Cluster Means

Provides standardized information for each centroid. Content varies based on input data.

Logistic Regression Details

The Model Detail screen includes the following information for Logistic Regression models:

Metrics

Provides training, test, and n-fold data for the following:

- Mean squared error (MSE)
- Root mean squared error (RMSE)
- Number of observations
- R-squared (R2)
- Logarithmic loss (Logloss)
- Area under the curve (AUC)
- Precision-recall area under the curve (PR AUC)
- Gini coefficient
- Mean per class error
- Akaike information criterion (AIC)
- Lambda
- Residual deviance
- Null deviance
- Null degree of freedom
- Residual degree of freedom

Maximum Metrics Threshold

Provides the Training Maximum Metrics Threshold for training, test, and n-fold data using the following metrics:

- max f1
- max f2
- max f0point5
- max accuracy
- max precision
- max recall
- max specificity
- max absolute_mcc
- max min_per_class_accuracy
- max mean_per_class_accuracy

Confusion Matrix

Illustrates the performance of a model on a set of training, test, and n-fold data for which the true values are known.

Standardized Coefficient Chart

Shows the most important predictors by providing the relative value of the coefficients, which indicates how much a change in input changes the objective.

GLM Coefficients

Shows coefficients for a Generalized Linear Model, which estimates regression models for outcomes following exponential distributions.

AUC Curves

Area under the curve; determines which of the used models predicts the classes best using training, test, and n-fold data.

Lift/Gain Curves

Evaluate the prediction ability of a binary classification model using training, test, and n-fold data.

Linear Regression Details

The Model Detail screen includes the following information for Linear Regression models:

Metrics

Provides training, test, and n-fold data for the following:

- Mean squared error (MSE)
- Root mean squared error (RMSE)
- Number of observations
- R-squared (R2)
- Mean residual deviance
- Mean absolute error (MAE)
- Root mean squared logarithmic error (RMSLE)
- Akaike information criterion (AIC)
- Lambda
- Residual deviance
- Null deviance
- Null degree of freedom
- Residual degree of freedom

Standardized Coefficient Chart

Shows the most important predictors by providing the relative value of the coefficients, which indicates how much a change in particular predictor coefficient value changes the objective value positively or negatively. Also charts the top 25 coefficients in the model.

GLM Coefficients

Shows coefficients for a Generalized Linear Model, which estimates regression models for outcomes following exponential distributions.

Random Forest Regression Details

The Model Detail screen includes the following information for Random Forest Regression models:

Metrics

Provides training, test, and n-fold data for the following:

- Mean squared error (MSE)
- Root mean squared error (RMSE)
- Number of observations
- R-squared (R2)
- Mean residual deviance
- Mean absolute error (MAE)
- Root mean squared logarithmic error (RMSLE)

Variable Importances

Provides importance values for each variable using the following metrics:

- Relative importance
- Scaled importance
- Percentage

Also charts the top 25 variables in the model.

Random Forest Classification Details—Binomial

The Model Detail screen includes the following information for **binomial** Random Forest Classification models:

Metrics

Provides training, test, and n-fold data for the following:

- Mean squared error (MSE)
- Root mean squared error (RMSE)
- Number of observations
- R-squared (R2)
- Logloss
- Area under the curve (AUC)
- Precision-recall area under the curve (PR AUC)
- Gini
- Mean per class error

Maximum Metrics Threshold

Provides the Training Maximum Metrics Threshold for training, test, and n-fold data using the following metrics:

- max f1
- max f2
- max f0point5
- max accuracy
- max precision
- max recall
- max specificity
- max absolute_mcc
- max min_per_class_accuracy
- max mean_per_class_accuracy

Confusion Matrix

Illustrates the performance of a model on a set of training, test, and n-fold data for which the true values are known.

Variable Importances

Provides importance values for each variable using the following metrics:

- Relative importance
- Scaled importance
- Percentage

Also charts the top 25 variables in the model.

AUC Curves

Area under the curve; determines which of the used models predicts the classes best using training, test, and n-fold data.

Lift/Gain Curves

Evaluates the prediction ability of a binary classification model using training, test, and n-fold data.

Random Forest Classification Details—Multinomial

The Model Detail screen includes the following information for **multinomial** Random Forest Classification models:

Metrics

Provides training, test, and n-fold data for the following:

- Mean squared error (MSE)
- Root mean squared error (RMSE)
- Number of observations
- R-squared (R²)
- Logloss
- Mean per class error

Confusion Matrix

Illustrates the performance of a model on a set of training, test, and n-fold data for which the true values are known.

Variable Importances

Provides importance values for each variable using the following metrics:

- Relative importance
- Scaled importance
- Percentage

Also charts the top 25 variables in the model.

Principal Component Analysis Details

The Model Detail screen includes the following information for PCA models:

Importance of components

Shows the principal components in order of importance based on the following metrics:

- Standard deviation
- Proportion of variance
- Cumulative proportion

Rotation

Charts the matrix of variable loadings, the weight by which each standardized original variable should be multiplied to get the component score.

Binning Management






Introduction to Binning Management

The Binning Management tab in Machine Learning Model Management shows a list of all binning on your Spectrum Technology Platform server. You can filter this list by entering a string in the text box; every field in the table will be searched for that string.

Several operations can be performed on binning. You can import, export, expose, unexpose, or delete binning. Exposed binnings are used by the Binning Lookup stage to apply previously defined binning to new data.

Binning Management Operations

Perform these operations by selecting a binning and clicking the appropriate button:



	Import a binning. Select whether to overwrite an existing binning of the same name, if appropriate.
	Export a binning. Select whether to overwrite an existing binning of the same name, if appropriate.
	Expose the binning to make it available to the Binning Lookup stage. If a binning is not exposed, it cannot be used for lookup.
	Unexpose the binning.
	Delete the binning. A binning cannot be deleted if it is exposed. Note: A user can delete any binning created by any user. At this time there are no user-specific permissions.



Configuration Settings

This page allows configuration of Java environment settings from within the application.

You can display these settings by clicking **Configuration Settings** on the **Machine Learning Model Management** toolbar.

Option	Description
Pool size	<p>This option specifies the maximum number of concurrent requests that will be handled by this database. This allows reuse of connections rather than opening a connection each time that a client requests a connection. This can significantly enhance performance.</p> <p>You will generally see the best results by setting this between one half to twice the number of CPUs on the server. The optimum size for most modules is the same as the number of CPUs.</p>

Option	Description
Minimum memory (MB)	<p>This specifies initial or minimum heap memory in megabytes used by the component in megabytes in the Java environment. Fine-tuning this setting in combination with the maximum memory setting can enhance performance in high load environments. This value must be greater than zero, but cannot exceed the Maximum memory setting.</p> <p>Note: The memory allocation for Machine Learning should be three to four times the size of the input file used in jobs where models are created. We recommend that the Minimum memory setting be at least 1 GB.</p>
Maximum memory (MB)	<p>Specifies the maximum heap memory in megabytes used by the component in megabytes in the Java environment. Fine-tuning this setting in combination with the minimum memory setting can enhance performance in high load environments. This value must be at least as large as the Minimum memory, but cannot exceed 65336 MB. The default value if this is left empty is 65336 MB.</p>
Java Properties	<p>Click to expand the list of Java properties that have been added for Machine Learning.</p> <ul style="list-style-type: none"> • To add a property, click the Add property button . • To delete properties, check the check box next to properties that you want to delete and click the Delete property button . • To edit a property name, click in the Name column and enter changes to the name. • To edit the value for property, click in the Value column and enter changes to the value. <p>Note: The Machine Learning Module uses port number 15431.</p>

Option	Description
Environment Variables	<p>Click to expand the list of environment variables that have been added for Machine Learning.</p> <ul style="list-style-type: none"> • To add a variable, click the Add variable button . • To delete variables, check the check box next to variables that you want to delete and click the Delete variable button . • To edit a variable name, click in the Name column and enter changes to the name. • To edit the value for variable, click in the Value column and enter changes to the value.
Process Arguments	<p>Expand this field to define process arguments that are not memory-related settings and cannot be expressed as Java properties.</p> <ul style="list-style-type: none"> • Process arguments—Enter process arguments as they would appear on the command line. • Reply Timeout (seconds)—Enter timeout in seconds for the process.

4 - Data Science

Demonstration Flows

In this section

Introduction.....	62
Supervised Learning: Loan Default Prediction.....	62
Unsupervised Learning: Segmentation.....	63



Introduction

Spectrum Machine Learning and Spectrum Analytics Scoring, along with modules to prepare data for modeling, are part of the Spectrum Data Science offer. These demonstrations show examples of data preparation, modeling, and model scoring. You can create your own dataflows using the step-by-step instructions, or you can use the provided dataflows as a reference.

Supervised Learning: Loan Default Prediction



Download the supervised learning demonstration

The Data Science supervised learning demonstration conducts loan default prediction using Lending Club data. It utilizes several files that together demonstrate the functionality of the Spectrum Technology Platform Data Science Solution in Enterprise Designer.

`Spectrum_DataScience_Supervised_Learning.zip` includes the following files:

- `Spectrum_DataScience_Supervised_Learning.pdf`—Documentation that walks you through how to build and use the single categorizer dataflow, the scoring dataflow, and all supporting files.
- `Data.zip`—The required input files, test files, and training files for each of the included dataflows.
 - `loan.csv`
 - `LoanStats_2016Q1.csv`
 - `LoanStats_2016Q2.csv`
 - `LoanStats_2016Q3.csv`
 - `testData.txt`
 - `testDataCollege.txt`
 - `testDataStable.txt`
 - `testDataThankful.txt`
 - `trainData.txt`
 - `trainDataCollege.txt`
 - `trainDataStable.txt`
 - `trainDataThankful.txt`
 - `training.xml`
 - `trainingCollege.xml`
 - `trainingStable.xml`
 - `trainingThanks.xml`

- `Lending_Club_Demo_DF_(V12.1).zip`—The dataflows for Spectrum Technology Platform 12.1.
 - `LendingClub_2007_2016Q12_v121_MultipleCategorizers.df`
 - `LendingClub_2007_2016Q1Q2_v121_SingleCategorizer.df`
 - `LendingClub_2016Q3_v121_SingleCategorizer_Scoring.df`
- `Lending_Club_Demo_DF_(V12.2).zip`—The dataflows for Spectrum Technology Platform 12.2.
 - `LendingClub_2007_2016Q12_v122_MultipleCategorizers.df`
 - `LendingClub_2007_2016Q1Q2_v122_SingleCategorizer.df`
 - `LendingClub_2016Q3_v122_SingleCategorizer_Scoring.df`
- `ReadMe.txt`—High-level descriptions and instructions for the previously mentioned files.

You can create your own dataflow by following the step-by-step instructions in the documentation, or you can use the included dataflows as references to confirm what the individual completed stages and dataflows as a whole should look like.

Unsupervised Learning: Segmentation



Download the unsupervised learning demonstration

The Data Science unsupervised learning demonstration conducts segmentation using Consumer Expenditure data. It utilizes several files that together demonstrate the functionality of the Spectrum Technology Platform Data Science Solution in Spectrum Enterprise Designer.

`Spectrum_DataScience_Unsupervised_Learning.zip` includes the following files:

- `Spectrum_DataScience_Unsupervised_Learning.pdf`—Documentation that walks you through how to build and use the primary dataflow, the subflow, the scoring dataflow, and all supporting files.
- `Data.zip`—The required input files and output files for each of the included dataflows.
 - **Input folder**—The required input files for each of the included dataflows
 - **Output folder**—The required output files for each of the included dataflows
 - **PythonBased folder**—Required input and output files to use optional Python processing in lieu of Group Statistics and Transformer stages in primary dataflow
- `Consumer_Expenditure_Demo_DF_(v12.1).zip`—The dataflows for Spectrum Technology Platform 12.1.
 - `ConsumerExpenditure_v121_sampleandcluster.df`
 - `ConsumerExpenditure_v121_sampleandcluster_subflow.df`

- `ConsumerExpenditure_v121_score.df`
- `ConsumerExpenditure_v121_subflow.df`
- **PythonBased** folder—Required dataflows, process flows, bat script, Python script and documentation to use optional Python processing in lieu of Group Statistics and Transformer stages in primary dataflow.
- **Consumer_Expenditure_Demo_DF_(v12.2).zip**—The dataflows for Spectrum Technology Platform 12.2
 - `ConsumerExpenditure_v122_sampleandcluster.df`
 - `ConsumerExpenditure_v122_sampleandcluster_subflow.df`
 - `ConsumerExpenditure_v122_score.df`
 - `ConsumerExpenditure_v122_subflow.df`
 - **PythonBased** folder—Required dataflows, process flows, bat script, Python script and documentation to use optional Python processing in lieu of Group Statistics and Transformer stages in primary dataflow.
- `ReadMe.txt`—High-level descriptions and instructions for the previously mentioned files.

You can create your own dataflow by following the step-by-step instructions in the documentation, or you can use the included dataflows as references to confirm what the individual completed stages and dataflows as a whole should look like.



2 Blue Hill Plaza, #1563
Pearl River, NY 10965
USA

www.precisely.com

© 2007, 2021 Precisely. All rights reserved.