## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 11</td>
<td>GLM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overview of GLM</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>GLM Properties</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>GLM Results</td>
<td>65</td>
</tr>
<tr>
<td>Chapter 12</td>
<td>Gradient Boosting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overview of Gradient Boosting</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>Gradient Boosting Properties</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>Gradient Boosting Results</td>
<td>75</td>
</tr>
<tr>
<td>Chapter 13</td>
<td>Imputation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overview of Imputation</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>Imputation Properties</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>Imputation Results</td>
<td>80</td>
</tr>
<tr>
<td>Chapter 14</td>
<td>Linear Regression</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overview of Linear Regression</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>Linear Regression Properties</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>Linear Regression Results</td>
<td>88</td>
</tr>
<tr>
<td>Chapter 15</td>
<td>Logistic Regression</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overview of Logistic Regression</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>Logistic Regression Properties</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>Logistic Regression Results</td>
<td>98</td>
</tr>
<tr>
<td>Chapter 16</td>
<td>Manage Variables</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overview of Manage Variables</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>Manage Variables Properties</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>Manage Variables Results</td>
<td>101</td>
</tr>
<tr>
<td>Chapter 17</td>
<td>Model Comparison</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overview of Model Comparison</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>Model Comparison Properties</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>Model Comparison Results</td>
<td>106</td>
</tr>
<tr>
<td>Chapter 18</td>
<td>Neural Network</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overview of Neural Network</td>
<td>109</td>
</tr>
<tr>
<td></td>
<td>Neural Network Properties</td>
<td>109</td>
</tr>
<tr>
<td></td>
<td>Neural Network Results</td>
<td>117</td>
</tr>
<tr>
<td>Chapter 19</td>
<td>Open Source Code</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overview of Open Source Code</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>Open Source Code Properties</td>
<td>122</td>
</tr>
<tr>
<td></td>
<td>Open Source Code Editor User Interface</td>
<td>122</td>
</tr>
<tr>
<td></td>
<td>Open Source Code Results</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td>Using the Open Source Code Node</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td>Installation of Python or R Software</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>Common Questions</td>
<td>126</td>
</tr>
<tr>
<td>Chapter 20</td>
<td>Quantile Regression</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overview of Quantile Regression</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>Quantile Regression Properties</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>Quantile Regression Results</td>
<td>133</td>
</tr>
</tbody>
</table>
### Chapter 21 / Replacement
- Overview of Replacement .................................................. 137
- Replacement Properties .................................................. 137
- Replacement Results .................................................. 138

### Chapter 22 / SAS Code
- Overview of SAS Code .................................................. 141
- SAS Code Properties .................................................. 141
- Code Editor User Interface ........................................... 142
- SAS Code Results .................................................. 148
- Using the SAS Code Node ............................................ 149

### Chapter 23 / Save Data
- Overview of Save Data .................................................. 151
- Save Data Properties .................................................. 151
- Save Data Results .................................................. 152

### Chapter 24 / Score Code Import
- Overview of Score Code Import ....................................... 153
- Using the Score Code Import Node .................................... 153
- Score Code Import Properties ........................................ 154
- Score Code Import Results ............................................ 154

### Chapter 25 / SVM
- Overview of SVM .................................................. 157
- SVM Properties .................................................. 157
- SVM Results .................................................. 161

### Chapter 26 / Text Mining
- Overview of Text Mining .................................................. 163
- Text Mining Properties .................................................. 163
- Text Mining Results .................................................. 164

### Chapter 27 / Transformations
- Overview of Transformations .......................................... 167
- Transformations Properties ........................................... 167
- Transformations Results ............................................ 170

### Chapter 28 / Variable Clustering
- Overview of Variable Clustering ........................................ 171
- Variable Clustering Properties ......................................... 172
- Variable Clustering Results ........................................... 173

### Chapter 29 / Variable Selection
- Overview of Variable Selection ......................................... 175
- Variable Selection Properties ........................................... 175
- Variable Selection Results ............................................ 186
Overview of Anomaly Detection

The **Anomaly Detection** node is a Data Mining Preprocessing node that identifies and excludes anomalies (observations) using the Support Vector Data Description (SVDD). Briefly, the SVDD formulation identifies outliers by determining the smallest possible hypersphere (built using support vectors) that encapsulates the training data points. The SVDD then excludes those data points that lie outside of the sphere that is built from the training data.

Anomaly Detection Properties

General Properties

- **Standardize interval inputs** — specifies whether to standardize interval input variables.
- **Gaussian kernel bandwidth** — specifies the bandwidth parameter for the Gaussian kernel (or Radial Basis Function). The default used by the node is tentative, so it is highly recommended that you try different values to determine what is best for the data.
- **Solver** — specifies the type of optimization solver that is used for Support Vector Data Description (SVDD) training. Possible values are **Active set** and **Stochastic subset**. The default value is **Active set**.
Anomaly fraction — specifies the expected fraction of the training data that consists of anomalies. The default value is 0.01. If you specify Stochastic subset as the solver, this value is not used.

Solver tolerance — specifies the tolerance value for the solver. The default value is 0.001.

Maximum number of iterations — specifies the maximum number of iterations for the solver.

Include counts and distance histogram reports — specifies whether to generate a counts table, which displays the number of observations marked as anomalies during training. The contents table is not generated because that would require an additional step of scoring the train data.

Stochastic Subset Options

These configurations are available only when you select Stochastic subset as the solver.

Maximum support vectors — specifies the maximum number of support vectors for the stochastic solver. The default value is 350.

Sample size — specifies the number of observations that are sampled in each iteration of the stochastic solver. You must specify a value greater than 0. The default value is 5.

Threshold tolerance — specifies the tolerance that is used to detect convergence of the threshold value. The default value is 0.01.

Center tolerance — specifies the tolerance that is used to detect convergence of the center. The default value is 0.01.

Convergence criterion — specifies a convergence criterion for the stochastic solver. If the radius and center values converge for this number of consecutive iterations, then convergence is declared. The default value is 3.

Random seed — specifies the seed value that is used for selecting a random sample. The default value is 12345.

Scored Output Roles

Anomaly indicator role — specifies how the anomaly indicator (the variable that indicates whether an observation is an anomaly) is used. Possible values are Input and Filter. Select Input to use the indicator as an input in the subsequent node. Select Filter to apply the indicator to the scoring data to exclude these rows. The default value is Filter.

Anomaly distance role — specifies the role that is assigned to the SVDD distance variable. Possible values are Rejected and Input. Select Input to use the distance variable as an input in the subsequent node. Select Rejected to set the variable to Rejected.
After running the node, you can open the Results window by right-clicking the node and selecting Results from the pop-up menu.

Results

- **Anomaly Counts** — displays the anomaly counts after scoring the train data. The anomaly counts include number and percentage of anomalies, observations used in training, observations with missing inputs, and total observations. If you do not select *Include counts and distance histogram reports*, the table will not be displayed.

- **SVDD Distance Histogram** — displays a bar chart that shows the distribution of observations using their respective SVDD distance (after scoring the train data). The distance is divided into 20 bins of equal size. Bins are color-coded to indicate whether the bars are less than or greater than the threshold that marks the boundary when identifying anomalies. If you do not select *Include counts and distance histogram reports*, this chart is not selected.

- **Training Results** — displays the results of the training procedure, which include the following:
  - number of support vectors
  - number of support vectors on boundary
  - number of outliers
  - number of dropped observations
  - threshold $R^2$ value
  - constant value
  - run time

- **Optimization Summary** — displays a summary of the anomaly detection run itself, including the number of iterations, objective value, infeasibility, optimization status, and the degenerate indicator variable.

- **Properties** — specifies the various properties selected before you run the node.

- **Output** — displays the SAS output of the anomaly detection run.
Overview of Batch Code

The Batch Code node enables you to import batch code models that were created in SAS Enterprise Miner. This node does not perform batch retraining of models that were created in Model Studio.

When the Batch Code node runs, a sample of the input data is provided to the MVA client and the batch code that represents the SAS Enterprise Miner process flow diagram is run on that sample. After the batch code runs, Model Studio retrieves either the model score code or the analytic store and EPCODE. The retrieved code is used to score the original input data in Model Studio. This scored data is used to produce the assessment results.

The target variable metadata in SAS Enterprise Miner and Model Studio must be identical.

Using the Batch Code Node

To use the Batch Code node, perform the following steps:

1. Build a process flow diagram in SAS Enterprise Miner. Your process flow diagram must end in a Score node.

2. Right-click the Score node in your diagram and select Export Path as SAS Program. Ensure that Run this path is enabled in the Export Path as SAS Program window. Specify a location to save the batch code and click OK.

3. In a Model Studio pipeline, add a Batch Code node to your pipeline.
4 In the **Batch Code** node properties pane, click **Open** and navigate to the file that you saved earlier. The batch code that you created in SAS Enterprise Miner is opened in the code editor.

5 Click  to save this code. Then click **Close**.

6 Continue building your pipeline. When you run your pipeline, the **Batch Code** node runs alongside all other nodes.

---

**Batch Code Properties**

- **Code editor** — invokes the Batch Code Editor. For more information about the SAS Code Editor see Code Editor User Interface on page 142.

- **Data Sample** — specifies the data sampling strategy.
  - **Sampling method** — specifies whether to perform **Simple random** sampling, **Stratified** sampling, or no data sampling.
  - **Sample using** — specifies whether you want to include a fixed number of observations or a percentage of the total observations in the sample.
  - **Number of Observations** — specifies the fixed number of observations to create the data sample.
  - **Percentage of Observations** — specifies the percentage of total observations to create the data sample.

- **Model Interpretability** — specifies the method of model interpretation in order to show how the model responds to changing inputs.
  - **Explain Model Inputs** — specifies the following parameters for the LIME algorithm:
    - **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.
    - **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.
    - **Cluster-based ICE plots** — the maximum number of variables (1-10).
  - **Explain Individual Predictions** — specifies whether to include **Number of clusters**, **Number of perturbed instances**, and **Number of iterations**. You can specify the **Distance metric** as either **Cosine** or **K prototype**. You can specify the **Kernel width** and the **Number of inputs to use in the explanation**. You can specify whether to **Create a score code for assigning clusters**.
Batch Code Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Node**
- **Path Score Code** — provides the optimized SAS code from SAS Enterprise Miner.
- **Score Inputs** — displays a table of data on the input variables for scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.
- **Score Outputs** — displays a table of data on the predicted response variable for scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.
- **EM Batch Code** — provides the batch code that was created by SAS Enterprise Miner.

**Assessment**
- **Predicted Reports** — displays the predicted and target mean as a function of the depth for the model. The predicted and target mean are given for each of the data roles. To examine the predicted mean as a function of the target mean, use the drop-down menu in the upper right corner. This result is displayed only if the target is an interval variable.
- **Lift Reports** — displays the cumulative lift as a function of the depth for the model. The cumulative lift is given for each of the data roles. To examine other statistics as a function of depth, use the drop-down menu in the upper right corner. Other statistics include lift, gain, % captured response, cumulative % captured response, % response, and cumulative % response. This result is displayed only if the target is a class variable.
- **ROC Reports** — displays the ROC (receiver operating characteristic) chart for a model, giving the sensitivity as a function of 1-specificity. The sensitivity is given for each of the data roles. To examine other statistics, use the drop-down menu in the upper right corner. Other statistics include accuracy and F1 score. This result is displayed only if the target is a class variable.
- **Fit Statistics** — displays a table of the fit statistics for the model, broken down by data role.

**Model Interpretability**
- **Input Relative Importance** — displays a table of the relative importance of the input variables, including variable level and label.
- **Partial Dependence** — displays the functional relationship between the input variable and the model prediction.
- **LIME Plots** — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed around a particular observation to a create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when **Explain Individual Predictions** is selected.
- **Local Instances** — displays the values for each input variable of the cluster center.

- **Local Explanation** — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.

- **Local Model** — displays the parameter estimates of the local linear model that is used to predict the cluster center.

- **Prediction** — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

- **LIME Score Code** — displays the SAS code that Model Studio used to assign observations to clusters. This is displayed only when **Create score code for assigning clusters** is selected.

- **Individual Conditional Expectation** — displays the functional relationship between the input variable and the model prediction. The relationships for all cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.
The Bayesian Network node is a Supervised Learning node that fits a Bayesian network model for a nominal target. A Bayesian network is a directed, acyclic graphical model in which the nodes represent random variables, and the links between the nodes represent conditional dependency between two random variables. The structure of a Bayesian network is based on the conditional dependency between the variables. As a result, a conditional probability table is created for each node in the network. Because of these features, Bayesian networks can be effective predictive models during supervised data mining. The following Bayesian network structures are available:

- Naive — A naive Bayesian network connects the target variable to each input variable. There are no other connections between the variables because the input variables are assumed to be conditionally independent of each other.

- Tree-Augmented — A Tree-Augmented Bayesian network connects the target variable to each input variable and connects the input variables in a tree structure. The tree that connects the input variables is based on the maximum spanning tree algorithm. In a Tree-Augmented network, each node can have at most two parents, one of which must be the target variable.

- Parent Child (PC) — A PC Bayesian network connects the target variable to each input variable. However, input variables can be either a parent of the target variable or a child of the target variable. The Bayesian Information Criterion is used to determine whether an input variable is a parent of the target variable or a child of the target variable.

- Markov Blanket — A Markov Blanket Bayesian network creates a set of connections between the target variable and the input variables. But it also permits connections between certain input variables. Only the children of the target variable can have an additional parent node. All other variables are
conditionally independent of the target variable and thus do not affect the classification model. The Bayesian Information Criterion is used to determine whether an input variable is a parent of the target variable or a child of the target variable.

Bayesian Network Properties

General Properties

- **Missing class inputs** — specifies how to handle missing values for class inputs. Possible values are **Exclude**, **Impute with mode**, and **Impute with level**. Choose **Exclude** to ignore the observations that have missing values in any of the class input variables. Choose **Impute with mode** to replace the missing values in any class input variable by the mode of the variable. Finally, choose **Impute with level** to treat the missing values in any class input variable as a separate level of the variable. The default value for this option is **Exclude**.

- **Missing interval inputs** — specifies how to handle missing values for interval inputs. Possible values are **Exclude** and **Impute with mean**. Choose **Exclude** to ignore the observations that have missing values in any of the interval variables. Choose **Impute with mean** to replace the missing values in any interval variable by the mean of the variable. The default value for this option is **Exclude**.

- **Number of bins** — specifies the number of bins for interval variables. Possible values range from 2 to 100. The default value is 10.

Variable Selection Options

- **Prescreen variables** — specifies whether to prescreen variables using independence tests between the target and each input variable. By default, this option is selected.

- **Variable selection** — specifies whether to select variables using conditional independence tests between the target and each input variable given the network. By default, this option is deselected.

- **Independence test statistics** — specifies the choice of statistics used for the independence test. Possible values are **Chi-Square**, **G-Square**, and **Chi and G-Square**. Select **Chi and G-Square** to use both independence test statistics. Both statistics must be satisfied for the independence test. The default value is **G-Square**.

- **Significance level** — specifies the significance level (p-value) for independence testing using Chi-Square and G-Square. The smaller the value you specify, the fewer that variables that are selected. The default value is 0.2.
Network Structure Options

- **Network structure** — specifies the network structure. This option enables you to select multiple structures. The following options are available:
  - Naive
  - Tree-Augmented
  - Parent-Child
  - Markov Blanket

If you want to choose the best structure among several structures, you can select multiple values in any combination. For more detail on these networks, please see the overview of this chapter. By default, all **Network structure** options are selected.

- **Maximum parents** — specifies the maximum number of parents that is allowed for each node in the network structure. Possible values range from 1 to 16. The default value is 5.

- **Choose best number of parents** — specifies whether to choose the best number of parents by trying from 1 to the value of the **Maximum parents** property above. By default, this option is selected.

- **Parenting method** — specifies the parenting method at each node. Possible values are One parent and Set of parents. Select One parent to add the best possible candidate as a parent of the node. Select Set of parents to test a set of variables among possible candidates to be the parents of each node. Then, the best set of variables are added as the parents of the node. The default value is Set of parents.

Binary Classification Cutoff

- **Specify node binary classification cutoff** — specifies whether to use the binary classification cutoff that is specified below for the node. Deselect to use the project binary classification cutoff for determining the predicted value for a binary target based on the posterior probabilities. By default, this value is deselected.

- **Node binary classification cutoff** — specifies the cutoff to use in the node for determining the predicted value for a binary target based on the posterior probabilities. This option is available only if you select **Specify node binary classification cutoff**. The default value is 0.5.

Model Interpretability

Specifies the method of model interpretation in order to show how the model responds to changing inputs.
**Explain Model Inputs** — specifies the following parameters for the LIME algorithm:

- **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.

- **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.

- **Cluster-based ICE plots** — the maximum number of variables (1-10).

**Explain Individual Predictions** — specifies whether to include **Number of clusters, Number of perturbed instances, and Number of iterations**. You can specify the **Distance metric** as either **Cosine** or **K prototype**. You can specify the **Kernel width** and the **Number of inputs to use in the explanation**. You can specify whether to **Create a score code for assigning clusters**.

---

**Bayesian Network Results**

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Node**

- **Bayesian Network** — displays the Bayesian network that the node created.

- **Validation Results** — displays a table of statistics for the automatic selection process. These statistics include best model, misclassification errors, significance threshold, prescreening, variable selection, structure, parenting method, and maximum number of parents.

- **Variable Selection** — displays a table of various computed statistics for each variable. These statistics include whether the variable was selected, chi-square, g-square statistics, degrees of freedom, \( p \)-value of chi-square, \( p \)-value of g-square statistics, mutual information, and conditional variables.

- **Order of Input Variables** — displays a table of input variables ranked by their score.

- **Score Inputs** — displays a table of input variables used in scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.

- **Score Outputs** — displays a table of predicted response variables that are generated during scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.

- **Path EP Score Code** — displays the SAS score code that was created by the node. The score code can be used outside of the Model Studio environment to score new data.

- **DS2 Package Code** — displays the SAS DS2 package code that was created by the node. The score code can be used outside of the Model Studio environment to score new data.

- **Train Code** — displays the SAS code that Model Studio used to train the node.
- **Properties** — specifies the various properties that are selected before running the node.

- **Output** — displays the SAS output of the Bayesian network run.

**Assessment**

- **Lift Reports** — displays the cumulative lift as a function of the depth for the model. The cumulative lift is given for each of the data roles. To examine other statistics as a function of depth, use the drop-down menu in the upper right corner. Other statistics include lift, gain, % captured response, cumulative % captured response, % response, and cumulative % response. This result is displayed only if the target is a class variable.

- **ROC Reports** — displays the ROC (receiver operating characteristic) chart for a model, giving the sensitivity as a function of 1-specificity. The sensitivity is given for each of the data roles. To examine other statistics, use the drop-down menu in the upper right corner. Other statistics include accuracy and F1 score. This result is displayed only if the target is a class variable.

- **Fit Statistics** — displays a table of the fit statistics for the model, broken down by data role.

**Model Interpretability**

- **Input Relative Importance** — displays a table of the relative importance of the input variables, including variable level and label.

- **Partial Dependence** — displays the functional relationship between the input variable and the model prediction.

- **LIME Plots** — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed around a particular observation to a create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when Explain Individual Predictions is selected.
  - **Local Instances** — displays the values for each input variable of the cluster center.
  - **Local Explanation** — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.
  - **Local Model** — displays the parameter estimates of the local linear model that is used to predict the cluster center.
  - **Prediction** — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

- **LIME Score Code** — displays the SAS code that Model Studio used to assign observations to clusters. This is displayed only when Create score code for assigning clusters is selected.

- **Individual Conditional Expectation** — displays the functional relationship between the input variable and the model prediction. The relationships for all cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.
Overview of Clustering

The **Clustering** node is a Data Mining Preprocessing node. Use the **Clustering** node to perform observation clustering based on distances that are computed from quantitative or qualitative variables (or both). The observations are divided into clusters such that every observation belongs to exactly one cluster. Clustering places observations into groups or clusters suggested by the data, such that observations within a cluster are similar and observations from different clusters are dissimilar.

This node uses the following algorithms:
- the $k$-means algorithm for clustering interval (quantitative) input variables
- the $k$-modes algorithm for clustering nominal (qualitative) input variables
- the $k$-prototypes algorithm for clustering mixed input that contains both interval and nominal variables

After clustering is performed, the characteristics of the clusters can be examined graphically using the Clustering Results. The consistency of clusters across variables is of particular interest. The multidimensional charts and plots enable you to graphically compare the clusters.

Note that the cluster identifier for each observation can be passed to subsequent nodes for use as an input, ID, or segment variable. The default is a segment variable.
Clustering Properties

General Properties

- **Cluster initialization** — specifies the method used to determine initial cluster membership. Here are the possible values:
  - Forgy — assigns $k$ data points as the $k$ initial clusters.
  - Random — randomly assigns each observation to a cluster.
- **Automatic gamma** — specifies whether to automatically compute the gamma coefficient in the mixed distance computation.
- **User specified gamma** — specifies the value for the coefficient gamma in the mixed distance computation.
- **Cluster variable role** — specifies the role that you want to assign to the cluster variable. By default, the cluster variable (segment identifier) is assigned a role of segment. Available roles are as follows:
  - ID
  - Input
  - Rejected
  - Segment
  The role of segment is useful for BY-group processing. Note that the segment identifier retains the selected variable role when it is passed to subsequent nodes in the pipeline.
- **Random seed** — specifies a positive integer to be used to start the pseudo-random number generator. The default value is 12345.

Interval Inputs

- **Missing interval inputs** — specifies the imputation method for interval input variables. Possible values are Exclude or Impute with mean. The default setting is Impute with mean.
- **Standardization method** — specifies the method for standardizing interval input variables. Here are the possible values:
  - None
  - Range
  - Z Score
  Specify Range to standardize on the range of the input variable, or specify Z Score to standardize on the calculated Z Score. Select None to disable standardization. The default setting is None.
- **Similarity distance** — specifies the distance measure for similarity measurement for interval input variables. Possible values are **Euclidean distance** and **Manhattan distance**. **Euclidean distance** calculates the line segment distance between two clusters. **Manhattan distance**, or taxicab distance, calculates the distance between two clusters using only axis-aligned movements. The default setting is **Euclidean distance**.

### Class Inputs

- **Missing class inputs** — specifies the imputation method for class input variables. Possible values are **Exclude** and **Impute with mode**. The default setting is **Exclude**.

- **Similarity distance** — specifies the distance measure for similarity measurement of class input variables. Here are the possible values:
  - Binary
  - Global frequency
  - Relative frequency

  **Binary** calculates a simple matching distance. **Global frequency** and **Relative frequency** calculate distances based on the frequency of class input variables in the input data table or in each cluster respectively. The default setting is **Binary**.

### Number of Clusters Estimation

- **Number of clusters method** — specifies the method used to estimate the number of clusters. Possible values are **Aligned box criterion** and **User specify**. Select **User specify** to enter the number of clusters in the **Number of clusters** field.

  Select the **Aligned box criterion** property to generate the number of clusters based on the following parameters:

  - **Number of reference data sets** — specifies the number of reference data sets to be created for each cluster candidate.

  - **Maximum number of clusters** — specifies the maximum number of clusters, with a default value of 6. If the number of observations is less than the number specified, then the number of clusters will be set to the number of observations. If the number of observations in a cluster is zero, then this cluster will not be displayed in the results.

  - **Minimum number of clusters** — specifies the minimum number of clusters. The default value is 2.

  - **Estimation criterion** — specifies the criterion to use to estimate the number of clusters that use the statistics obtained in the ABC method. Here are the possible values:
    - **All criteria** — specifies that the number of clusters is determined by a combination of the other three available property settings.
First peak value — specifies that the number of clusters is determined by the first peak among the peak values in gap statistics.

First peak with one-standard-error — specifies that the number of clusters is determined by the smallest $k$ such that the gap value for that $k$ is greater than the one-standard-error adjusted gap value for $k+1$.

Global peak value — specifies that the number of clusters is determined by the maximum value among all peak values in gap statistics.

The default value is Global peak value.

Alignment method — specifies the method for aligning the reference data set based on the input data. Possible values are None and PCA. If set to None, the node generates the reference data set from a uniform distribution over the range of values for each subset of the input data set. If set to PCA, the node generates the reference data set from a uniform distribution over a box aligned with the principal components of each subset of the input data set.

Stop Criterion

Stop method — specifies the stop criterion method. The possible values are Cluster change and Within cluster distance. Cluster change is the default.

Cluster change parameter — specifies the percentile of observations that do not change their cluster for the iteration. The range is between 0 and 100, and the default is 20. If you select Cluster change as the Stop method, the cluster calculation will conclude when the number of observations that change clusters is smaller than the chosen parameter value.

Within cluster distance parameter — specifies the difference of within-cluster distance change between iterations. If you select Within cluster distance for the Stop method, the cluster calculation will conclude if the cluster distance changes by less than the given parameter after an iteration.

Maximum number of iterations — specifies the maximum number of iterations for the algorithm to perform, with a default value of 10. In each iteration, each observation is assigned to the nearest cluster centroid, and the centroids are recomputed.

Clustering Results

After running the node, you can open the Results window by right-clicking the node and selecting Results from the pop-up menu.

Results

Segment Plot — displays clusters by segment number. The size of each pie slice reflects each segment’s percentage of all the clustered data. Each segment ID appears as you click a given pie slice.

ABC Statistics — displays the relationship between the number of clusters and the gap statistic using ABC. It also displays the selected number of clusters.
based on ABC. Therefore, it is available only when that method is chosen to estimate the number of clusters. Click on the graph to see the exact gap statistic values at each point.

- **Cluster Centroids** — displays the centroid coordinates for each of the clusters. Use the **Options** tab to select which coordinates appear in the table.

- **Node Score Code** — displays SAS score code that was created by the node. The SAS score code can be used outside the Model Studio environment to score new data.

- **Properties** — specifies the various properties selected before running the node.

- **Output** — displays the SAS output of the cluster run.
Overview of Data Exploration

You will frequently find it useful to profile a data set before continuing analysis. Much like PROC CONTENTS, the Data Exploration node enables you to profile a data set. The Data Exploration node is a Miscellaneous node that selects a subset of variables to provide a representative snapshot of the data. Variables can be selected to show most important inputs, or to indicate "suspicious variables"—that is, variables with anomalous statistics. The Data Exploration node can be placed anywhere in a pipeline, except after the Model Comparison node.

Data Exploration Properties

Variable Selection

- **Variable selection criterion** — specifies whether to display the most important inputs or suspicious variables. You control the selection of suspicious variables by specifying screening criteria. Possible values are Importance and Screening. The default value is Importance.

- **Maximum number of variables to select** — specifies the maximum number of important inputs to display. The default value is 50. This option is unavailable if the Variable selection criterion is Screening.

- **Screening Cutoffs** — specifies the screening criteria for suspicious variables. These options are available only if the Variable selection criterion is Screening.
- **Missing values** — specifies the cutoff for flagging variables with a high percentage of missing values. Specify a percentage. The default value is 25.

- **Class levels** — specifies the cutoff for flagging high cardinality class variables. Specify the number of class levels. The default value is 32.

- **Dominant mode** — specifies the cutoff for flagging class variables with dominant levels. Specify a percentage. The default value is 80.

- **Rare mode** — specifies the cutoff for flagging class variables with rare modes. Specify a percentage. The default value is 20.

- **Skewness** — specifies the cutoff for flagging skewed interval variables. Specify an absolute skewness value. (For example, a value of 1 flags variables with skewness greater than 1 or less than -1.) The default value is 1.

- **Peakiness** — specifies the cutoff for flagging peaky (leptokurtic) interval variables. Specify a positive kurtosis value. The default value is 2.

- **Flatness** — specifies the cutoff for flagging interval variables with thick tails (that is, platykurtic distributions). Specify a negative kurtosis value. The default value is -1.

### Interval Variables

- **Number of bins** — specifies the number of bins for histograms of interval variables. Possible values range from 4 to 32. The default value is 8.

- **Cluster plot** — specifies whether to display clustering of interval variable distributions. By default, this option is deselected.

- **Minimum number of interval variables** — specifies the minimum number of interval variables needed to produce a cluster plot. If there are fewer interval variables, the plot is not produced. This option is available only if Cluster plot is selected. The default value is 20.

- **t-SNE projection** — specifies whether to perform t-SNE projections of input variables. This option is deselected by default.

- **t-SNE perplexity** — specifies the t-SNE perplexity that controls the separation of points in the projected space. Possible values range from 1 to 100. The default value is 30.

### Data Exploration Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

- **Results**
  - **Important Inputs** — displays a bar chart of the input variables, ranked by importance. This bar chart is available only if the **Variable selection criterion** is **Importance**.

- **Class Variable Summaries** — displays a bar chart of the number of levels for each of the class variables. Hidden categories are displayed by blue bars, and non-hidden categories are displayed by orange bars. To examine the percent of values equal to the mode for each class variable, use the drop-down menu in the upper right corner.

- **Suspicious Class Variables** — displays a table of class variables that have been flagged for having suspicious characteristics. These characteristics include too many levels, rare mode, dominant mode, and too many missing values. This table is available only if the variable selection criterion is **Screening**.

- **Suspicious Interval Variables** — displays a table of interval variables that have been flagged for having suspicious characteristics. These characteristics include skewness, peakiness, flat (thick tails), and too many missing values. This table is available only if the variable selection criterion is **Screening**.

- **Class Variable Distributions** — displays a bar chart of the frequency percentage of each value for a given class variable. To switch the displayed class variable, use the drop-down menu in the upper right corner.

- **Interval Variable Moments** — displays a table of the interval variables with their associated statistics. These statistics include minimum, maximum, mean, standard deviation, skewness, kurtosis, relative variability, and the mean plus or minus two standard deviations.

- **Interval Variable Summaries** — displays a scatter plot that represents the deviation from normality (that is, kurtosis on the Y axis and skewness on the X axis) for each interval variable. To examine a bar chart of the relative variability for each interval variable, use the drop-down menu in the upper right corner.
  
  **Note:** Relative variability is useful for comparing variables with similar scales, such as several income variables.

- **Interval Variable Distributions** — displays a bar chart of the frequency percentage of each bin for a given interval variable. To switch the displayed interval variable, use the drop-down menu in the upper right corner.

- **Missing Values** — displays a bar chart of the percentage of missing values for each variable.

- **Target by Input Variable Crosstabulations** — displays a bar chart of the frequency percentage of each level of the target variable for a given input variable. To switch the displayed variable, use the drop-down menu in the upper right corner. This bar chart is available only if the target is a class variable.

- **Target Variable Means** — displays a bar chart of the mean value of the target variable for each level of the input variable. To switch the displayed variable, use the drop-down menu in the upper right corner. This bar chart is available only if the target is an interval variable.

- **t-SNE Projection of Interval Inputs** — displays a 3-D nonlinear projection of interval inputs using t-Distributed Stochastic Neighbor Embedding (t-SNE). The t-SNE plot can help identify naturally occurring clusters in the data.

- **Node Score Code** — displays the SAS score code that was created by the node. The SAS score code can be used outside of the Model Studio environment in custom user applications.

- **Properties** — specifies the various properties selected before running the node.

- **Output** — displays the SAS output of the data exploration run.
Overview of Decision Tree

The Decision Tree node is a Supervised Learning node. An empirical tree represents a segmentation of the data that is created by applying a series of simple rules. Each rule assigns an observation to a segment based on the value of one input. One rule is applied after another, resulting in a hierarchy of segments within segments. The hierarchy is called a tree, and each segment is called a node. The original segment contains the entire data set and is called the root node of the tree. A node with all its successors forms a branch of the node that created it. The final nodes are called leaves. For each leaf, a decision is made and applied to all observations in the leaf. The type of decision depends on the context. In predictive modeling, the decision is the predicted value.

You can use the Decision Tree node to create decision trees that do one of the following tasks:

- classify observations based on the values of nominal or interval inputs
- predict responses based on the values of nominal or interval inputs

Decision trees produce a set of rules that can be used to generate predictions for a new data set. This information can then be used to drive business decisions. For example, in database marketing, decision trees can be used to develop customer profiles that help marketers target promotional mailings in order to generate a higher response rate.
Decision Tree Properties

Splitting Options

- **Grow criterion** — specifies the criterion for splitting a parent node into child nodes. The following options are available:
  - **Class target criterion** — specifies the splitting criterion to use for determining best splits on inputs that are given a class target. Here are the possible values:
    - CHAID
    - Chi-square
    - Entropy
    - Gini
    - Information gain ratio
  The default value is Information gain ratio.
  - **Interval target criterion** — specifies the splitting criterion to use for determining the best splits on inputs that are given an interval target. Here are the possible values:
    - CHAID
    - F test
    - Variance
  The default value is Variance.
  - **Significance level** — specifies the significance level for the splitting criteria CHAID, Chi-Square, and F Test. The default value is 0.2.
  - **Bonferroni** — specifies whether to apply a Bonferroni adjustment to the top p-values for the splitting criteria CHAID, Chi-Square, and F Test. By default, this option is deselected.

- **Maximum number of branches** — specifies the maximum number of branches that a splitting rule produces. The default value of 2 results in binary splits. Possible values range from 2 to 10.

- **Maximum depth** — specifies the maximum number of generations in nodes. The original node, generation 0, is called the root node. The children of the root node are the first generation. Possible values range from 1 to 50. The default value is 10.

- **Minimum leaf size** — specifies the smallest number of training observations that a leaf can have. The default value is 5.

- **Missing values** — specifies how a splitting rule handles an observation with missing values. Here are the possible values:
  - Ignore
- Largest branch
- Most correlated branch
- Separate branch
- Use as machine smallest
- Use in search

Missing values are either used as a value during the split search or assigned to a node based on this selection. The default value is **Use in search**.

- **Minimum missing use in search** — specifies the minimum number of missing values needed in a splitting variable for missing to be treated a separate level. This property is available only when **Use in search** is specified for **Missing values**. The default value is 1.

- **Number of interval bins** — specifies the number of bins used for interval inputs. Bin size is \((\text{maximum value} - \text{minimum value})/\text{IntervalBins}\). The default value is 20.

- **Interval bin method** — specifies the method used to bin the interval input variables. Select **Bucket** to divide inputs into evenly spaced intervals based on the difference between maximum and minimum values. Select **Quantile** to divide inputs into approximately equal sized groups. The default value is **Bucket**.

- **Surrogate rules** — specifies the number of surrogate rules. The default value is 0.

- **Use input once** — specifies that no splitting rule will be based on an input variable that has already been used in a splitting rule of an ancestor node. By default, this is deselected.

- **Perform clustering-based split search** — specifies that a clustering-based search algorithm, instead of an exhaustive search, be used for determining the best split for each input for each tree node. When this option is selected, the order of bins is ignored for interval inputs. By default, this is deselected.

### Pruning Options

- **Subtree method** — specifies how to construct the subtree in terms of subtree methods. Possible values include the following:
  - **C4.5** — The pruning is done with a C4.5 algorithm.
  - **Cost complexity** — The subtree with a minimum leaf-penalized ASE is chosen.
  - **Reduced error** — The smallest subtree with the best assessment value is chosen.

  **C4.5** is available only for class targets. With **Reduced error** pruning, the assessment measure for class targets is misclassification rate, and the assessment measure for interval targets is ASE. The default value is **Cost complexity**.

- **Selection method** — specifies how to construct the subtree in terms of selection methods. Possible values include the following:
  - **Automatic** — specifies the appropriate subtree for the specified subtree pruning method.
- **Cost complexity alpha** — specifies the subtree by cost-complexity. The default value is 0.
- **Largest** — specifies the full tree.
- **N** — specifies the largest subtree with at most \( N \) leaves.
  The default value is **Automatic**.

  - **Number of leaves** — specifies the number of leaves that are used in creating the subtree when the subtree selection method is set to \( N \). The default value is 1.
  - **Confidence** — specifies the binomial distribution confidence level to use to determine the error rates of merged and split nodes. The default value is 0.25. This option is available only when C4.5 is the pruning method.
  - **Cross validation folds** — specifies the number of cross validation folds to use for cost-complexity pruning when there is no validation data. Possible values range from 2 to 20. The default value is 10.
  - **1–SE rule** — specifies whether to perform the one standard error rule when performing cross-validated cost complexity pruning. By default, this is deselected.

**Seed Property**

- **Seed** — specifies the random number seed used for cross validation cost complexity or for autotuning. The default value is 12345.

**Tree Diagram Options**

- **Display embedded bar charts in tree diagram** — specifies whether to display an embedded bar chart of the target levels in each node of a tree diagram for a class target. By default, this is selected.
- **Class target node color** — specifies the value by which to color the nodes in the tree diagram and the treemap for a class target. Here are the possible values:
  - **Probability of event**
  - **Proportion correctly classified**
  - **Single color**
  The default value is **Probability of event**.
- **Interval target node color** — specifies the value by which to color the nodes in the tree diagram and the treemap for an interval target. Possible values are **Average** and **Single color**. The default value is **Average**.
Perform Autotuning

This property selects whether to perform autotuning of any decision tree parameters. Warning: Performing autotuning can substantially increase run time. If Perform Autotuning is selected, the following options are available:

- **Maximum Depth** — specifies whether to autotune the maximum depth parameter. If selected, the following options are available:
  - **Initial value** — specifies the initial value for autotuning maximum depth. The default value is 10. Use the From and To options to specify the range. The default From value is 1, and the default To value is 19.

- **Interval input bins** — specifies whether to autotune the number of interval input bins. If selected, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the number of interval input bins. The default value is 20. Use the From and To options to specify the range. The default From value is 20, and the default To value is 200.

- **Grow Criterion** — specifies whether to autotune the grow criterion. If selected, the following options are available:
  - **Class target** — specifies the list of values for autotuning the grow criterion for a class target. Select a subset of the following values:
    - Entropy
    - CHAID
    - Information gain ratio
    - Gini
    - Chi-square
  - **Interval target** — specifies the list of values for autotuning the grow criterion for an interval target. Select a subset of the following values:
    - Variance
    - F test
    - CHAID

- **Search Options** — specifies the options for autotuning searching. The following options are available:
  - **Search method** — specifies the tuning search method. Here are the possible values:
    - **Bayesian** — specifies the Bayesian method. This method uses priors to seed the iterative optimization.
    - **Genetic algorithm** — specifies the genetic algorithm method. This method uses an initial Latin Hypercube sample that seeds a genetic algorithm. The genetic algorithm generates a new population of alternative configurations at each iteration.
    - **Grid** — specifies the grid method. This method performs an exhaustive search of all configurations. Each hyperparameter of interest is discretized into a desired set of values to be studied. Models are trained and
assessed for all combinations of values across all the hyperparameters (thus forming a multi-dimensional “grid”).

- **Latin hypercube sample** — specifies the Latin Hypercube method. This method performs an optimized grid search that is uniform in each tuning parameter, but random in combinations.

- **Random** — specifies the Random method. This method generates a single sample of purely random configurations.

The default value is **Genetic algorithm**.

- **Number of evaluations per iteration** — specifies the number of tuning evaluations in one iteration. This option is available only if the **Search method** is **Genetic algorithm** or **Bayesian**. The default value is 10.

- **Maximum number of evaluations** — specifies the maximum number of tuning evaluations. This option is available only if the **Search method** is **Genetic algorithm** or **Bayesian**. The default value is 50.

- **Maximum number of iterations** — specifies the maximum number of tuning iterations. This option is available only if the **Search method** is **Genetic algorithm** or **Bayesian**. The default value is 5.

- **Maximum number of points in model** — specifies the maximum number of points in the model. This option is available only if the **Search method** is **Bayesian**. The default value is 100.

- **Sample size** — specifies the sample size. This option is available only if the **Search method** is **Random** or **Latin hypercube sample**. The default value is 50.

**General Options** — specifies the general properties for autotuning. The following options are available:

- **Validation method** — specifies the validation method for finding the objective value. If your data is partitioned, then that partition is used. **Validation method**, **Validation data proportion**, and **Cross validation number of folds** are all ignored. Here are the possible values:
  - **Partition** — specifies using the partition validation method. With partition, you specify proportions to use for randomly assigning observations to each role.
  - **K-fold cross validation** — specifies using the cross validation method. In cross validation, each model evaluation requires \( k \) training executions (on \( k-1 \) data folds) and \( k \) scoring executions (on one holdout fold). This increases the evaluation time by approximately a factor of \( k \).

For small to medium data tables, cross validation provides, on average, a better representation of error across the whole data table. **Partition** is the default value.

- **Training data proportion** — specifies the proportion of data to be used for training the **Partition** validation method. The default value is 0.7.

- **Validation data proportion** — specifies the proportion of data to be used for the **Partition** validation method. The default value is 0.3.

- **Cross validation number of folds** — specifies the number of partition folds in the cross validation process (the \( k \) defined above). Possible values range from 2 to 20. The default value is 5.
Class target objective function — specifies the objective function to optimize for tuning parameters for a nominal target. Here are the possible values:

- Area under the curve
- Average squared error
- F0.5 score
- F1 score
- Gamma
- Gini coefficient
- Kolmogorov-Smirnov statistic
- Misclassification rate
- Multi-class log loss
- Root average squared error
- Tau

The default value is Misclassification rate.

Interval target objective function — specifies the objective function to optimize for tuning parameters for an interval target. Here are the possible values:

- Average squared error
- Mean absolute error
- Mean squared logarithmic error
- Root average squared error
- Root mean absolute error
- Root mean squared logarithmic error

The default value is Average squared error.

Maximum time (minutes) — specifies the maximum time (in minutes) for the optimization tuner. The default value is 60.

Maximum training time for single model (minutes) — specifies the maximum time (in minutes) for training a single model.

Binary Classification Cutoff

Specify node binary classification cutoff — specifies whether to use the binary classification cutoff specified below for the node. If deselected, the project binary classification cutoff is used for determining the predicted value for a binary target based on the posterior probabilities. By default, this value is deselected.

Node binary classification cutoff — specifies the cutoff to use in the node for determining the predicted value for a binary target based on the posterior probabilities. This option is available only if you select Specify node binary classification cutoff. The default value is 0.5.
Model Interpretability

Specifies the method of model interpretation in order to show how the model responds to changing inputs.

- **Explain Model Inputs** — specifies the following parameters for the LIME algorithm:
  - **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.
  - **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.
  - **Cluster-based ICE plots** — the maximum number of variables (1-10).

- **Explain Individual Predictions** — specifies whether to include **Number of clusters**, **Number of perturbed instances**, and **Number of iterations**. You can specify the **Distance metric** as either **Cosine** or **K prototype**. You can specify the **Kernel width** and the **Number of inputs to use in the explanation**. You can specify whether to **Create a score code for assigning clusters**.

Decision Tree Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Node**

- **Tree Diagram** — displays the decision tree. The line width of the tree is proportionally given by the ratio of the number of observations in the branch to the number of observations in the root node. When you move your mouse pointer over a node of a tree, a text box displays information about the node. If **Display embedded bar charts in tree diagram** was selected, a bar chart will be displayed. This bar contains the counts of each target level in each node. In addition, you can color the nodes by probability of event or proportion correctly classified. For an interval target, you can color the nodes by the target average.

- **Treemap** — displays a compact graphical display of the tree, which is similar to the tree in the **Tree Diagram** window. The rectangular regions that represent the tree nodes are colored by probability of event or proportion correctly classified for a class target or the average of an interval target. The node width is proportional to the number of observations in the node. Selecting a node in the **Treemap** window displays information about the node.

- **Pruning Error Plot** — displays a graph of the misclassification rate for a class target or ASE for an interval target. These values are given as a function of the number of leaves in a subtree for each data partition when pruning data exists. A reference line is drawn to indicate which subtree was selected as the final model.
Cross Validation Cost-Complexity — displays a plot with data on cross validation cost-complexity. This plot is displayed only when cross validation cost-complexity pruning is performed. For a class target, the plot displays the misclassification rate. For an interval target, this plot displays ASE as a function of the number of leaves in a subtree. Error bars are also given, indicating the average error rate plus or minus one standard error. A vertical reference line is drawn to indicate which subtree was selected as the final model. A horizontal reference line is drawn to represent the “1-SE Rule” even if it is not used for subtree selection.

Variable Importance — displays a table of importance data for each variable. These statistics include train importance, importance standard deviation, and relative importance.

Autotune Best Configuration — displays a table of the resulting values of parameters that were autotuned. This table will not be displayed if autotuning is not selected.

Autotune Results — displays the iterative results of the autotuning process, showing how each of the parameter values evolve as the model iterates. This table will not be displayed if autotuning is not selected.

Node Score Code — displays the SAS score code that was created by the node. The SAS score code can be used outside of the Model Studio environment in custom user applications.

Score Inputs — displays a table of data on the input variables for scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.

Score Outputs — displays a table of data on the predicted response variable for scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.

Path Score Code — displays the SAS score code that was created by the node, as well as all preceding nodes. The SAS model score code can be used outside of the Model Studio environment in custom user applications.

Train Code — displays the SAS code that Model Studio used to train the node.

Properties — specifies the various properties selected before running the node.

Output — displays the SAS output of the decision tree run.

Assessment

Predicted Reports — displays the predicted and target mean as a function of the depth for the model. The predicted and target mean are given for each of the data roles. To examine the predicted mean as a function of the target mean, use the drop-down menu in the upper right corner. This result is displayed only if the target is an interval variable.

Lift Reports — displays the cumulative lift as a function of the depth for the model. The cumulative lift is given for each of the data roles. To examine other statistics as a function of depth, use the drop-down menu in the upper right corner. Other statistics include lift, gain, % captured response, cumulative % captured response, % response, and cumulative % response. This result is displayed only if the target is a class variable.

ROC Reports — displays the ROC (receiver operating characteristic) chart for a model, giving the sensitivity as a function of 1-specificity. The sensitivity is given for each of the data roles. To examine other statistics, use the drop-down menu
in the upper right corner. Other statistics include accuracy and F1 score. This result is displayed only if the target is a class variable.

- **Fit Statistics** — displays a table of the fit statistics for the model, broken down by data role.

**Model Interpretability**

- **Input Relative Importance** — displays a table of the relative importance of the input variables, including variable level and label.

- **Partial Dependence** — displays the functional relationship between the input variable and the model prediction.

- **LIME Plots** — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed around a particular observation to create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when **Explain Individual Predictions** is selected.
  - **Local Instances** — displays the values for each input variable of the cluster center.
  - **Local Explanation** — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.
  - **Local Model** — displays the parameter estimates of the local linear model that is used to predict the cluster center.
  - **Prediction** — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

- **LIME Score Code** — displays the SAS code that Model Studio used to assign observations to clusters. This is only displayed when **Create score code for assigning clusters** is selected.

- **Individual Conditional Expectation** — displays the functional relationship between the input variable and the model prediction. The relationships for all cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.
Overview of Ensemble

The Ensemble node is a Postprocessing node. The Ensemble node creates new models by combining the posterior probabilities (for class targets) or the predicted values (for interval targets) from multiple predecessor models. The new model is then used to score the project and assess the new model.

One common ensemble approach is to use multiple modeling methods, such as a neural network and a decision tree, to obtain separate models from the same training data set. The component models from the two complementary modeling methods are integrated by the Ensemble node to form the final model solution.

Note:

The ensemble model can be more accurate than the individual models only if the individual models disagree with one another. You should always compare the model performance of the ensemble model with the individual models. You can compare models in a Model Comparison node. One useful feature of this component is the ability to assemble complex models such as forest or gradient boosting models.
Ensemble Properties

Interval Target

- **Predicted values** — specifies the function to combine models for interval targets. Possible values are Average and Maximum. The default value is Average.

Class Target

- **Posterior probabilities** — specifies the function to combine models for class targets. Here are the possible values:
  - Average
  - Geometric Mean
  - Maximum
  The default value is Average.

Binary Classification Cutoff

- **Specify node binary classification cutoff** — specifies whether to use the binary classification cutoff specified below for the node. If deselected, the project binary classification cutoff is used for determining the predicted value for a binary target based on the posterior probabilities. By default, this value is deselected.

- **Node binary classification cutoff** — specifies the cutoff to use in the node for determining the predicted value for a binary target based on the posterior probabilities. This option is available only if Specify node binary classification cutoff is selected. The default value is 0.5.

Model Interpretability

Specifies the method of model interpretation in order to show how the model responds to changing inputs.

- **Explain Model Inputs** — specifies the following parameters for the LIME algorithm:
- **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.

- **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.

- **Cluster-based ICE plots** — the maximum number of variables (1-10).

- **Explain Individual Predictions** — specifies whether to include **Number of clusters**, **Number of perturbed instances**, and **Number of iterations**. You can specify the **Distance metric** as either **Cosine** or **K prototype**. You can specify the **Kernel width** and the **Number of inputs to use in the explanation**. You can specify whether to **Create a score code for assigning clusters**.

---

### Ensemble Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

#### Node

- **Path EP Score Code** — displays the SAS score code that was created by the node. The EP Score code will be available only if an analytic store model is ensembled.

- **DS2 Package Code** — displays the SAS DS2 package code that was created by the node. The score code can be used outside of the Model Studio environment to score new data.

- **Path Score Code** — displays the SAS score code that was created by the node, as well as all preceding nodes. The SAS model score code can be used outside of the Model Studio environment in custom user applications.

- **Score Inputs** — displays a table of data on the input variables for scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.

- **Score Outputs** — displays a table of data on the predicted response variable for scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.

- **Properties** — specifies the various properties selected before running the node.

- **Models** — displays a table of the statistical models that were combined by the **Ensemble** node.

#### Assessment

- **Predicted Reports** — displays the predicted and target mean as a function of the depth for the model. The predicted and target mean are given for each of the data roles. To examine the predicted mean as a function of the target mean, use the drop-down menu in the upper right corner. This result is displayed only if the target is an interval variable.

- **Lift Reports** — displays the cumulative lift as a function of the depth for the model. The cumulative lift is given for each of the data roles. To examine other
statistics as a function of depth, use the drop-down menu in the upper right corner. Other statistics include lift, gain, % captured response, cumulative % captured response, % response, and cumulative % response. This result is displayed only if the target is a class variable.

- **ROC Reports** — displays the ROC (receiver operating characteristic) chart for a model, giving the sensitivity as a function of 1-specificity. The sensitivity is given for each of the data roles. To examine other statistics, use the drop-down menu in the upper right corner. Other statistics include accuracy and F1 score. This result is displayed only if the target is a class variable.

- **Fit Statistics** — displays a table of the fit statistics for the model, broken down by data role.

**Model Interpretability**

- **Input Relative Importance** — displays a table of the relative importance of the input variables, including variable level and label.

- **Partial Dependence** — displays the functional relationship between the input variable and the model prediction.

- **LIME Plots** — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed around a particular observation to a create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when **Explain Individual Predictions** is selected.
  - **Local Instances** — displays the values for each input variable of the cluster center.
  - **Local Explanation** — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.
  - **Local Model** — displays the parameter estimates of the local linear model that is used to predict the cluster center.
  - **Prediction** — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

- **LIME Score Code** — displays the SAS code that Model Studio used to assign observations to clusters. This is displayed only when **Create score code for assigning clusters** is selected.

- **Individual Conditional Expectation** — displays the functional relationship between the input variable and the model prediction. The relationships for all cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.
Overview of Feature Extraction

The Feature Extraction node is a Data Mining Preprocessing node. Use the Feature Extraction node to create new features from the initial set of data. These features encapsulate the central properties of a data set and represent it in a low dimensional space. The initial data set of raw features might be too large and unwieldy to be effectively managed, requiring an unreasonable amount of computing resources. Alternatively, the data set might be too robust, causing a classification algorithm to overfit, and providing poor extrapolation in the event of new observations. In either case, the Feature Extraction can be used to provide a more manageable, representative subset of input variables.

The Feature Extraction node offers four methods. You can either specify the method that you want to use or let the node automatically select the method based on the number of inputs in your data set. The available methods include the following:

- Principal Component Analysis (PCA)
- Singular Value Decomposition (SVD)
- Robust Principal Component Analysis
- Autoencoder

The automatic selection uses PCA when the number of interval inputs is less than or equal to 500. Otherwise, SVD with randomized optimization is used. Only the training partition is used for extracting the new features for the PCA and SVD methods. For the autoencoder method, early stopping is available. Early stopping can be performed based on the validation partition that already exists in your data or on a validation partition created from the training data.
Note: The PCA, SVD, and robust PCA methods are available only for interval inputs. The autoencoder is available for both class and interval inputs.

Feature Extraction Properties

Input Variables Property

- **Feature extraction method** — specifies the method used for feature extraction. Here are the possible values:
  - Autoencoder
  - Automatic
  - Principal component analysis (PCA)
  - Robust PCA (RPCA)
  - Singular value decomposition (SVD)

  The choice of feature extraction method will determine the subsequent available options. If Automatic is selected, PCA will be used if the number of input variables is less than or equal to 500. Otherwise, SVD with Randomized optimization will be used. Note that PCA, SVD, and RPCA use interval inputs only.

- **Reject original input variables** — specifies whether the role of the original input variables should be set to Rejected. Note that for PCA, SVD, and RPCA, only interval inputs are rejected.

PCA Options

- **Eigenvalue source** — specifies the source matrix to calculate eigenvalues and eigenvectors. Possible values are Correlation, using the correlation matrix, and Covariance, using the covariance matrix.

- **Component prefix** — specifies the principal component prefix, which is a valid SAS variable name. The default prefix string is PC. Length is limited to 25 characters.

- **Apply fixed number** — specifies whether to use a fixed number of principal components to be passed to successor nodes. If this option is selected, the Fixed number of principal components can range from 1 to 50.

- **Apply maximum number** — specifies whether to set a maximum limit for the number of principal components to be passed to successor nodes. If this option is selected, the Maximum number can range from 1 to 50.

- **Cumulative variance cutoff** — specifies the cumulative proportional variance cutoff value. The last principal components with cumulative proportional variance greater than this cutoff value are not passed to successor nodes. The default value is 0.99.
- **Minimum variance increment** — specifies the minimum increment for proportional variance. Principal components with a proportional increment less than this cutoff value are not passed to successor nodes.

### RPCA Options

- **Penalty weight** — specifies the weight to place on the penalty that is applied to sparse matrices in Augmented LaGrange Multiplier (ALM) optimization. A lower value places emphasis on sparsity.

- **Optimization method for RPCA** — specifies the optimization method for RPCA. Possible values include the following:
  - **Accelerated proximal gradient** — specifies the APG method, which is a relaxation of the PCP with an unconstrained optimization formulation.
  - **Augmented lagrange multiplier** — specifies the ALM method. This method is a relaxation of the principal component pursuit, or PCP, as a nonlinear unconstrained optimization problem.

  The APG algorithm is generally slower than the ALM algorithm. The APG algorithm is suggested when the observations are significantly corrupted by noise.

- **Maximum iterations for RPCA** — specifies the maximum number of iterations to solve the RPCA. The default is 1000.

- **Tolerance for RPCA** — specifies the tolerance criteria to solve RPCA—that is, the desired accuracy of the recovered solution. The default is 0.0000001.

### Additional RPCA and SVD Options

- **Input standardization** — specifies whether to center, scale, or standardize (center and scale) input variables using the traditional standardization method. The possible values are as follows:
  - **Center**
  - **None**
  - **Scale**
  - **Z score**

- **Maximum rank** — specifies the maximum number of latent variables that are passed to successor nodes. This number will be set to the number of input variables if it exceeds that value. The default value is 100.

- **Component prefix** — specifies the prefix for output variables, which should be a valid SAS variable name. Length is limited to 25 characters. The default prefix string is COMP.

- **Optimization method for SVD** — specifies the optimization method used to solve the SVD problem. The possible values are as follows:
  - **Automatic**
  - **Eigen**
Iterative

Randomized

The default Automatic selection will use Eigen optimization if the number of input variables is less than or equal to 500. Otherwise, Randomized optimization is used. Randomized optimization is recommended for wide data because it is a faster approximation.

Autoencoder Options

- Input standardization — specifies the method that is used to standardize the interval input variables. The possible values are as follows:
  - None
  - Range
  - Z Score

Range is the default.

- Use missing as level — specifies whether missing values should be treated as a level for class inputs.

- Feature prefix — specifies the prefix for output variables, which should be a valid SAS variable name. Length is limited to 25 characters. The default prefix string is Hidden.

- Hidden layer options — specifies the configurations for the hidden layer stage of the network.

  - Number of hidden layers — specifies the number of hidden layers to include in the neural network. The possible values are 3 and 5. The default value is 3.

  - Middle hidden layer: number of neurons (features) — specifies the number of hidden neurons in the middle hidden layer for an autoencoder. For a network with three hidden layers, this number is the second hidden layer. For a network with five hidden layers, this is the third hidden layer. This is the number of features to use to represent the inputs. The default value is 10.

  - First and last hidden layer: number of neurons — specifies the number of hidden neurons in the first and last hidden layer for an autoencoder. The default value is 100.

  - Second and fourth hidden layer: number of neurons — specifies the number of hidden neurons in the second and fourth hidden layer for an autoencoder when Number of hidden layers is 5. The default value is 50.

- Hidden layer activation function — specifies the activation function to use in the hidden layers. The possible function choices include the typical smooth functions that you encounter in neural networks. If Varies is selected, you individually select the activation function for each hidden layer. The default value is Tanh.

- Common Optimization Options

  - Number of tries — specifies the number of times to train the network. This is done by using different initial estimates for the weights. Possible values range from 1 to 64. The default value is 1.
- **Maximum iterations** — specifies the maximum number of iterations allowed within each try. The default value is 300.

- **Stagnation for early stopping** — specifies the number of successive iterations without improvement in the validation error before stopping the optimization early. The value 0 indicates that early stopping is not performed. The default value is 5.

- **Random seed** — specifies the random seed to use for generating random numbers to initialize the network weights. The default value is 12345.

- **L1 weight decay** — specifies the weight decay for L1 regularization. The default value is 0.

- **L2 weight decay** — specifies the weight decay for L2 regularization. The default value is 0.1.

- **SGD Optimization Options** — specifies the configurations for the stochastic gradient descent.

  - **Learning rate** — specifies the learning rate parameter for SGD optimization. The default value is 0.001.

  - **Annealing rate** — specifies the annealing rate parameter for SGD optimization. The default value is 0.000001.

  - **Input layer dropout ratio** — specifies the dropout ratio for the input layer when SGD optimization is used. The default rate is 0.

  - **Hidden layer dropout ratio** — specifies the dropout ratio for the hidden layer when SGD optimization is used. The default rate is 0.

  - **Minibatch size** — specifies the size of the minibatches used for SGD optimization. The default value is 10.

  - **Momentum** — specifies the value for momentum for SGD optimization. The default value is 0.

  - **Create deterministic results** — specifies whether to create deterministic (reproducible) results using the specified SGD seed.

    Note: Selecting this property can significantly increase run time.

    If **Create deterministic results** is selected, the **SGD seed** can be set. The default value is 12345.

- **Create validation** — specifies whether a validation sample should be created from the incoming training data. If **Create validation** is selected, the following configurations are available:

  - **Validation proportion** — specifies the probability of a given record being selected for the validation partition. The default value is 0.3.

  - **Partition seed** — specifies the partition seed to generate the sample that will be used for validation. The default value is 12345.

---

**Feature Extraction Results**

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.
Note: The results available for this node will vary based on the feature extraction method.

Results

- **Principal Components Coefficient** — displays the coefficient values for each principal component. Switch the selected principal component using the drop-down menu in the upper right corner. This display is available only if the feature extraction method is Principal component analysis, Robust PCA, or Singular value decomposition.

- **Eigenvalue Plots** — displays the eigenvalue that corresponds to each principal component via a line graph. The vertical line that corresponds to the number of Principal Components selected is in bold. Switch the display between Eigenvalue, Proportional Eigenvalue, Cumulative Proportional Eigenvalue, and Log Eigenvalue using the drop-down menu in the upper right corner. This display is available only if the feature extraction method is Principal component analysis, Robust PCA, or Singular value decomposition.

- **2D Representation of the Data Points Projections** — displays a scatter plot relating two principal components, with the target variable that determines the color of the point. Switch the principal component pair being displayed using the drop-down menu in the upper right corner. This display is available only if the feature extraction method is Principal component analysis.

- **Iteration Plot** — displays a graph that measures the Valid Error after a given number of iterations. Switch the display between Valid Error, Objective, and Loss using the drop-down menu in the upper right corner. This graph is available only if the feature extraction method is Autoencoder.

- **2D Representation of Hidden Neurons** — displays a scatter plot that relates two hidden neurons in the middle layer, with the target variable that determines the color of the point. Switch the hidden neuron pair being displayed using the drop-down menu in the upper right corner. This plot is available only if the feature extraction method is Autoencoder and Middle hidden layer: number of neurons (features) is less than 6.

- **Node Score Code** — displays SAS score code that was created by the node. The SAS score code can be used outside the Model Studio environment to score new data.

- **Properties** — specifies the various properties that are selected before running the node.

- **Output** — displays the SAS output of the feature extraction run. The output given will vary based on the selected feature extraction method.
Overview of Filtering

The Filtering node is a Data Mining Preprocessing node. Based on data values in the training data set, it is used to filter the training, validation, and test data sets. You can use filters to exclude certain observations, such as extreme outliers and errant data that you do not want to include in your mining analysis. Filtering extreme values from the training data tends to produce better models because the parameter estimates are more stable. The Filtering node ignores target and rejected variables.

Filtering Properties

Class Variable Filtering

- **Class filtering method** — specifies the method for filtering class variables. Levels that do not exceed the rare value cutoff are excluded. Here are the possible values:
  - None
  - Rare values (count)
  - Rare values (percentage)
- **Frequency cutoff** — specifies the maximum frequency for excluding rare values. The default value is 1.
- **Percentage cutoff** — specifies the maximum percentage for excluding rare values. The default value is 1.

- **Maximum number of levels cutoff** — specifies the maximum number of class levels cutoff. Only Class variables that have fewer levels than the cutoff value are considered for filtering. The default value is 25.

- If **None** is selected for the class filtering method, there is no cutoff available.

- **Keep missing class values** — specifies whether to keep missing levels for class variables.

---

### Interval Variable Filtering

- **Interval filtering limits method** — specifies the default method for calculating interval variable filtering limits. Here are the possible values:
  - Extreme percentiles
  - Median absolute deviation (MAD)
  - Metadata limits
  - None
  - Standard deviation from the mean

- **Cutoff for standard deviations** — specifies the number of standard deviations from the mean to be used as a cutoff value. That is, values that exceed the specified number of standard deviations away from the mean will be filtered out. The default value is 3.

- **Cutoff for median absolute deviations** — specifies the number of deviations from the median to be used as a cutoff value. That is, values that exceed the specified number of absolute deviations away from the median will be filtered out. The default value is 9.

- **Cutoff for extreme percentiles** — specifies the cutoff percentile for extreme percentiles. The default value is 0.5.

- **Alternate interval filtering limits method** — specifies the method for calculating interval variable filtering limits where metadata lacks stored limits. If the interval filtering limits method is **Metadata limits**, any limits stored in metadata are used for filtering.

  Here are the possible values:
  - Extreme percentiles
  - Median absolute deviation (MAD)
  - None
  - Standard deviation from the mean

- **Cutoff for standard deviations** — specifies the maximum number of standard deviations from the mean that an observation is allowed to be before it is filtered.

- **Keep missing interval values** — specifies whether to keep missing values for interval variables.
Filter Indicator Property

- **Filter indicator usage** — specifies how the filter indicator is used. The filter indicator is a flag that indicates whether a row of data contains an outlier or rare value. The possible values are **Filter** and **Model input**. For **Model input**, the filter indicator is used as an input in the modeling process. For **Filter**, the filter indicator is applied to the training, validation, and test data to exclude the outlier and rare value rows.

*Note:* Rows that contain missing values are also flagged in the filtering indicator, as required by the **Keep missing interval values** and **Keep missing class values** properties.

Filtering Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Results**

- **Excluded Class Values** — displays a table of all the class variable levels that were filtered from the training data set. The table provides data for variable label, variable role, level, train count, train percent, and filter method.

- **Limits for Interval Variables** — displays a table that shows the upper and lower limits that were established by the filter method for interval variables. The table provides data for variable label, variable role, limits method, lower limit, upper limit, and keep missing values.

- **Number of Observations** — displays the number of observations for the training data, noting the original amount, the number that were excluded, and the number that were included.

- **Node Score Code** — displays the SAS score code that was created by the node. The SAS score code can be used outside the Model Studio environment in custom user applications.

- **Properties** — specifies the various properties that were selected before running the node.

- **Output** — displays the SAS output of the filtering run.
Overview of Forest

The Forest node is a Supervised Learning node that creates a predictive model called a forest. A forest consists of several decision trees that differ from each other in two ways. First, the training data for a tree is a sample with replacement from all available observations. Second, the input variables that are considered for splitting a node are randomly selected from all available inputs. In other respects, trees in a forest are trained like standard trees.

The training data for an individual tree excludes some of the available data. The data that is withheld from training is called the out-of-bag sample. The out-of-bag sample is used to assess the fit of the model.

The Forest node accepts interval and class target variables. For an interval target, the prediction in a leaf of an individual tree equals the average of the target values among the bagged training observations in that leaf. For a class target, the posterior probability of a target category equals the proportion of that category among the bagged training observations in that leaf. Predictions or posterior probabilities are then averaged across all the trees in the forest. Averaging over trees with different training samples reduces the dependence of the predictions on a particular training sample. Increasing the number of trees does not increase the risk of overfitting the data and can decrease it. However, if the predictions from different trees are correlated, then increasing the number of trees makes little or no improvement.
Forest Properties

Number of Trees Property

- **Number of trees** — specifies the number of trees in the forest. The default value is 100.

- **Class target voting method** — specifies the method to use for calculating the predicted probability for a class target. Here are the possible values:
  - **Probability** — uses the average target level probability across all trees.
  - **Majority** — uses the proportion of trees in the forest that predicted the level as target.

  The default value is **Probability**.

Tree-Splitting Options

- **Class target criterion** — specifies the splitting criterion to use for determining the best splits on inputs given a class target. Here are the possible values:
  - **CHAID**
  - **Chi-square**
  - **Entropy**
  - **Gini**
  - **Information gain ratio**

  The default value is **Information gain ratio**.

- **Interval target criterion** — specifies the splitting criterion to use for determining the best splits on inputs given an interval target. Here are the possible values:
  - **CHAID**
  - **F test**
  - **Variance**

  The default value is **Variance**.

- **Maximum number of branches** — specifies the maximum number of branches to consider for a teach node split in the tree. Possible values range from 2 to 5. The default value is 2.

- **Maximum depth** — specifies the maximum depth for each generated tree within the forest. Possible values range from 1 to 50. The default value is 20.

- **Minimum leaf size** — specifies the smallest number of training observations that a new branch can have. That number is expressed as the count of the
available observations. Possible values range from 1 to 64. The default value is 5.

- **Missing values** — specifies the action to take when there are values missing. You can specify to **Ignore**, **Use as machine smallest**, or **Use in search**. The default value is **Use in search**.

- **Minimum missing use in search** — specifies a threshold for using missing values in the split search. The default value is 1.

- **Number of interval bins** — specifies the number of bins that are used for interval inputs. Bin size is (maximum value - minimum value)/IntervalBins. The default value is 20.

- **Interval bin method** — specifies the method used to bin the interval input variables. Select **Bucket** to divide inputs into evenly spaced intervals based on the difference between maximum and minimum values. Select **Quantile** to divide inputs into approximately equal sized groups. The default value is **Bucket**.

- **In-bag sample proportion** — specifies the proportion of training observations to train a tree with—that is, the “in-bag” proportion. The default value is 0.6.

- **Use default number of inputs to consider per split** — specifies whether to use the default number of inputs to consider per split—that is, the square root of the number of inputs. By default, this option is selected.

- **Number of inputs to consider per split** — specifies the number of input variables that are randomly sampled to use per split. This option is available only if you deselect **Use default number of inputs to consider per split**. The default value is 100.

- **Number of inputs to subset with LOH** — specifies the number of input variables to further sample using the LOH method. When set to 0, no further sampling of inputs is performed. The default value is 0.

- **Seed** — specifies the seed for generating random numbers. This option is used to select training observations for each tree and to select candidate variables in each node to split on. The default value is 12345.

## Perform Autotuning

This feature specifies whether to perform autotuning of any forest parameters. Warning: Performing autotuning can substantially increase run time. If **Perform Autotuning** is selected, the following options are available:

- **Maximum Depth** — specifies whether to autotune the maximum depth parameter. If you select this option, the following options are available:
  - **Initial value** — specifies the initial value for autotuning maximum depth. Possible values range from 1 to 50. The default value is 20. Use the **From** and **To** options to specify the range. The default **From** value is 1, and the default **To** value is 29.

- **Number of Trees** — specifies whether to autotune the tree number parameter. If you select this option, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the number of trees. The default value is 100. Use the **From** and **To** options to specify the range. The default **From** value is 20, and the default **To** value is 150.
- **In-bag Sample Proportion** — specifies whether to autotune the in-bag sample proportion parameter. If this option is selected, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the in-bag sample proportions. The default value is 0.6. Use the **From** and **To** options to specify the range. The default **From** value is 0.1, and the default **To** value is 0.9.

- **Number of Inputs per Split** — specifies whether to autotune the number of inputs per split parameter. If this option is selected, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the number of inputs per split. The default value is 100. Use the **From** and **To** options to specify the range. The default **From** value is 1, and the default **To** value is 100.

- **Search Options** — specifies the autotune search method and properties. The following options are available:
  - **Search method** — specifies the tuning search method. Here are the possible values:
    - **Bayesian** — specifies the Bayesian method. This method uses priors to seed the iterative optimization.
    - **Genetic algorithm** — specifies the genetic algorithm method. This method uses an initial Latin Hypercube sample that seeds a genetic algorithm to generate a new population of alternative configurations at each iteration.
    - **Grid** — specifies the grid method. This method performs an exhaustive search of all configurations. Each hyperparameter of interest is discretized into a desired set of values to be studied. Models are trained and assessed for all combinations of values across all the hyperparameters (thus forming a multi-dimensional “grid”).
    - **Latin hypercube sample** — specifies the Latin Hypercube method. This method performs an optimized grid search that is uniform in each tuning parameter, but random in combinations.
    - **Random** — specifies the Random method. This method generates a single sample of purely random configurations.

  The default value is **Genetic algorithm**.

  - **Number of evaluations per iteration** — specifies the number of tuning evaluations in one iteration. This option is available only if the **Search method** is **Genetic algorithm** or **Bayesian**. The default value is 10.

  - **Maximum number of evaluations** — specifies the maximum number of tuning evaluations. This option is available only if the **Search method** is **Genetic algorithm** or **Bayesian**. The default value is 50.

  - **Maximum number of iterations** — specifies the maximum number of tuning iterations. This option is available only if the **Search method** is **Genetic algorithm** or **Bayesian**. The default value is 5.

  - **Maximum number of points in model** — specifies the maximum number of points in the model. This option is available only if the **Search method** is **Bayesian**. The default value is 100.

  - **Sample size** — specifies the sample size. This option is available only if the **Search method** is **Random** or **Latin hypercube sample**. The default value is 50.
- **General Options** — specifies the general properties for autotuning. The following options are available:

  - **Validation method** — specifies the validation method for finding the objective value. Note that if your data is partitioned, then that partition is used. **Validation method, Validation data proportion, and Cross validation number of folds** are all ignored. Here are the possible values:
    - **Partition** — specifies using the partition validation method. With partition, you will specify proportions to use for randomly assigning observations to each role.
    - **K-fold cross validation** — specifies using the cross validation method. In cross validation, each model evaluation requires \( k \) training executions (on \( k-1 \) data folds) and \( k \) scoring executions (on one holdout fold). This increases the evaluation time by approximately a factor of \( k \).

  For small to medium data tables, cross validation provides, on average, a better representation of error across the whole data table. **Partition** is the default value.

  - **Training data proportion** — specifies the proportion of training data to be used for the **Partition** validation method. The default value is 0.7.

  - **Validation data proportion** — specifies the proportion of data to be used for the **Partition** validation method. The default value is 0.3.

  - **Cross validation number of folds** — specifies the number of partition folds in the cross validation process (the \( k \) defined above). Possible values range from 2 to 20. The default value is 5.

  - **Class target objective function** — specifies the objective function to optimize for tuning parameters for a class target. Here are the possible values:
    - Area under the curve
    - Average squared error
    - F0.5 score
    - F1 score
    - Gamma
    - Gini coefficient
    - Kolmogorov-Smirnov statistic
    - Misclassification rate
    - Multi-class log loss
    - Root average squared error
    - Tau

  The default value is **Kolmogorov-Smirnov statistic**.

  - **Interval target objective function** — specifies the objective function to optimize for tuning parameters for an interval target. Here are the possible values:
    - Average squared error
    - Mean absolute error
    - Mean squared logarithmic error
- Root average squared error
- Root mean absolute error
- Root mean squared logarithmic error

The default value is Average squared error.

- **Maximum time (minutes)** — specifies the maximum time (in minutes) for the optimization tuner. The default value is 60.
- **Maximum training time for single model (minutes)** — specifies the maximum training time (in minutes) for the single model.

### Binary Classification Cutoff

- **Specify node binary classification cutoff** — specifies whether to use the binary classification cutoff specified below for the node. If you deselect, the project binary classification cutoff is used for determining the predicted value for a binary target based on the posterior probabilities. By default, this value is deselected.

- **Node binary classification cutoff** — specifies the cutoff to use in the node for determining the predicted value for a binary target based on the posterior probabilities. This option is available only if Specify node binary classification cutoff is selected. The default value is 0.5.

### Model Interpretability

Specifies the method of model interpretation in order to show how the model responds to changing inputs.

- **Explain Model Inputs** — specifies the following parameters for the LIME algorithm:
  - **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.
  - **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.
  - **Cluster-based ICE plots** — the maximum number of variables (1-10).

- **Explain Individual Predictions** — specifies whether to include Number of clusters, Number of perturbed instances, and Number of iterations. You can specify the Distance metric as either Cosine or K prototype. You can specify the Kernel width and the Number of inputs to use in the explanation. You can specify whether to Create a score code for assigning clusters.
Forest Results

After running the node, you can open the Results window by right-clicking the node and selecting Results from the pop-up menu.

Node

- **Error Plot** — displays the error statistics for the forest model, giving the average squared error as a function of the number of trees. The average squared error is given for each of the data roles, as well as for the out-of-bag data. To examine the misclassification rate as a function of the number of trees when you have a class target, use the drop-down menu in the upper right corner.

- **Variable Importance** — displays a table of importance data for each variable. These statistics include train importance, importance standard deviation, and relative importance.

- **Autotune Best Configuration** — displays a table of the resulting values of parameters that were autotuned. This table will not be displayed if autotuning is not selected.

- **Autotune Results** — displays the iterative results of the autotuning process, showing how each of the parameter values evolve as the model iterates. This table will not be displayed if autotuning is not selected.

- **Score Inputs** — displays a table of data on the input variables for scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.

- **Score Outputs** — displays a table of data on the predicted response variable for scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.

- **Path EP Score Code** — displays the SAS code that was created by the node. The score code can be used outside of the Model Studio environment to score new data.

- **DS2 Package Code** — displays the SAS DS2 package code that was created by the node. The score code can be used outside of the Model Studio environment to score new data.

- **Train Code** — displays the SAS code that Model Studio used to train the node.

- **Properties** — specifies the various properties selected before running the node.

- **Output** — displays the SAS output of the Forest run. For the Forest node, the output includes model information, number of observations read and used, variable importance, fit statistics, predicted probability variables, and predicted target variable.

Assessment

- **Predicted Reports** — displays the predicted and target mean as a function of the depth for the model. The predicted and target mean are given for each of the data roles. To examine the predicted mean as a function of the target mean, use the drop-down menu in the upper right corner. This result is displayed only if the target is an interval variable.
Lift Reports — displays the cumulative lift as a function of depth for the model. The cumulative lift is given for each of the data roles. To examine other statistics as a function of depth, use the drop-down menu in the upper right corner. Other statistics include lift, gain, % captured response, cumulative % captured response, % response, and cumulative % response. This result is displayed only if the target is a class variable.

ROC Reports — displays the ROC (receiver operating characteristic) chart for a model, giving the sensitivity as a function of 1-specificity. The sensitivity is given for each of the data roles. To examine other statistics, use the drop-down menu in the upper right corner. Other statistics include accuracy and F1 score. This result is displayed only if the target is a class variable.

Fit Statistics — displays the fit statistics for the model, broken down by data role.

Model Interpretability

Input Relative Importance — displays a table of the relative importance of the input variables, including variable level and label.

Partial Dependence — displays the functional relationship between the input variable and the model prediction.

LIME Plots — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed around a particular observation to create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when Explain Individual Predictions is selected.

- Local Instances — displays the values for each input variable of the cluster center.
- Local Explanation — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.
- Local Model — displays the parameter estimates of the local linear model that is used to predict the cluster center.
- Prediction — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

LIME Score Code — displays the SAS code that Model Studio used to assign observations to clusters. This is only displayed when Create score code for assigning clusters is selected.

Individual Conditional Expectation — displays the functional relationship between the input variable and the model prediction. The relationships for all cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.
Overview of GLM

The GLM node is a Supervised Learning node that fits a generalized linear model (GLM) for an interval target with a specified target distribution and link function. A GLM is an extension of the traditional linear model. This extension allows the population mean to depend on a linear predictor through a nonlinear link function. For example, GLMs can be used to model traditional insurance measures such as claim frequency, severity, or pure premium. Claim frequency is typically modeled with a Poisson distribution and a logarithmic link function. Claim severity is typically modeled with a gamma distribution and a logarithmic link function. Pure premiums are modeled with the Tweedie distribution.

The GLM node can fit models for standard distributions in the exponential family, as well as the beta and generalized Poisson distributions.
GLM Properties

General Properties

- **Target probability distribution** — specifies the probability distribution to use in the model for the target. Here are the possible values:
  - Beta
  - Binary
  - Exponential
  - Gamma
  - Generalized Poisson
  - Geometric
  - Inverse Gaussian
  - Negative binomial
  - Normal
  - Poisson
  - Tweedie

  The default value is **Poisson**.

- **Link function** — specifies the link function to link the response mean to the linear predictor. Here are the possible values:
  - Complementary log-log
  - Identity
  - Inverse
  - Inverse squared
  - Log
  - Log-log
  - Logit
  - Probit

  The default value is **Log**.

Effects Options

- **Class input order** — specifies the sort order for the class inputs. For the GLM coding that is used, the last level according to the sorted order is used as the reference level. Here are the possible values:
- Formatted
- Formatted descending
- Frequency
- Frequency descending
- Unformatted
- Unformatted descending

The default value is Frequency.

- **Two-factor interactions** — specifies whether to include all two-factor interactions of class variables in the model. By default, Two-factor interactions is deselected.

- **Factor split** — specifies whether levels of a factor can enter or leave a model independently. This is done automatically for LASSO selection. By default, Factor split is deselected.

- **Spline** — specifies whether to expand interval inputs into cubic B-spline bases with three equally spaced knots. (This action yields seven design matrix columns for each of the variables.) If Spline is selected, Polynomial is unavailable. By default, Spline is deselected.

- **Spline split** — specifies whether each individual column in the design matrix that corresponds to the spline effect is treated as a separate effect that can enter or leave the model. This option is available only if Spline is selected. By default, Spline split is deselected.

- **Polynomial** — specifies whether to use model polynomial effects up to the degree specified for all interval variables. If Polynomial is selected, Spline is unavailable. By default, Polynomial is deselected.

- **Polynomial degree** — specifies the polynomial degree when polynomial terms are included in the model. Possible values are 2 and 3. The default value is 2.

- **Suppress intercept** — specifies whether the intercept should be suppressed. By default, Suppress intercept is deselected.

- **Model missing values for inputs** — specifies whether to model missing values for inputs by including indicator values in the model for missing values for interval inputs and imputing values in the original effect with the mean as well as treating missing values for class inputs as valid levels. By default, Model missing values for inputs is deselected.

- **Use missing as level** — specifies whether missing values should be treated as a level for class inputs. By default, Use missing as level is deselected.

- **Standardize interval inputs** — specifies whether interval inputs should be standardized (centered and scaled) when used as main effects. By default, Standardize interval inputs is deselected.

### Selection Method Property

- **Selection method** — specifies a model selection method. Here are the possible values:
- **Backward** — specifies that training is done by starting with all candidate effects in the model and then removing effects until the Stay significance level or the stop criterion is met.

- **Fast backward** — specifies that training is done by starting with all effects in the model and then deleting effects without refitting the model.

- **Forward** — specifies that training is done by starting with no candidate effects in the model and then adding effects until the Entry significance level or the stop criterion is met.

- **LASSO** — specifies that training is done using the group LASSO method, which adds and removes effects by a sequence of LASSO steps.

- **None** — specifies that there is no model selection method.

- **Stepwise** — specifies that training is done as in the forward model but that it might remove effects already in the model.

The default value is **Stepwise**.

### Selection Options

- **Effect-selection criterion** — specifies the criterion that the procedure uses to determine the order in which effects enter or leave at each step of the selection method. Here are the possible values:
  - **AIC** — specifies Akaike’s Information Criterion. Smaller values indicate better models, and AIC values can become negative. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model.
  - **AICC** — specifies Corrected Akaike’s Information Criterion. This version of AIC adjusts the value to account for sample size. The result is that extra effects penalize AICC more than AIC. As the sample size increases, AICC and AIC converge.
  - **SBC (BIC)** — specifies Schwarz’s Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). SBC is an increasing function of the model’s residual sum of squares and the number of effects. Unexplained variations in the response variable and the number of effects increase the value of the SBC. As a result, a lower SBC implies either fewer explanatory variables, better fit, or both. SBC penalizes free parameters more strongly than AIC.
  - **Significance level** — specifies the standard statistical significance level criterion.

If the **Selection method** above is **LASSO**, this option is unavailable. The default value is **SBC (BIC)**.

- **Selection-process stopping criterion** — specifies the criterion to stop the selection process. Here are the possible values:
  - **AIC** — specifies Akaike’s Information Criterion. Smaller values indicate better models, and AIC values can become negative. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model.
- **AICC** — specifies Corrected Akaike’s Information Criterion. This version of AIC adjusts the value to account for sample size. The result is that extra effects penalize AICC more than AIC. As the sample size increases, AICC and AIC converge.

- **SBC (BIC)** — specifies Schwarz’s Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). SBC is an increasing function of the model’s residual sum of squares and the number of effects. Unexplained variations in the response variable and the number of effects increase the value of the SBC. As a result, a lower SBC implies either fewer explanatory variables, better fit, or both. SBC penalizes free parameters more strongly than AIC.

- **Significance level** — specifies the standard statistical significance level criterion.

- **Validation ASE** — average square error (ASE) of the model is computed using the validation data, and selection stops at the step where the ASE starts to increase. This method requires partitioned data. The default value is **SBC (BIC)**.

- **Model-selection criterion** — specifies the criterion to choose the model from the list of models at each step of the selection process that yields the best value of the specified criterion. If the optimal value of the specified criterion occurs for models at more than one step, then the model that has the smallest number of parameters is chosen. Here are the possible values:

  - **AIC** — specifies Akaike’s Information Criterion. Smaller values indicate better models, and AIC values can become negative. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model. The model that has the minimal AIC value is chosen.

  - **AICC** — specifies Corrected Akaike’s Information Criterion. This version of AIC adjusts the value to account for sample size. The result is that extra effects penalize AICC more than AIC. As the sample size increases, AICC and AIC converge.

  - **SBC (BIC)** — specifies Schwarz’s Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). SBC is an increasing function of the model’s residual sum of squares and the number of effects. Unexplained variations in the response variable and the number of effects increase the value of the SBC. As a result, a lower SBC implies either fewer explanatory variables, better fit, or both. SBC penalizes free parameters more strongly than AIC. The model that has the minimal SBC value is chosen.

  - **Validation ASE** — average square error (ASE) of the model is computed using the validation data, and the model that has the minimal ASE is chosen. This method requires partitioned data. The default value is **SBC (BIC)**.

- **Entry significance level** — specifies the significance level for adding variables in the forward and stepwise directions. The default value is 0.05.

- **Stay significance level** — specifies the significance level for removing variables in backward and stepwise directions. The default value is 0.05.

- **Maximum number of effects** — specifies the maximum number of effects to be considered in any model during the selection process. If a model at some step of the selection process contains the specified maximum number of effects, then no
additional effects are considered. If Maximum number of effects is set to 0 (the default value), this option is ignored.

- **Minimum number of effects** — specifies the minimum number of effects to be considered in any model during the backward selection process. If Minimum number of effects is set to 0 (the default value), this option is ignored.

- **Maximum number of steps** — specifies the maximum number of selection steps that are performed. If Maximum number of steps is set to 0 (the default value), this option is ignored.

- **Hierarchy** — specifies whether and how the model hierarchy requirement is applied. Model hierarchy refers to the requirement that, for any term to be in the model, all model effects that are contained in the term must be present in the model. For example, in order for the interaction A*B to enter the model, the main effects A and B must also be in the model. Likewise, neither effect A nor effect B can leave the model while the interaction A*B is in the model. Here are the possible values:
  - All variables — specifies that both interval and class variables are subject to the model hierarchy requirement.
  - Class variables — specifies that only class variables are subject to the model hierarchy requirement.
  - None — specifies that the hierarchy requirement is never applied.

The default value is None.

### Optimization Options

- **Optimization technique** — specifies the optimization method used when fitting a model. Here are the possible values:
  - Conjugate-gradient
  - Double-dogleg
  - Dual quasi-Newton
  - Nelder-Mead simplex
  - Newton-Raphson
  - Newton-Raphson with ridging
  - None
  - Trust-region

The default value is **Newton-Raphson with ridging**. For more information, see the PROC NL MIXED documentation in SAS/STAT 14.3 User’s Guide.

Note: Optimization options are not available for the LASSO selection method. LASSO uses the Nesterov algorithm as the default optimization technique.

- **Maximum number of iterations** — specifies the maximum number of iterations of any optimization. The default value depends on the optimization technique used:

<table>
<thead>
<tr>
<th>Optimization Technique</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjugate-gradient</td>
<td></td>
</tr>
<tr>
<td>Double-dogleg</td>
<td></td>
</tr>
<tr>
<td>Dual quasi-Newton</td>
<td></td>
</tr>
<tr>
<td>Nelder-Mead simplex</td>
<td></td>
</tr>
<tr>
<td>Newton-Raphson</td>
<td></td>
</tr>
<tr>
<td>Newton-Raphson with ridging</td>
<td></td>
</tr>
<tr>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Trust-region</td>
<td></td>
</tr>
</tbody>
</table>

Optimization Options
<table>
<thead>
<tr>
<th>Optimization Technique</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjugate-gradient</td>
<td>400</td>
</tr>
<tr>
<td>Double-dogleg</td>
<td>200</td>
</tr>
<tr>
<td>Dual quasi-Newton</td>
<td>200</td>
</tr>
<tr>
<td>Nelder-Mead simplex</td>
<td>1000</td>
</tr>
<tr>
<td>Newton-Raphson</td>
<td>50</td>
</tr>
<tr>
<td>Newton-Raphson with ridging</td>
<td>50</td>
</tr>
<tr>
<td>Trust-region</td>
<td>50</td>
</tr>
</tbody>
</table>

To use the default value, leave **Maximum number of iterations** blank or use a dot.

- **Maximum number of function evaluations** — specifies the maximum number of function calls of any optimization. The default value depends on the optimization technique used:

<table>
<thead>
<tr>
<th>Optimization Technique</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjugate-gradient</td>
<td>1000</td>
</tr>
<tr>
<td>Double-dogleg</td>
<td>500</td>
</tr>
<tr>
<td>Dual quasi-Newton</td>
<td>500</td>
</tr>
<tr>
<td>Nelder-Mead simplex</td>
<td>3000</td>
</tr>
<tr>
<td>Newton-Raphson</td>
<td>125</td>
</tr>
<tr>
<td>Newton-Raphson with ridging</td>
<td>125</td>
</tr>
<tr>
<td>Trust-region</td>
<td>125</td>
</tr>
</tbody>
</table>

To use the default value, leave **Maximum number of function evaluations** blank or use a dot.

- **Maximum CPU time in second** — specifies an upper limit of CPU time (in seconds) for the optimization process. The default value is the largest floating-point double representation of your computer. To use the default value, do not enter a value for **Maximum CPU time in second**, or enter a dot.

- **Minimum number of iterations** — specifies the minimum number of iterations in any optimization. The default value is 1. To use the default value, do not enter a value for **Minimum number of iterations** or enter a dot.

- **Normalize objective function** — specifies whether the objective function should be normalized during optimization by the reciprocal of the used frequency count. By default, this option is selected.
Convergence Options

- **Absolute function convergence** — specifies the threshold for absolute function convergence. The default value is the negative square root of the largest double-precision value. To use the default value, do not enter a value for Absolute function convergence or enter a dot.

- **Absolute function difference convergence** — specifies the threshold for absolute function difference convergence. The default value is 0. To use the default value, do not enter a value for Absolute function difference convergence or enter a dot.

- **Absolute gradient convergence** — specifies the threshold for absolute gradient convergence. The default value is 1E-5. To use the default value, do not enter a value for Absolute gradient convergence or enter a dot.

- **Relative function difference convergence** — specifies the relative function difference convergence. The default value is the machine precision times 2. To use the default value, do not enter a value for Relative function difference convergence or enter a dot.

- **Relative gradient convergence** — specifies the threshold for relative gradient convergence. The default value is 1E-8. To use the default value, leave Relative gradient convergence blank or use a dot.

Tweedie Model Options

These configurations are available only when the **Target probability distribution** is **Tweedie**.

- **Optimization method** — specifies the optimization method for iterative estimation of Tweedie model parameters. Here are the possible values:
  - **EQL/Likelihood** — specifies using EQL for a sample of the data, followed by Tweedie log likelihood for the full data.
  - **Extended quasi-likelihood** — specifies using extended quasi-likelihood (EQL) for a sample of the data, followed by EQL for the full data.
  - **Final likelihood** — specifies using a four-stage approach of EQL and likelihood.
  - **Likelihood** — specifies using Tweedie log likelihood for a sample of the data, and then specifies again for the full data.

  The default value is EQL/Likelihood.

- **Use fixed power parameter** — specifies whether to use a fixed power parameter. Otherwise, iterative estimation is used for the Tweedie power parameter with the specified starting value. By default, Use fixed power parameter is selected.

- **Fixed power parameter value** — specifies the value of the Tweedie power parameter when it is fixed. The default value is 1.5.
- **Starting power parameter value** — specifies the starting value for the iterative estimation of the Tweedie power parameter. This option is available only if **Use fixed power parameter** is deselected. The default value is 1.5.

- **Sample fraction for Tweedie starting values** — specifies the fraction of training data to use to compute the starting values for the Tweedie distribution. The default value is 1.

---

### Relativity Plot Properties

- **Create relativity plots** — specifies whether relativity plots are created for each class main effect that is included in the final model. By default, this option is deselected.

- **Confidence level** — specifies the confidence level as a percentage of confidence intervals used in relativity plots. This option is available only if **Create relativity plots** is selected. The default value is 95.

---

### Model Interpretability

Specifies the method of model interpretation in order to show how the model responds to changing inputs.

- **Explain Model Inputs** — specifies the following parameters for the LIME algorithm:
  - **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.
  - **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.
  - **Cluster-based ICE plots** — the maximum number of variables (1-10).

- **Explain Individual Predictions** — specifies whether to include **Number of clusters**, **Number of perturbed instances**, and **Number of iterations**. You can specify the **Distance metric** as either **Cosine** or **K prototype**. You can specify the **Kernel width** and the **Number of inputs to use in the explanation**. You can specify whether to **Create a score code for assigning clusters**.

---

### GLM Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Node**
- **Relativity Plots** — displays a relativity plot for each class main effect included in the final model. This plot shows the relativity for each level with regard to the reference level—that is, the exponential of the parameter estimate. The bands on the plots indicate the lower and upper confidence limits, which are based on the specified **Confidence level**. You can select which variable to display in the plot using the drop-down menu in the upper right corner. This result is displayed only if **Create relativity plots** is selected.

- **t Values by Parameter** — displays a bar chart of the t values associated with each of the input parameters, as well as the intercept. Positive t values are denoted with blue bars, and negative t values are denoted with orange bars.

- **Parameter Estimates** — displays a table of various statistics related to estimates for the parameters. These statistics include the t value, sign, estimate, absolute estimate, the p-value, chi-square value, and standard error.

- **Selection Summary** — displays a table of the iterations, adding in and removing effects. This table also includes the SBC and optimal SBC.

- **GLM Fit Statistics** — displays a table of data on the GLM fit statistics. The table includes the statistic, a corresponding qualitative description, and the training, validation, and testing values.

- **Node Score Code** — displays the SAS score code that was created by the node. The SAS score code can be used outside of the Model Studio environment to score new data.

- **Score Inputs** — displays a table of data on the input variables for scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.

- **Score Outputs** — displays a table of data on the predicted response variable for scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.

- **Path Score Code** — displays the SAS score code that was created by the node, as well as all preceding nodes. The SAS model score code can be used outside of the Model Studio environment to score new data.

- **Train Code** — displays the SAS code that Model Studio used to train the node.

- **Properties** — specifies the various properties selected before running the node.

- **Output** — displays the SAS output of the GLM run.

**Assessment**

- **Predicted Reports** — displays the predicted and target mean as a function of the depth for the model. The predicted and target mean are given for each of the data roles. To examine the predicted mean as a function of the target mean, use the drop-down menu in the upper right corner.

- **Fit Statistics** — displays the fit statistics for the model, broken down by data role.

**Model Interpretability**

- **Input Relative Importance** — displays a table of the relative importance of the input variables, including variable level and label.

- **Partial Dependence** — displays the functional relationship between the input variable and the model prediction.

- **LIME Plots** — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed.
around a particular observation to create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when Explain Individual Predictions is selected.

- **Local Instances** — displays the values for each input variable of the cluster center.
- **Local Explanation** — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.
- **Local Model** — displays the parameter estimates of the local linear model that is used to predict the cluster center.
- **Prediction** — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

- **LIME Score Code** — displays the SAS code that Model Studio used to assign observations to clusters. This is displayed only when Create score code for assigning clusters is selected.
- **Individual Conditional Expectation** — displays the functional relationship between the input variable and the model prediction. The relationships for all cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.
Overview of Gradient Boosting

The **Gradient Boosting** node is a Supervised Learning node. Gradient boosting is an iterative approach that creates multiple trees where, typically, each tree is based on an independent sample without replacement of the data. The gradient boosting model hones its predictions by minimizing a specified loss function, such as average square error. The first step creates a baseline tree. Each subsequent tree is fit to the residuals of the previous tree, and the loss function is minimized. This process is repeated a specific number of times. The final model is a single function, which is an aggregation of the series of trees that can be used to predict the target value of a new observation.

The term “stochastic gradient boosting” refers to training each new tree based on a subsample of the data. This typically results in a better model. For gradient boosting models, each new observation is fed through a sequence of trees that are created to predict the target value of each new observation.
Gradient Boosting Properties

General Properties

- **Number of trees** — specifies the number of iterations in the boosting series. For interval and binary targets, the number of iterations equals the number of trees. For a nominal target, a separate tree is created for each target category in each iteration. The default value is 100.
- **Learning rate** — specifies the step size for gradient descent optimization. The default value is 0.1.
- **Subsample rate** — specifies the proportion of training observations to train a tree with. A different training sample is taken in each iteration. Trees trained in the same iteration have the same training data. The default value is 0.5.
- **L1 regularization** — specifies the L1 regularization parameter. The default value is 0.
- **L2 regularization** — specifies the L2 regularization parameter. The default value is 0.
- **Seed** — specifies the seed for generating random numbers. The subsample rate property uses this value to select a training sample at each iteration. The default value is 12345.

Tree-Splitting Options

- **Maximum number of branches** — specifies the maximum number of branches to consider for a teach node split in the tree. Possible values range from 2 to 5. The default value is 2.
- **Maximum depth** — specifies the maximum number of generations of nodes. The original node, generation 0, is called the root node. The children of the root node are the first generation. Possible values range from 1 to 50. The default value is 6.
- **Minimum leaf size** — specifies the smallest number of training observations that a leaf can have. The default value is 5.
- **Missing values** — specifies the action to take when there are values missing. You can specify to Ignore, Use as machine smallest, or Use in search. The default value is Use in search.
- **Minimum missing use in search** — specifies a threshold for using missing values in the split search. The default value is 1.
- **Number of interval bins** — specifies the number of bins in which to bin the interval input variables. The default value is 20.
- **Interval bin method** — specifies the method used to bin the interval input variables. Select **Bucket** to divide inputs into evenly spaced intervals based on the difference between maximum and minimum values. Select **Quantile** to divide inputs into approximately equal sized groups. The default value is **Bucket**.

- **Use default number of inputs to consider per split** — specifies whether to use the default number of inputs to consider per split. By default, this option is selected.

- **Number of inputs to consider per split** — specifies the number of input variables randomly sampled to use per split. This option is not available if **Use default number of inputs to consider per split** is selected. The default value is 100.

---

**Perform Early Stopping**

**Perform Early Stopping** — specifies whether to stop training when the model begins to overfit. The training stops if the relative error in each of the $N$ consecutive iterations (**Stagnation**) is less than a threshold (**Tolerance**). Early stopping cannot be used if there is no validation partition. By default, this option is selected. The following options are available:

- **Stagnation** — specifies the number of consecutive iterations ($N$) for early stopping. The default value is 5.

- **Tolerance** — specifies the threshold for early stopping. The default value is 0.

---

**Perform Autotuning**

This feature specifies whether to perform autotuning of any gradient boosting parameters. Warning: Performing autotuning can substantially increase run time. If **Perform Autotuning** is selected, the following options are available:

- **L1 Regularization** — specifies whether to autotune L1 Regularization. If this option is selected, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the L1 Regularization. The default value is 0. Use the **From** and **To** options to specify the range. The default **From** value is 0, and the default **To** value is 10.

- **L2 Regularization** — specifies whether to autotune L2 Regularization. If this option is selected, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the L2 Regularization. The default value is 0. Use the **From** and **To** options to specify the range. The default **From** value is 0, and the default **To** value is 10.

- **Learning Rate** — selects whether to autotune the learn rate. If this option is selected, the following options are available:
  - **Initial value** — specifies the initial value for autotuning the learn rate. The default value is 0.1. Use the **From** and **To** options to specify the range. The default **From** value is 0.01, and the default **To** value is 1.

- **Number of Inputs per Split** — selects whether to autotune the number of inputs per split. If selected, the following option is available:
- **Initial value** — specifies the initial value for the number of inputs per split. The default value is 100. Use the From and To options to specify the range. The default From value is 1, and the default To value is 100.

- **Number of trees** — selects whether to autotune the number of trees in a boosting series. If selected, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the number of iterations. The default value is 100. Use the From and To options to specify the range. The default From value is 20, and the default To value is 150.

- **Subsample Rate** — selects whether to autotune the subsample rate. If selected, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the subsample rate. The default value is 0.5. Use the From and To options to specify the range. The default From value is 0.1, and the default To value is 1.

- **Search Options** — specifies the options for autotuning searching. The following options are available:
  - **Search method** — specifies the tuning search method. Here are the possible values:
    - **Bayesian** — specifies the Bayesian method. This method uses priors to seed the iterative optimization.
    - **Genetic algorithm** — specifies the genetic algorithm method. This method uses an initial Latin Hypercube sample that seeds a genetic algorithm to generate a new population of alternative configurations at each iteration.
    - **Grid** — specifies the grid method. This method performs an exhaustive search of all configurations. Each hyperparameter of interest is discretized into a desired set of values to be studied. Models are trained and assessed for all combinations of values across all the hyperparameters (thus forming a multi-dimensional “grid”).
    - **Latin hypercube sample** — specifies the Latin Hypercube method. This method performs an optimized grid search that is uniform in each tuning parameter, but random in combinations.
    - **Random** — specifies the Random method. This method generates a single sample of purely random configurations.

The default value is **Genetic algorithm**.

- **Number of evaluations per iteration** — specifies the number of tuning evaluations in one iteration. This option is available only if the Search method is **Genetic algorithm** or **Bayesian**. The default value is 10.

- **Maximum number of evaluations** — specifies the maximum number of tuning evaluations. This option is available only if the Search method is **Genetic algorithm** or **Bayesian**. The default value is 50.

- **Maximum number of iterations** — specifies the maximum number of tuning iterations. This option is available only if the Search method is **Genetic algorithm** or **Bayesian**. The default value is 5.

- **Maximum number of points in model** — specifies the maximum number of points in the model. This option is available only if the Search method is **Bayesian**. The default value is 100.
- **Sample size** — specifies the sample size. This option is available only if the **Search method** is **Random** or **Latin hypercube sample**. The default value is 50.

- **General Options** — specifies the general properties for autotuning. The following options are available:
  - **Validation method** — specifies how to partition the data for assessing the models. Note that if your data is partitioned, then that partition is used and **Validation method**, **Validation data proportion**, and **Cross validation number of folds** are all ignored. Here are the possible values:
    - **Partition** — specifies using a single partition of a training set. With partition, you will specify proportions to use for randomly assigning observations to each role.
    - **K-fold cross validation** — specifies using the k-fold cross validation method. In k-fold cross validation, each model evaluation requires k training executions (on k-1 data folds) and k scoring executions (on one holdout fold). This increases the evaluation time by approximately a factor of k.

    For small to medium data tables, cross validation provides, on average, a better representation of error across the whole data table. **Partition** is the default value.
  - **Training data proportion** — specifies the proportion of training data to be used for the **Partition** validation method. The default value is 0.7.
  - **Validation data proportion** — specifies the proportion of data to be used for the **Partition** validation method. The default value is 0.3.
  - **Cross validation number of folds** — specifies the number of partition folds in the cross validation process (the k defined above). Possible values range from 2 to 20. The default value is 5.
  - **Class target objective function** — specifies the objective function to optimize for tuning parameters for a class target. Here are the possible values:
    - **Area under the curve**
    - **Average squared error**
    - **F0.5 score**
    - **F1 score**
    - **Gamma**
    - **Gini coefficient**
    - **Kolmogorov-Smirnov statistic**
    - **Misclassification rate**
    - **Multi-class log loss**
    - **Root average squared error**
    - **Tau**

    The default value is **Kolmogorov-Smirnov statistic**.
  - **Interval target objective function** — specifies the objective function to optimize for tuning parameters for an interval target. Here are the possible values:
- Average squared error
- Mean absolute error
- Mean squared logarithmic error
- Root average squared error
- Root mean absolute error
- Root mean squared logarithmic error

The default value is **Average squared error**.

- **Maximum time (minutes)** — specifies the maximum time (in minutes) for the optimization tuner. The default value is 60.

- **Maximum training time for single model (minutes)** — specifies the maximum training time (in minutes) for the single model.

## Binary Classification Cutoff

- **Specify node binary classification cutoff** — selects whether to use the binary classification cutoff specified below for the node. If deselected, the project binary classification cutoff is used for determining the predicted value for a binary target based on the posterior probabilities. By default, this value is deselected.

- **Node binary classification cutoff** — specifies the cutoff to use in the node for determining the predicted value for a binary target based on the posterior probabilities. This option is available only if **Specify node binary classification cutoff** is selected. The default value is 0.5.

## Model Interpretability

Specifies the method of model interpretation in order to show how the model responds to changing inputs.

- **Explain Model Inputs** — specifies the following parameters for the LIME algorithm:
  
  - **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.
  
  - **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.

  - **Cluster-based ICE plots** — the maximum number of variables (1-10).

- **Explain Individual Predictions** — specifies whether to include **Number of clusters**, **Number of perturbed instances**, and **Number of iterations**. You can specify the **Distance metric** as either **Cosine** or **K prototype**. You can specify the **Kernel width** and the **Number of inputs to use in the explanation**. You can specify whether to **Create a score code for assigning clusters**.
Gradient Boosting Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Node**

- **Error Plot** — displays a graph of the average squared error as a function of the number of trees. The average squared error is given for each of the data roles.

- **Variable Importance** — displays a table of importance data for each variable. These statistics include train importance, importance standard deviation, and relative importance.

- **Autotune Best Configuration** — displays a table of the resulting values of parameters that were autotuned. This table will not be displayed if autotuning is not selected.

- **Autotune Results** — displays the iterative results of the autotuning process, showing how each of the parameter values evolve as the model iterates. This table will not be displayed if autotuning is not selected.

- **Score Inputs** — displays a table of data on the input variables for scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.

- **Score Outputs** — displays a table of data on the predicted response variable for scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.

- **Path EP Score Code** — displays the SAS code that was created by the node. The score code can be used outside of the Model Studio environment to score new data.

- **DS2 Package Code** — displays the SAS DS2 package code that was created by the node. The score code can be used outside of the Model Studio environment to score new data.

- **Train Code** — displays the SAS code that Model Studio used to train the node.

- **Properties** — specifies the various properties selected before running the node.

- **Output** — displays the SAS output of the Gradient Boosting run. For the **Gradient Boosting** node, the output includes the model information, number of observations read and used, variable importance, fit statistics, and predicted target variable.

**Assessment**

- **Predicted Reports** — displays the predicted and target mean as a function of the depth for the model. The predicted and target mean are given for each of the data roles. To examine the predicted mean as a function of the target mean, use the drop-down menu in the upper right corner. This result is displayed only if the target is an interval variable.

- **Lift Reports** — displays the cumulative lift as a function of the depth for the model. The cumulative lift is given for each of the data roles. To examine other statistics as a function of depth, use the drop-down menu in the upper right corner. Other statistics include lift, gain, % captured response, cumulative %
captured response, % response, and cumulative % response. This result is displayed only if the target is a class variable.

- **ROC Reports** — displays the ROC (receiver operating characteristic) chart for a model, giving the sensitivity as a function of 1-specificity. The sensitivity is given for each of the data roles. To examine other statistics, use the drop-down menu in the upper right corner. Other statistics include accuracy and F1 score. This result is displayed only if the target is a class variable.

- **Fit Statistics** — displays a table of the fit statistics for the model, broken down by data role.

**Model Interpretability**

- **Input Relative Importance** — displays a table of the relative importance of the input variables, including variable level and label.

- **Partial Dependence** — displays the functional relationship between the input variable and the model prediction.

- **LIME Plots** — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed around a particular observation to create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when **Explain Individual Predictions** is selected.

  - **Local Instances** — displays the values for each input variable of the cluster center.
  
  - **Local Explanation** — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.

  - **Local Model** — displays the parameter estimates of the local linear model that is used to predict the cluster center.

  - **Prediction** — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

- **LIME Score Code** — displays the SAS code that Model Studio used to assign observations to clusters. This is displayed only when **Create score code for assigning clusters** is selected.

- **Individual Conditional Expectation** — displays the functional relationship between the input variable and the model prediction. The relationships for all cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.
Overview of Imputation

The Imputation node is a Data Mining Preprocessing node. Use the Imputation node to replace missing values in data sets that are used for data mining.

Data mining databases often contain observations that have missing values for one or more variables. Missing values can result from data collection errors, incomplete customer responses, and actual system and measurement failures. They might also result from a revision of the data collection scope over time, such as tracking new variables that were not included in the previous data collection schema.

If an observation contains a missing value, then by default that observation is not used for modeling by nodes such as Neural Network or Regression. However, rejecting all incomplete observations might ignore useful or important information that is still contained in the nonmissing variables. Rejecting all incomplete observations might also bias the sample, since observations that have missing values might have other things in common as well.

How should missing data values be treated? There is no single correct answer. Choosing the "best" missing value replacement technique inherently requires the researcher to make assumptions about the true (missing) data. For example, researchers often replace a missing value with the mean of the variable. This approach assumes that the variable's data distribution follows a normal population response. Replacing missing values with the mean, median, or another measure of central tendency is simple. But it can greatly affect a variable's sample distribution. You should use these replacement statistics carefully and only when the effect is minimal.
Imputation Properties

General Properties

- **Impute non-missing variables** — specifies whether to generate an imputation score code regardless of the existence of missing values in the training data. By default, this option is deselected.

- **Missing percentage cutoff** — specifies the maximum percent of missing values that are allowed for a variable to be imputed. Variables whose missing percentage exceeds this cutoff are ignored. The default value is 50.

- **Reject original variables** — specifies whether to mark original variables that are imputed as rejected. Deselecting this option will keep original variables as input. By default, this option is selected.

- **Summary statistics** — specifies whether to generate summary statistics for the imputed variables. By default, this option is deselected.

Class Inputs

- **Default method** — specifies the default transformation method for all class input variables. Note that any transformation specified in the metadata takes precedence over the method specified here. Possible values are as follows:
  - **Constant Value** — specifies that missing values are replaced with a character specified in the **Constant character value** field.
  - **Count** — specifies that missing values are replaced with the variable’s most frequent value. There are no other configurations to set.
  - **Distribution** — specifies that missing values will be replaced with randomly assigned values from an empirical distribution of the nonmissing values of the variable. As a result, the Distribution imputation typically does not significantly change the distribution of the data. The initial seed value for randomization is specified in the **Distribution method random seed** field. The default value is 12345.
  - **None** — specifies that only the imputations specified on the **Data** pane are performed.

  The default value is **Count**.

Interval Inputs

- **Default method** — specifies the imputation statistic that you want to use to replace missing interval variables. Here are the possible values:
- **Constant value** — specifies that missing values will be replaced with the value that is specified in the **Constant number value** field. The default value is 0.

- **Maximum** — specifies that missing values will be replaced with the maximum value for the variable found in training.

- **Mean** — specifies that missing values will be replaced with the arithmetic average.

- **Median** — specifies that missing values will be replaced with the midpoint of a frequency distribution of the observed values.

- **Midrange** — specifies that missing values will be replaced with the maximum value plus the minimum value divided by 2.

- **Minimum** — specifies that missing values will be replaced with the minimum value for the variable found in training.

- **None** — specifies that only the imputations specified on the **Data** pane are performed.

  The default value is **Mean**.

- **Data limits for calculating values** — specifies how the data is used to calculate values used for default methods. Here are the possible values:
  - **All data** — specifies that the values used are based on the entire set of data.
  - **Trimmed** — specifies that trimmed data is used.
  - **Winsorized** — specifies that Winsorized data is used.

  The default value is **All data**.

*Note:* The **Data limits for calculating values** field is available only when the default method above is set to **Mean**, **Midrange**, **Minimum**, or **Maximum**.

- When the default method is set to **Constant value**, the **Constant number value** option specifies the default numeric value that will be used to replace missing numeric values (class or interval).

---

### Constant Values

- **Constant character value** — specifies the replacement value that is used when the default class imputation method is **Constant value**.

- **Constant number value** — specifies the replacement value that is used when the default interval imputation method is **Constant value**.

---

### Indicators

- **Single indicator** — specifies the creation of a single variable that indicates a count of any inputs that were missing in training.

- **Unique indicator** — specifies the creation of a separate variable for each input with missing values in the training data.
If either the Single indicator or Unique indicator options are selected, the Indicator subject option is available. This property specifies the subject of the indicator variables to be created. Here are the possible values:

- **Imputed variables** — specifies to set the indicator for variables that have been imputed.
- **Missing variables** — specifies to set the indicator for missing values.

The default value is **Imputed variables**.

If either the Single indicator or Unique indicator options are selected, the Indicator role option is available. This property specifies the role to be assigned to the created indicator variables. Here are the possible values:

- **Input**
- **Rejected**

The default value is **Rejected**.

---

**Imputation Results**

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Results**

- **Input Variable Statistics** — displays relevant imputation statistics on input variables. This table gives the variable name, variable level, number of missing values, percent of missing values, and whether the variable is imputable (1 for yes, 0 for no). It also lists the minimum, maximum, mean, midrange, and standard deviation for each input.

- **Imputation Summary for Interval Variables** — displays a summary of the imputation run for interval variables. This table gives the following properties:
  - Imputed Variable
  - Method
  - Input Variable
  - Indicator Variable
  - Value
  - Number of Imputed Values
  - Percentage of Imputed Values
  - Minimum
  - Maximum
  - Mean
  - Importance Standard Deviation
  - Kurtosis
  - Midrange
  - Skewness
- Variable Label

- **Imputation Summary for Class Variables** — displays a summary of the imputation run for class variables. This table gives the following properties:
  - Imputed Variable
  - Method
  - Input Variable
  - Indicator Variable
  - Value
  - Variable Level
  - Number of Imputed Values
  - Percentage of Imputed Values
  - Type
  - Variable Length
  - Variable Label

- **Node Score Code** — displays the SAS score code that was created by the node. The SAS score code can be used outside of the Model Studio environment in custom user applications.

- **Properties** — specifies the various properties selected before running the node.

- **Output** — displays the SAS output of the imputation run.
Overview of Linear Regression

The **Linear Regression** node is a Supervised Learning node. A linear regression attempts to predict the value of an interval target variable as a linear function of one or more features. The linear regression uses the least squares method to determine the model. The least squares method creates a line of best fit by minimizing the residual sum of squares for every instance in the input data set. The residual sum of squares is the vertical distance between an instance and the line of best fit. The least squares method assumes the distribution of the target variable is normal with constant variance.

The linear regression requires an interval target variable and at least one feature.

Linear Regression Properties

**Effects Options**

- **Two-factor interactions** — specifies whether to include all two-factor interactions of nominal variables in the model. By default, this option is deselected.

- **Factor split** — specifies whether levels of a nominal variable can enter or leave a model independently. This is done automatically for LASSO selection. For other selection methods, this option is deselected.
Spline — specifies whether to expand interval inputs into cubic B-spline bases with three equally spaced knots. (Expansion yields seven design matrix columns for each of the variables.) Spline and polynomial effects cannot be specified together. By default, this option is deselected.

Spline split — specifies whether each individual column in the design matrix that corresponds to the spline effect is treated as a separate effect that can enter or leave the model. By default, this option is deselected. This option is available only if Spline is selected.

Polynomial — specifies whether to use polynomial effects up to the specified polynomial degree for all interval variables. Spline and polynomial effects cannot be specified together. By default, this option is deselected.

Polynomial degree — specifies the polynomial degree when polynomial terms are included in the model. This option is available only if Polynomial has been selected. Possible values are 2 and 3. The default value is 2.

Suppress intercept — specifies whether to suppress the intercept. By default, this is deselected.

Model missing values for inputs — specifies whether to model missing values for inputs by including indicator values in the model for missing values for interval inputs and imputing missing values in the original effect with the mean as well as treating missing values for class inputs as valid levels. By default, this is deselected.

Use missing as level — specifies whether missing values should be treated as a separate level for all nominal inputs. By default, this is deselected.

Selection Method Property

Selection method — specifies a model selection method. Here are the possible values:

- Adaptive LASSO — specifies to use adaptive weights when applying L1 regularization of the LASSO method. By default, ordinary least squares estimates of the regression coefficients are used as adaptive weights.

- Backward — specifies that training is done by least squares regression that starts with all candidate features in the model. Features are removed one at a time until the stop criterion is met.

- Forward — specifies that training is done by least squares regression that starts with no candidate features in the model. Features are added one at a time until the stop criterion is met.

- LASSO — specifies that training is done using the group LASSO method, which adds and removes features by a sequence of LASSO steps.

- None — specifies not to perform variable selection.

- Stepwise — specifies that training is done by least squares regression that starts as in the forward model but might remove features already in the model.

The default value is Stepwise.
Selection Options

- **Effect-selection criterion** — specifies the criterion that the procedure uses to determine the order in which features enter or leave at each step of the selection method. Here are the possible values:
  - **Adjusted R-square** — specifies the Adjusted R-square value. The Adjusted R-Square adjusts the R-square value by accounting for the addition of more features. Values can range from 0 to 1. Values closer to 1 are preferred.
  - **AIC** — specifies Akaike’s Information Criterion. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model. Smaller values indicate better models, and AIC values can become negative.
  - **AICC** — specifies Corrected Akaike’s Information Criterion. AICC adjusts the AIC value by accounting for sample size. Extra features penalize AICC more than AIC. As the sample size increases, AICC and AIC converge.
  - **Mallows’ Cp** — specifies Mallows’s Cp value. Smaller values indicate better models, and Cp values can become negative.
  - **R-square** — specifies the R-squared value. R-squared value is an indicator of how well the model fits the data. R-squared values can range from 0 to 1. Values closer to 1 are preferred.
  - **SBC (BIC)** — specifies Schwarz’s Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). Unexplained variations in the response variable and the number of features increase the value of the SBC. Smaller values indicate better models. SBC penalizes total number of model parameters more strongly than AIC, and hence tends to choose more sparse models.
  - **Significance level** — specifies the standard statistical significance level criterion.

If the **Selection method** above is LASSO, this option is unavailable. The default value is **SBC (BIC)**.

- **Selection-process stopping criterion** — specifies the criterion to stop the selection process. Here are the possible values:
  - **Adjusted R-square** — specifies the Adjusted R-square value. The Adjusted R-Square value adjusts the R-square value by accounting for the number of features in the model. Values can range from 0 to 1. Values closer to 1 are preferred. Selection stops at the step where adjusted R-square starts to decrease.
  - **AIC** — specifies Akaike’s Information Criterion. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model. Smaller values indicate better models, and AIC values can become negative. Selection stops at the step where AIC starts to increase.
  - **AICC** — specifies Corrected Akaike’s Information Criterion. AICC adjusts the AIC value by accounting for sample size. Extra features penalize AICC more than AIC. Smaller values indicate better models. As the sample size increases, AICC and AIC converge. Selection stops at the step where AICC starts to increase.
- **Mallows’ Cp** — specifies Mallows’s Cp value. Smaller values indicate better models, and Cp values can become negative.

- **None** — specifies that the selection process continues until all the features are in the model or if a size-based limit (see **Maximum number of effects**) is achieved.

- **Predicted RSS** — specifies that the selection process continues until the predicted residual sum of squares (RSS) starts to increase. This option is not valid with **LASSO** selection.

- **SBC (BIC)** — specifies Schwarz’s Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). Unexplained variations in the response variable and the number of features increase the value of the SBC. SBC penalizes total number of parameters more strongly than AIC, and hence tends to choose more sparse models. Selection stops at the step where SBC starts to increase.

- **Significance level** — specifies the standard statistical significance level criterion. Selection stops at the step where the significance level exceeds the specified level. The default value is 0.05.

- **Validation ASE** — average square error (ASE) of the model that is computed by using the validation data. Selection stops at the step where the ASE starts to increase. This method requires partitioned data.

The default value is **SBC (BIC)**.

- **Model-selection criterion** — specifies the criterion to choose the model (from the list of models at each step of the selection process) that yields the best value of the specified criterion. If the optimal value of the specified criterion occurs for models at more than one step, then the model that has the smallest number of parameters is chosen. Here are the possible values:

  - **Adjusted R-square** — specifies the Adjusted R-square value. The Adjusted R-Square value adjusts the R-square value by accounting for the number of features in the model. Values can range from 0 to 1. The model with the largest R-square value is chosen.

  - **AIC** — specifies Akaike’s Information Criterion. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model. Smaller values indicate better models, and AIC values can become negative. The model that has the minimal AIC value is chosen.

  - **AICC** — specifies Corrected Akaike’s Information Criterion. AICC adjusts the AIC value by accounting for sample size. Extra features penalize AICC more than AIC. Smaller values indicate better models. As the sample size increases, AICC and AIC converge.

  - **Mallows’ Cp** — specifies Mallows’s Cp value. Smaller values indicate better models, and the model with the minimal Cp is chosen.

  - **Predicted RSS** — specifies that the model that has the minimal predicted residual sum of squares (RSS) is chosen. This option is not valid with **LASSO** selection.

  - **SBC (BIC)** — specifies Schwarz Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). Unexplained variations in the response variable and the number of features increase the value of the SBC. SBC penalizes the total number of parameters more strongly than AIC and therefore tends to choose more sparse models. The model that has the minimal SBC value is chosen.
- **Validation ASE** — average square error (ASE) of the model is computed by using the validation data. The model that has the minimal validation ASE is chosen. This method requires partitioned data.

The default value is **SBC (BIC)**.

- **Entry significance level** — specifies the significance level for adding variables in the forward and stepwise directions. The default value is 0.05.

- **Stay significance level** — specifies the significance level for removing variables in backward and stepwise directions. The default value is 0.05.

- **Maximum number of effects** — specifies the maximum number of effects in any model that is considered during the selection process. If a model at some step of the selection process contains the specified maximum number of effects, then no additional effects are considered. If **Maximum number of effects** is set to 0 (the default value), this option is ignored.

- **Minimum number of effects** — specifies the minimum number of effects in any model that is considered during the backward selection process. If **Minimum number of effects** is set to 0 (the default value), this option is ignored.

- **Maximum number of steps** — specifies the maximum number of selection steps that are performed. If **Maximum number of steps** is set to 0 (the default value), this option is ignored.

- **Hierarchy** — specifies whether and how the model hierarchy requirement is applied. Model hierarchy refers to the requirement that, for any term to be in the model, all model effects that are contained in the term must be present in the model. For example, in order for the interaction A*B to enter the model, the main effects A and B must also be in the model. Likewise, neither effect A nor effect B can leave the model while the interaction A*B is in the model. Here are the possible values:
  - **All variables** — specifies that both interval and nominal variables are subject to the model hierarchy requirement.
  - **Class variables** — specifies that only nominal variables are subject to the model hierarchy requirement.
  - **None** — specifies that the hierarchy requirement is never applied.

The default value is **None**.

---

**Model Interpretability**

Specifies the method of model interpretation in order to show how the model responds to changing inputs.

- **Explain Model Inputs** — specifies the following parameters for the LIME algorithm:
  - **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.
  - **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.
Cluster-based ICE plots — the maximum number of variables (1-10).

- Explain Individual Predictions — specifies whether to include Number of clusters, Number of perturbed instances, and Number of iterations. You can specify the Distance metric as either Cosine or K prototype. You can specify the Kernel width and the Number of inputs to use in the explanation. You can specify whether to Create a score code for assigning clusters.

Linear Regression Results

After running the node, you can open the Results window by right-clicking the node and selecting Results from the pop-up menu.

Node

- t Values by Parameter — displays bar charts for the t values in the final model. The bars are color-coded to indicate the algebraic signs of the coefficients.

- Parameter Estimates — displays a table of the various statistics related to estimates for the parameters. These statistics include the t value, sign, estimate, absolute estimate, the p-value, chi-square value, and standard error.

- Selection Summary — displays a table of the iterations, adding in and removing effects. This table also includes the SBC and optimal SBC values.

- Regression Fit Statistics — displays a table of data on the linear regression fit statistics. The table includes the statistics, a corresponding qualitative description, and the values.

- Node Score Code — displays the SAS score code that was created by the node. The SAS score code can be used outside of the Model Studio environment to score new data.

- Score Inputs — displays a table of data on the input variables for scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.

- Score Outputs — displays a table of data on the predicted response variable for scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.

- Path Score Code — displays the SAS score code that was created by the node, as well as all preceding nodes. The SAS model score code can be used outside of the Model Studio environment to score new data.

- Train Code — displays the SAS code that Model Studio used to train the node.

- Properties — specifies the various properties selected before running the node.

- Output — displays the SAS output of the linear regression run.

Assessment

- Predicted Reports — displays the predicted and target mean as a function of the depth for the model. The predicted and target mean are given for each of the data roles. To examine the predicted mean as a function of the target mean, use the drop-down menu in the upper right corner. This result is displayed only if the target is an interval variable.
- **Fit Statistics** — displays a table of the fit statistics for the model, broken down by data role.

**Model Interpretability**

- **Input Relative Importance** — displays a table of the relative importance of the input variables, including variable level and label.

- **Partial Dependence** — displays the functional relationship between the input variable and the model prediction.

- **LIME Plots** — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed around a particular observation to create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when **Explain Individual Predictions** is selected.
  - **Local Instances** — displays the values for each input variable of the cluster center.
  - **Local Explanation** — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.
  - **Local Model** — displays the parameter estimates of the local linear model that is used to predict the cluster center.
  - **Prediction** — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

- **LIME Score Code** — displays the SAS code that Model Studio used to assign observations to clusters. This is displayed only when **Create score code for assigning clusters** is selected.

- **Individual Conditional Expectation** — displays the functional relationship between the input variable and the model prediction. The relationships for all cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.
Overview of Logistic Regression

The Logistic Regression node is a Supervised Learning node. A logistic regression attempts to predict the value of a binary response variable. A logistic regression analysis models the natural logarithm of the odds ratio as a linear combination of the explanatory variables. This approach enables the logistic regression model to approximate the probability that an individual observation belongs to the level of interest.

Logistic Regression Properties

General Properties
- Binary target link function — specifies the link that links the binary target to a linear function of input variables. Here are the possible values:
  - Complementary log-log
  - Log-log
  - Logit
- **Probit**
  The default value is **Logit**.

- **Nominal target link function** — specifies the link function that links the nominal target to a linear function of input variables. Cumulative link functions treat the target as an ordinal variable. Here are the possible values:
  - **Cumulative complementary log-log**
  - **Cumulative log-log**
  - **Cumulative logit**
  - **Cumulative probit**
  - **Generalized logit**
  The default value is **Generalized logit**.

---

### Effects Options

- **Class input order** — specifies the sort order for the class inputs. Here are the possible values:
  - **Formatted**
  - **Formatted descending**
  - **Frequency**
  - **Frequency descending**
  - **Unformatted**
  - **Unformatted descending**
  The default value is **Formatted**.

- **Two-factor interactions** — specifies whether to include all two-factor interactions of nominal variables in the model. By default, this option is deselected.

- **Factor split** — specifies whether levels of a nominal variable enter or leave a model independently. This is done automatically for LASSO selection. For other selection methods, this option is deselected.

- **Spline** — specifies whether to expand interval inputs into cubic B-spline bases with three equally spaced knots. (This action yields seven design matrix columns for each of the variables.) Spline and polynomial effects cannot be specified together. By default, this option is deselected.

- **Spline split** — specifies whether each individual column in the design matrix that corresponds to the spline effect is treated as a separate effect that can enter or leave the model. By default, this option is deselected. This option is available only if **Spline** is selected.

- **Polynomial** — specifies whether to use model polynomial effects up to the specified polynomial degree for all interval variables. Spline and polynomial effects cannot be specified together. By default, this option is deselected.

- **Polynomial degree** — specifies the polynomial degree when polynomial terms are included in the model. This option is available only if **Polynomial** has been selected. Possible values are 2 and 3. The default value is 2.
- **Suppress intercept** — specifies whether to suppress the intercept. By default, this is deselected.

- **Model missing values for inputs** — specifies whether to model missing values for inputs by including indicator values in the model for missing values for interval inputs and imputing missing values in the original effect with the mean as well as treating missing values for class inputs as valid levels. By default, this is deselected.

- **Use missing as level** — specifies whether missing values should be treated as a separate level for all nominal inputs. By default, this is deselected.

### Selection Method Property

- **Selection method** — specifies a model selection method. Here are the possible values:
  - **Backward** — specifies that training is done by least squares regression with all candidate effects in the model. Features are removed until the stop criterion is met.
  - **Fast backward** — specifies that training is done by least squares regression that starts with all effects in the model. Features are deleted without refitting the model.
  - **Forward** — specifies that training is done by least squares regression that starts with no candidate effects in the model. Effects are added until the Entry significance level or the stop criterion is met.
  - **LASSO** — specifies that training is done using the group LASSO method, adding and removing effects by a sequence of LASSO steps.
  - **None** — specifies not to perform variable selection.
  - **Stepwise** — specifies that training is done by least squares regression. The training starts as in the forward model, but it might remove effects already in the model.

The default value is **Stepwise**.

### Selection Options

- **Effect-selection criterion** — specifies the criterion that the procedure uses to determine the order in which effects enter or leave at each step of the selection method. Here are the possible values:
  - **AIC** — specifies Akaike’s Information Criterion. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model. Smaller values indicate better models, and AIC values can become negative.
  - **AICC** — specifies Corrected Akaike’s Information Criterion. AICC adjusts the AIC value by accounting for sample size. Extra features penalize AICC more than AIC. As the sample size increases, AICC and AIC converge.
- **SBC (BIC)** — specifies Schwarz Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). Unexplained variations in the response variable and the number of features increase the value of the SBC. Smaller values indicate better models. SBC penalizes total number of model parameters more strongly than AIC and therefore tends to choose more sparse models.

- **Significance level** — specifies the standard statistical significance level criterion.

If the Selection method above is LASSO, this option is unavailable. The default value is SBC (BIC).

- **Selection-process stopping criterion** — specifies the criterion to stop the selection process. Here are the possible values:
  - **AIC** — specifies Akaike’s Information Criterion. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model. Smaller values indicate better models, and AIC values can become negative. Selection stops at the step where AIC starts to increase.
  - **AICC** — specifies Corrected Akaike’s Information Criterion. AICC adjusts the AIC value by accounting for sample size. Extra features penalize AICC more than AIC. Smaller values indicate better models. As the sample size increases, AICC and AIC converge. Selection stops at the step where AICC starts to increase.
  - **SBC (BIC)** — specifies Schwarz Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). Unexplained variations in the response variable and the number of features increase the value of the SBC. SBC penalizes the total number of model parameters more strongly than AIC and therefore tends to choose more sparse models. Selection stops at the step where SBC starts to increase.
  - **Significance level** — specifies the standard statistical significance level criterion. Selection stops at the step where significance level exceeds the specified level. The default value is 0.05.
  - **Validation ASE** — average square error (ASE) of the model is computed by using the validation data, and selection stops at the step where the validation ASE starts to increase. This method requires partitioned data. The default value is SBC (BIC).

- **Model-selection criterion** — specifies the criterion to choose the model (from the list of models at each step of the selection process) that yields the best value of the specified criterion. If the optimal value of the specified criterion occurs for models at more than one step, then the model that has the smallest number of parameters is chosen. Here are the possible values:
  - **AIC** — specifies Akaike’s Information Criterion. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model. Smaller values indicate better models, and AIC values can become negative. The model that has the minimal AIC value is chosen.
  - **AICC** — specifies Corrected Akaike’s Information Criterion. AICC adjusts the AIC value by accounting for sample size. Extra features penalize AICC more than AIC. As the sample size increases, AICC and AIC converge.
  - **SBC (BIC)** — specifies Schwarz Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). Unexplained variations in the
response variable and the number of features increase the value of the SBC. SBC penalizes total number of model parameters more strongly than AIC and therefore tends to choose more sparse models. The model that has the minimal SBC value is chosen.

- **Validation ASE** — average square error (ASE) of the model is computed by using the validation data, and the model that has the minimal validation ASE is chosen. This method requires partitioned data.

The default value is **SBC (BIC)**.

- **Entry significance level** — specifies the significance level for adding variables in the forward and stepwise directions. The default value is 0.05.
- **Stay significance level** — specifies the significance level for removing variables in backward and stepwise directions. The default value is 0.05.
- **Maximum number of effects** — specifies the maximum number of effects in any model that is considered during the selection process. If a model at some step of the selection process contains the specified maximum number of effects, then no additional effects are considered. If **Maximum number of effects** is set to 0 (the default value), this option is ignored.
- **Minimum number of effects** — specifies the minimum number of effects in any model that is considered during the backward selection process. If **Minimum number of effects** is set to 0 (the default value), this option is ignored.
- **Maximum number of steps** — specifies the maximum number of selection steps that are performed. If **Maximum number of steps** is set to 0 (the default value), this option is ignored.
- **Hierarchy** — specifies whether and how the model hierarchy requirement is applied. Model hierarchy refers to the requirement that, for any term to be in the model, all model effects that are contained in the term must be present in the model. For example, in order for the interaction A*B to enter the model, the main effects A and B must also be in the model. Likewise, neither effect A nor effect B can leave the model while the interaction A*B is in the model. Here are the possible values:
  - **All variables** — specifies that both interval and nominal variables are subject to the model hierarchy requirement.
  - **Class variables** — specifies that only nominal variables are subject to the model hierarchy requirement.
  - **None** — specifies that the hierarchy requirement is never applied.

The default value is **None**.

---

### Optimization Options

- **Optimization technique** — specifies the optimization method used when fitting a model. Here are the possible values:
  - **Conjugate-gradient**
  - **Double-dogleg**
  - **Dual quasi-Newton**
  - **Nelder-Mead simplex**
Newton-Raphson
Newton-Raphson with ridging
None
Trust-region

The default value is **Newton-Raphson with ridging**. For more information, see the PROC NLMIXED documentation in SAS/STAT 14.3 User’s Guide.

**Note:** Optimization options are not available for the LASSO selection method. LASSO uses the Nesterov algorithm as the default optimization technique.

- **Maximum number of iterations** — specifies the maximum number of iterations of any optimization. The default value depends on the optimization technique used:

<table>
<thead>
<tr>
<th>Optimization Technique</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjugate-gradient</td>
<td>400</td>
</tr>
<tr>
<td>Double-dogleg</td>
<td>200</td>
</tr>
<tr>
<td>Dual quasi-Newton</td>
<td>200</td>
</tr>
<tr>
<td>Nelder-Mead simplex</td>
<td>1000</td>
</tr>
<tr>
<td>Newton-Raphson</td>
<td>50</td>
</tr>
<tr>
<td>Newton-Raphson with ridging</td>
<td>50</td>
</tr>
<tr>
<td>Trust-region</td>
<td>50</td>
</tr>
</tbody>
</table>

To use the default value, leave **Maximum number of iterations** blank or use a dot.

- **Maximum number of function evaluations** — specifies the maximum number of function calls of any optimization. The default value depends on the optimization technique used:

<table>
<thead>
<tr>
<th>Optimization Technique</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjugate-gradient</td>
<td>1000</td>
</tr>
<tr>
<td>Double-dogleg</td>
<td>500</td>
</tr>
<tr>
<td>Dual quasi-Newton</td>
<td>500</td>
</tr>
<tr>
<td>Nelder-Mead simplex</td>
<td>3000</td>
</tr>
<tr>
<td>Newton-Raphson</td>
<td>125</td>
</tr>
<tr>
<td>Newton-Raphson with ridging</td>
<td>125</td>
</tr>
<tr>
<td>Trust-region</td>
<td>125</td>
</tr>
</tbody>
</table>
To use the default value, leave **Maximum number of function evaluations** blank or use a dot.

- **Maximum CPU time in second** — specifies an upper limit of CPU time (in seconds) for the optimization process. The default value is the largest floating-point double representation of your computer. To use the default value, leave **Maximum CPU time in second** blank or use a dot.

- **Minimum number of iterations** — specifies the minimum number of iterations in any optimization. The default value is 1. To use the default value, leave **Minimum number of iterations** blank or use a dot.

- **Normalize objective function** — selects whether the objective function should be normalized during optimization by the reciprocal of the used frequency count. By default, this option is selected.

---

**Convergence Options**

- **Absolute function convergence** — specifies the threshold for absolute function convergence. The default value is the negative square root of the largest double-precision value. To use the default value, leave **Absolute function convergence** blank or use a dot.

- **Absolute function difference convergence** — specifies the threshold for absolute function difference convergence. The default value is 0. To use the default value, leave **Absolute function difference convergence** blank or use a dot.

- **Absolute gradient convergence** — specifies the threshold for absolute gradient convergence. The default value is 1E-5. To use the default value, leave **Absolute gradient convergence** blank or use a dot.

- **Relative function difference convergence** — specifies the relative function difference convergence. The default value is the machine precision times 2. To use the default value, leave **Relative function difference convergence** blank or use a dot.

- **Relative gradient convergence** — specifies the threshold for relative gradient convergence. The default value is 1E-8. To use the default value, leave **Relative gradient convergence** blank or use a dot.

---

**Binary Classification Cutoff**

- **Specify node binary classification cutoff** — specifies whether to use the binary classification cutoff specified below for the node. If this option is deselected, the project binary classification cutoff is used for determining the predicted value for a binary target based on the posterior probabilities. By default, this value is deselected.

- **Node binary classification cutoff** — specifies the cutoff to use in the node for determining the predicted value for a binary target based on the posterior probabilities. This option is available only if **Specify node binary classification cutoff** is selected. The default value is 0.5.
Model Interpretability

Specifies the method of model interpretation in order to show how the model responds to changing inputs.

- **Explain Model Inputs** — specifies the following parameters for the LIME algorithm:
  - **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.
  - **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.
  - **Cluster-based ICE plots** — the maximum number of variables (1-10).

- **Explain Individual Predictions** — specifies whether to include **Number of clusters**, **Number of perturbed instances**, and **Number of iterations**. You can specify the **Distance metric** as either Cosine or K prototype. You can specify the **Kernel width** and the **Number of inputs to use in the explanation**. You can specify whether to **Create a score code for assigning clusters**.

Logistic Regression Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Node**

- **t Values by Parameter** — displays bar charts for the t values in the final model. The bars are color-coded to indicate the algebraic signs of the coefficients.

- **Parameter Estimates** — displays a table of the various statistics related to estimates for the parameters. These statistics include the t value, sign, estimate, absolute estimate, the p-value, chi-square value, and standard error.

- **Selection Summary** — displays a table of the iterations, adding in and removing effects. This table also includes the SBC and optimal SBC values.

- **Regression Fit Statistics** — displays a table of data on the logistic regression fit statistics. The table includes the statistic, a corresponding qualitative description, and the training, validation, and testing values.

- **Node Score Code** — displays the SAS score code that was created by the node. The SAS score code can be used outside of the Model Studio environment to score new data.

- **Score Inputs** — displays a table of data on the input variables for scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.
Score Outputs — displays a table of data on the predicted response variable for scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.

Path Score Code — displays the SAS score code that was created by the node, as well as all preceding nodes. The SAS model score code can be used outside of the Model Studio environment to score new data.

Train Code — displays the SAS code that Model Studio used to train the node.

Properties — specifies the various properties selected before running the node.

Output — displays the SAS output of the logistic regression run.

Assessment

Lift Reports — displays the cumulative lift as a function of the depth for the model. The cumulative lift is given for each of the data roles. To examine other statistics as a function of depth, use the drop-down menu in the upper right corner. Other statistics include lift, gain, % captured response, cumulative % captured response, % response, and cumulative % response. This result is displayed only if the target is a class variable.

ROC Reports — displays the ROC (receiver operating characteristic) chart for a model, giving the sensitivity as a function of 1-specificity. The sensitivity is given for each of the data roles. To examine other statistics, use the drop-down menu in the upper right corner. Other statistics include accuracy and F1 score. This result is displayed only if the target is a class variable.

Fit Statistics — displays a table of the fit statistics for the model, broken down by data role.

Model Interpretability

Input Relative Importance — displays a table of the relative importance of the input variables, including variable level and label.

Partial Dependence — displays the functional relationship between the input variable and the model prediction.

LIME Plots — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed around a particular observation to a create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when Explain Individual Predictions is selected.

- Local Instances — displays the values for each input variable of the cluster center.
- Local Explanation — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.
- Local Model — displays the parameter estimates of the local linear model that is used to predict the cluster center.
- Prediction — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

LIME Score Code — displays the SAS code that Model Studio used to assign observations to clusters. This is displayed only when Create score code for assigning clusters is selected.

Individual Conditional Expectation — displays the functional relationship between the input variable and the model prediction. The relationships for all
cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.
Overview of Manage Variables

The Manage Variables node is a preprocessing node that enables you to make modifications to the metadata while it is within a Model Studio pipeline. The options available to you are a subset of the options available under the Data tab. For example, you cannot modify the target variable attributes, or change the target variable. For more information about the Data tab, see Data Management Overview in the Model Studio User’s Guide.

Manage Variables Properties

To manage variables, place the Manage Variables node on the diagram, right-click the node, and select Run. Let the node run, right-click the node again, and then select the Manage Variables option. This opens a window that lists the set of variables that you can edit (change the role, level, order, and so on). After editing a variable, save the edit in the upper right corner, and click Close.

Manage Variables Results

After running the node or editing the metadata, you can open the Results window by right-clicking the node and selecting Results from the pop-up menu.

Results
- Incoming Variables — displays a list of the incoming variables and their attributes from the data set, including input and target variables. This list also includes summary statistics.
- **Metadata Changes** — displays a list of the attribute alterations made to the variables. Only variables that have been changed will show up here.

- **Properties** — specifies a list of the properties used by the node. Note that these are not set by you. They are static properties.

- **Output** — displays the SAS output of the manage variables run.
Overview of Model Comparison

The Model Comparison node is a Model Studio node that is automatically added to a pipeline when a Supervised Learning node is also added. The Model Comparison node enables you to compare the performance of competing models using various benchmarking criteria.

There are many assessment criteria that can be used to compare models. For class targets, these include measures of error, lift-based measures, and measures derived from the Receiver Operating Characteristic (ROC) curve. You can select the measure, and specify the depth to use when applying a lift-based measure or the cutoff to use when applying an ROC-based measure. For interval targets, there are various measures of error available for choosing the champion model. All measures of assessment are computed for each of the data partitions that are available (train, validate, and test). You can also select which data partition to use for selecting the champion.

Note: If multiple supervised learning nodes are connected to the Model Comparison node, then only successfully completed models will be compared. Models that have failed or been stopped are not considered. The selected model from the Model Comparison node of each pipeline is added to the Pipeline Comparison tab. This enables you to compare models from the different pipelines in your project and to select a champion model. To add models that were not selected by the Model Comparison node to the Pipeline Comparison tab, right-click on the given model and select Add challenger model.
Model Comparison Properties

General Properties

- **Class selection statistic** — specifies the fit statistic to use for selecting the champion model when there is a class target. Here are the possible values:
  - Accuracy
  - Area under curve (C statistic)
  - Average squared error
  - Captured response
  - Cumulative lift
  - Default
  - F1 score
  - False discovery rate
  - False positive rate
  - Gain
  - Gini
  - Kolmogorov-Smirnov statistic (KS)
  - Lift
  - Misclassification (Event)
  - Misclassification (MCE)
  - Multiclass log loss
  - ROC separation
  - Root average squared error

If **Default** is selected (it is selected by default), the class selection statistic chosen in the **Project Settings** menu in the upper right corner is used. In **Project Settings**, the default is **Kolmogorov-Smirnov statistic (KS)**.

**Note:** **Misclassification (MCE)** is the true misclassification rate. That is, every observation where the observed target level is predicted to be a different level counts in the misclassification rate. **Misclassification (Event)** considers only the classification of the event level versus all other levels. Thus, a non-event level classified as another non-event level does not count in the misclassification. For binary targets, these two measures are the same. The **Misclassification (Event)** statistic is computed in the context of the ROC report. That is, at each cutoff value, this measure is calculated.

For more information, see the **PROC ASSESS** documentation in SAS Visual Statistics 8.3: Procedures Guide.
Interval selection statistic — specifies the fit statistic to use for selecting the champion model when there is an interval target. Here are the possible values:

- Average squared error
- Default
- Root average squared error
- Root mean absolute error
- Root mean squared logarithmic error

If Default is selected (it is selected by default), the interval selection statistic chosen in the Project Settings menu in the upper right corner is used. The default in Project Settings is Average squared error.

For more information, see the PROC ASSESS documentation in SAS Visual Statistics 8.3: Procedures Guide.

Selection partition — specifies from which of the data roles the fit statistic is used for selecting the champion model. Here are the possible values:

- Default
- Test
- Train
- Validate

If Default is selected (it is selected by default), the selection partition chosen in the Project Settings menu in the upper right corner is used. The default in Project Settings is Validate.

Selection Depth — specifies the depth to use when there is a class target. This option is available for the following class selection statistics:

- Lift
- Cumulative Lift
- Gain
- Gini
- Captured response
- Cumulative captured response

Here are the possible values:

- 5
- 10
- 15
- 20
- Default

If Default is selected (it is by default), the selection depth chosen in the Project Settings menu in the upper right corner is used. The default value in Project Settings is 10.

ROC-based cutoff — specifies the posterior probability cutoff to use for classifying a binary target when there is a class target. This option is available for the following class selection statistics:

- F1 score
Model Comparison Results

After running the node, you can open the Results window by right-clicking the node and selecting Results from the pop-up menu.

**Node**

- **Model Comparison** — displays a table of the results of the model comparison. The table includes which model is the selected model, as well as a list of the various comparison statistics. To change which comparison statistics are displayed, click the Options tab found in the right-most column, and select Manage Columns. You can then select a hidden column and move it to the displayed section, or select a displayed column and move it to the hidden section.

  The Model Comparison table is a combination of the fit statistics table, the lift statistics at the specified depth, and the ROC-based statistics at the specified cutoff value for all models being compared.

- **Properties** — displays a table of the properties used to select the best model. This includes the property names and values.

**Assessment**

- **Predicted Reports** — displays the predicted mean of the target as a function of the actual target mean for each model and each of the data roles. To examine the predicted mean and the actual target mean both as a function of depth, use the drop-down menu in the upper right corner. This result is displayed only if the target is an interval variable.

- **Lift Statistics (for class target)** — displays the % response as a function of the depth for the models. To examine other statistics as a function of depth, use the drop-down menu in the upper right corner. Other statistics include cumulative lift, lift, gain, % captured response, cumulative % captured response, and cumulative % response.

- **ROC Statistics (for class target)** — displays accuracy as a function of the cutoff for the models and each of the data roles. You can also examine the F1 score as a function of the cutoff or the ROC chart that displays the sensitivity as a function of 1-specificity. To examine these other statistics, use the drop-down menu in the upper right corner.

- **Fit Statistics** — displays the fit statistics for the models, broken down by data role. To change which comparison statistics are displayed, click the Options tab
found in the right-most column, and select Manage Columns. You can then select a hidden column and move it to the displayed section, or select a displayed column and move it to the hidden section.
Overview of Neural Network

The Neural Network node is a Supervised Learning node. A neural network is a statistical model that is designed to mimic the biological structures of the human brain. Neural networks consist of predictors (input variables), hidden layers, a target (or output) layer, and the connections between each of those. This node trains a fully connected, multilayer perceptron neural network with up to 10 hidden layers. Predictors can also be connected directly to the output layer. You can specify an activation function for each hidden layer and the target layer.

Neural Network Properties

General Properties

- **Include missing inputs** — specifies whether to impute missing values with the mean for interval inputs and to treat missing as a level for class inputs. By default, this is deselected.
- **Input standardization** — specifies the method that is used to standardize the interval input variables. Here are the possible values:
  - Midrange
  - None
  - Z score
  The default value is **Midrange**.
- **Number of hidden layers** — specifies the number of hidden layers to include in the neural network. Possible values range from 0 to 10. The default value is 1.

### Hidden Layer Options
- **Use same number of neurons in hidden layers** — specifies to use the same number of neurons in each hidden layer. By default, this is selected. If deselected, the **Custom Hidden Layer Options** menu enables you to vary the number of neurons in each hidden layer.
- **Number of neurons per hidden layer** — specifies the number of neurons in each hidden layer. The default value is 50.
- **Hidden layer activation function** — specifies the activation function to use in the hidden layers. Here are the possible values:
  - Exponential
  - Identity
  - Logistic
  - ReLU
  - Sine
  - Softplus
  - Tanh
  - Varies
  Choosing **Varies** enables you to vary the activation function in each hidden layer, using the **Custom Hidden Layer Options** menu. The default value is **Tanh**.

### Target Layer Options
- **Direct connections** — specifies whether direct connections from nodes in the input layer to nodes in the output layer should be included in the network. By default, this is deselected.
- **Interval target standardization** — specifies the method that is used to standardize the interval target variable. Here are the possible values:
  - Midrange
  - None
- **Z score**
  The default value is **Midrange**.

- **Interval target error function** — specifies the target layer error function for interval targets. When there are no hidden layers, the normal distribution is used. Here are the possible values:
  - **Gamma**
  - **Normal**
  - **Poisson**

  The default value is **Normal**.

- **Interval target activation function** — specifies the target layer activation function for interval targets. When there are no hidden layers, the identity function is used. Here are the possible values:
  - **Identity**
  - **Sine**
  - **Tanh**

  If the **Interval target error function** is **Gamma** or **Poisson**, this option is unavailable, and the Exponential activation function is used. The default value is **Identity**.

### Common Optimization Options

- **Optimization method** — specifies the optimization method to use to train the network. This option is only in effect for networks with less than 6 hidden layers.
  - **Automatic** — Selects the optimization method based on the number of hidden layers. LBFGS is used when there are 2 or less hidden layers. SGD is used otherwise.
  - **LBFGS**
  - **SGD**

  LBFGS cannot be used in networks with more than 2 hidden layers. The default value is **Automatic**.

- **Number of tries** — specifies the number of times to train the network. This is done by using different initial estimates for the weights. Possible values range from 1 to 64. The default value is 1.

- **Maximum iterations** — specifies the maximum number of iterations allowed within each try. The default value is 300.

- **Maximum time** — specifies the maximum time (in minutes) allowed for optimizations. When this value is reached, the optimization terminates the search and returns the results. The value of 0 indicates that no maximum time is set. The default value is 0.

- **Random seed** — specifies the random seed to use for generating random numbers to initialize the network weights. The default value is 12345.

- **L1 weight decay** — specifies the weight decay for L1 regularization. The default value is 0.
- **L2 weight decay** — specifies the weight decay for L2 regularization. The default value is 0.1.
- **Hidden layer dropout ratio** — specifies the dropout ratio for the hidden layers when SGD optimization is used. The default value is 0.
- **Input layer dropout ratio** — specifies the dropout ratio for the input layer when SGD optimization is used. The default value is 0.

### SGD Options (≥3 Hidden Layers)

*SGD Options* — specifies the configurations for stochastic gradient descent (SGD). This menu is available only if there are 3 or more hidden layers.

- **Learning rate** — specifies the learning rate parameter for SGD optimization. The default value is 0.001.
- **Annealing rate** — specifies the annealing rate parameter for SGD optimization. The default value is 0.000001.
- **Minibatch size** — specifies the size of the minibatches used for SGD optimization. The default value is 50.
- **Momentum** — specifies the value for momentum for SGD optimization. The default value is 0.
- **Create deterministic results** — specifies whether to create deterministic (reproducible) results using the specified SGD seed. Note that selecting this option can significantly increase run time. By default, this option is deselected.
- **SGD seed** — specifies the seed to use for SGD optimization to create reproducible results. The default value is 12345.

For neural networks with more than 5 hidden layers, the SGD options are different. The following options are available:

- **Learning rate** — specifies the learning rate parameter for SGD optimization. The default value is 0.001.
- **Maximum epochs** — specifies the maximum number of epochs. One epoch is reached after a single pass through the data. The default value is 10.
- **Minibatch size** — specifies the size of the minibatches used for SGD optimization. The default value is 50.

### Advanced Options

- **SGD algorithm** — specifies the SGD algorithm to use when training more than 5 hidden layers. Possible values are as follows:
  - **Adam SGD**
  - **Momentum SGD**
  - **SGD**

  The default value is *Adam SGD*.

- **First moment exponential decay rate** — specifies the exponential decay rate for the first moment in the Adam learning algorithm. This option is available only if **SGD algorithm** is *Adam SGD*. The default value is 0.9.
− Second moment exponential decay rate — specifies the exponential decay rate for the second moment in the Adam learning algorithm. This option is available only if SGD algorithm is Adam SGD. The default value is 0.999.

− Momentum — specifies the momentum value for SGD optimization. This option is available only if SGD algorithm is Momentum SGD. The default value is 0.9.

− Minimum gradient value — specifies the minimum gradient value. All gradients that are less than the specified value are set to the specified value. The default value is 0.

− Maximum gradient value — specifies the maximum gradient value. All gradients that are greater than the specified value are set to the specified value. The default value is 0.

− Learning rate policy — specifies the learning rate policy. Here are the possible values:
  - Fixed decay
  - Inverse decay
  - Polynomial decay
  - Step down

− Gamma — specifies the gamma value for the learning rate policy. This option is available only if Learning rate policy is Inverse decay or Step down. The default value is 0.1.

− Power — specifies the power for the learning rate policy. This option is available only if Learning rate policy is Inverse decay or Polynomial decay. The default value is 0.75.

− Step size — specifies the step size for the learning rate policy. This option is available only if Learning rate policy is Step down. The default value is 10.

Perform Early Stopping

Perform Early Stopping — specifies whether to stop training when the model begins to overfit. The training stops after \( N \) consecutive iterations (Stagnation) without improvement in the validation partition. Early stopping cannot be used if there is no validation partition. By default, this option is selected. The following options are available:

− Stagnation — specifies the number of consecutive iterations (\( N \)) for early stopping. The default value is 5.

− Validation error goal — specifies a goal for early stopping based on the validation error rate. When the error gets below this value, the optimization will stop. This option is in effect only for networks with fewer than 6 hidden layers. The value of 0 indicates that no validation error goal is set. The default value is 0.
Perform Autotuning

This feature specifies whether to perform autotuning of any neural network parameters.

**Note:** Autotuning is available only when **Number of hidden layers** is less than 6.

Warning: Performing autotuning can substantially increase run time. If **Perform Autotuning** is selected, the following options are available:

- **Number of Hidden Layers** — specifies whether to autotune the number of hidden layers. If selected, the following option is available:
  - **Hidden layers initial value** — specifies the initial value for autotuning the number of hidden layers. The default value is 1. Use the **From** and **To** options to specify the range. The default **From** value is 0, and the default **To** value is 2.
  - **Neurons initial value** — specifies the initial value for autotuning the number of neurons. The default value is 1. Use the **From** and **To** options to specify the range. The default **From** value is 1, and the default **To** value is 100.

- **L1 Weight Decay** — specifies whether to autotune the L1 weight decay parameter. If selected, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the L1 weight decay.
    - The default value is 0. Use the **From** and **To** options to specify the range. The default **From** value is 0, and the default **To** value is 10.

- **L2 Weight Decay** — specifies whether to autotune the L2 weight decay parameter. If selected, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the L2 weight decay.
    - The default value is 0. Use the **From** and **To** options to specify the range. The default **From** value is 0, and the default **To** value is 10.

- **Learning Rate** — specifies whether to autotune the learning rate for the hidden layers. If selected, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the hidden layer learning rate. The default value is 0.001. Use the **From** and **To** options to specify the range. The default **From** value is 0, and the default **To** value is 0.1.

- **Annealing Rate** — specifies whether to autotune the annealing rate for the hidden layers. If selected, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the hidden layer annealing rate. The default value is 0.001. Use the **From** and **To** options to specify the range. The default **From** value is 0.000001, and the default **To** value is 0.1.

- **Search Options** — specifies the options for autotuning searching. The following options are available:
  - **Search method** — specifies the tuning search method. Here are the possible values:
    - **Bayesian** — specifies the Bayesian method. This method uses priors to seed the iterative optimization.
- **Genetic algorithm** — specifies the genetic algorithm method. This method uses an initial Latin Hypercube sample that seeds a genetic algorithm to generate a new population of alternative configurations at each iteration.

- **Grid** — specifies the grid method. This method performs an exhaustive search of all configurations. Each hyperparameter of interest is discretized into a desired set of values to be studied. Models are trained and assessed for all combinations of values across all the hyperparameters (thus forming a multi-dimensional “grid”).

- **Latin hypercube sample** — specifies the Latin Hypercube method. This method performs an optimized grid search that is uniform in each tuning parameter, but random in combinations.

- **Random** — specifies the Random method. This method generates a single sample of purely random configurations.

The default value is **Genetic algorithm**.

- **Number of evaluations per iteration** — specifies the number of tuning evaluations in one iteration. This option is available only if the **Search method** is Genetic algorithm or Bayesian. The default value is 10.

- **Maximum number of evaluations** — specifies the maximum number of tuning evaluations. This option is available only if the **Search method** is Genetic algorithm or Bayesian. The default value is 50.

- **Maximum number of iterations** — specifies the maximum number of tuning iterations. This option is available only if the **Search method** is Genetic algorithm or Bayesian. The default value is 5.

- **Maximum number of points in model** — specifies the maximum number of points in the model for a Bayesian search. This option is available only if the **Search method** is Bayesian. The default value is 100.

- **Sample size** — specifies the sample size. This option is available only if the **Search method** is Random or Latin hypercube sample. The default value is 50.

- **General Options** — specifies the general properties for autotuning. The following options are available:

  - **Validation method** — specifies the validation method for finding the objective value. Note that if your data is partitioned, then that partition is used and **Validation method**, **Validation data proportion**, and **Cross validation number of folds** are all ignored. Here are the possible values:

    - **Partition** — specifies using the partition validation method. With partition, you will specify proportions to use for randomly assigning observations to each role.

    - **K-fold cross validation** — specifies using the cross validation method. In cross validation, each model evaluation requires \( k \) training executions (on \( k-1 \) data folds) and \( k \) scoring executions (on one holdout fold). This increases the evaluation time by approximately a factor of \( k \).

    For small to medium data tables, cross validation provides, on average, a better representation of error across the whole data table. **Partition** is the default value.

  - **Training data proportion** — specifies the proportion of data to be used for training the **Partition** validation method. The default value is 0.7.
- **Validation data proportion** — specifies the proportion of data to be used for the Partition validation method. The default value is 0.3.

- **Cross validation number of folds** — specifies the number of partition folds in the cross validation process (the $k$ defined above). Possible values range from 2 to 20. The default value is 5.

- **Class target objective function** — specifies the objective function to optimize for tuning parameters for a class target. Here are the possible values:
  - Area under the curve
  - Average squared error
  - F0.5 score
  - F1 score
  - Gamma
  - Gini coefficient
  - Kolmogorov-Smirnov statistic
  - Misclassification rate
  - Multi-class log loss
  - Root average squared error
  - Tau
  The default value is Kolmogorov-Smirnov statistic.

- **Interval target objective function** — specifies the objective function to optimize for tuning parameters for an interval target. Here are the possible values:
  - Average squared error
  - Mean absolute error
  - Mean squared logarithmic error
  - Root average squared error
  - Root mean absolute error
  - Root mean squared logarithmic error
  The default value is Average squared error.

- **Maximum time (minutes)** — specifies the maximum time (in minutes) for the optimization tuner. The default value is 60.

- **Maximum training time for single model (minutes)** — specifies the maximum time (in minutes) for a single model train. To indicate no maximum, enter no value in the field.

### Binary Classification Cutoff

- **Specify node binary classification cutoff** — specifies whether to use the binary classification cutoff specified below for the node. If deselected, the project
binary classification cutoff is used for determining the predicted value for a binary target based on the posterior probabilities. By default, this value is deselected.

- **Node binary classification cutoff** — specifies the cutoff to use in the node for determining the predicted value for a binary target based on the posterior probabilities. This option is available only if **Specify node binary classification cutoff** is selected. The default value is 0.5.

---

**Model Interpretability**

Specifies the method of model interpretation in order to show how the model responds to changing inputs.

- **Explain Model Inputs** — specifies the following parameters for the LIME algorithm:
  - **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.
  - **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.
  - **Cluster-based ICE plots** — the maximum number of variables (1-10).

- **Explain Individual Predictions** — specifies whether to include **Number of clusters**, **Number of perturbed instances**, and **Number of iterations**. You can specify the **Distance metric** as either **Cosine** or **K prototype**. You can specify the **Kernel width** and the **Number of inputs to use in the explanation**. You can specify whether to **Create a score code for assigning clusters**.

---

**Neural Network Results**

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

- **Node**
  - **Network Diagram: Top 200 Weights** — displays a diagram of the neural network when there are less than 6 hidden layers in the network. Only the links that correspond to the top 200 weights are displayed. You can interactively control the range of these weights to display.
  - **Iteration Plot** — displays a line graph of the validation error, if reported, as a function of the epoch. To examine the loss or objective as a function of the epoch, use the drop-down menu in the upper right corner.
  - **Autotune Best Configuration** — displays a table of the resulting values of parameters that were autotuned. This table will not be displayed if autotuning is not selected.
Autotune Results — displays the iterative results of the autotuning process, showing how each of the parameter values evolve as the model iterates. This table will not be displayed if autotuning is not selected.

Node Score Code — displays the SAS score code that was created by the node. The SAS score code can be used outside of the Model Studio environment to score new data.

Note: If you train a network with more than 5 hidden layers, you get EP score code instead of DATA step code since an analytic store is used.

Score Inputs — displays a table of data on the input variables for scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.

Score Outputs — displays a table of data on the predicted response variable for scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.

Path EP Score Code — displays the SAS score code that was created by the node when training a network with more than 5 hidden layers. The score code can be used outside of the Model Studio environment to score new data.

DS2 Package Code — displays the SAS DS2 package code that was created by the node. The score code can be used outside of the Model Studio environment to score new data.

Path Score Code — displays the SAS score code that was created by the node, as well as all preceding nodes. The SAS model score code can be used outside of the Model Studio environment to score new data.

Note: If you trained a network with more than 5 hidden layers, this code is not included in the results.

Train Code — displays the SAS code that Model Studio used to train the node.

Properties — specifies the various properties selected before running the node.

Output — displays the SAS output of the neural network run.

Assessment

Predicted Reports — displays the predicted and target mean as a function of the depth for the model. The predicted and target mean are given for each of the data roles. To examine the predicted mean as a function of the target mean, use the drop-down menu in the upper right corner. This result is displayed only if the target is an interval variable.

Lift Reports — displays the cumulative lift as a function of the depth for the model. The cumulative lift is given for each of the data roles. To examine other statistics as a function of depth, use the drop-down menu in the upper right corner. Other statistics include lift, gain, % captured response, cumulative % captured response, % response, and cumulative % response. This result is displayed only if the target is a class variable.

ROC Reports — displays the ROC (receiver operating characteristic) chart for a model, giving the sensitivity as a function of 1-specificity. The sensitivity is given for each of the data roles. To examine other statistics, use the drop-down menu in the upper right corner. Other statistics include accuracy and F1 score. This result is displayed only if the target is a class variable.

Fit Statistics — displays a table of the fit statistics for the model, broken down by data role.

Model Interpretability
- **Input Relative Importance** — displays a table of the relative importance of the input variables, including variable level and label.

- **Partial Dependence** — displays the functional relationship between the input variable and the model prediction.

- **LIME Plots** — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed around a particular observation to create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when **Explain Individual Predictions** is selected.
  - **Local Instances** — displays the values for each input variable of the cluster center.
  - **Local Explanation** — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.
  - **Local Model** — displays the parameter estimates of the local linear model that is used to predict the cluster center.
  - **Prediction** — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

- **LIME Score Code** — displays the SAS code that Model Studio used to assign observations to clusters. This is displayed only when **Create score code for assigning clusters** is selected.

- **Individual Conditional Expectation** — displays the functional relationship between the input variable and the model prediction. The relationships for all cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.
Overview of Open Source Code

The Open Source Code node is a Miscellaneous node that can run Python or R code. This node can subsequently be moved to the Supervised Learning group if a Python or R model needs to be assessed and included in the Model Comparison node.

Because this code is not executed in CAS, a sample of the data is created and downloaded to avoid the movement of large data. The default sample size is 10,000 observations. Use the Data Sample properties to control the sample size and method. Apply caution and do not specify the entire data table or a relatively large sample when the input data is large. When you use the Model Comparison node to compare the results of your Python or R modes note that this node might not be using the entire input data set.

Input data is accessed by the Python or R code via a comma-separated value (CSV) file or as a data frame. When Generate data frame is selected, a data frame is generated from the CSV. In this case, the input data is available in dm_inputdf, which is a pandas data frame for Python code or an R data frame for R code. When the data is partitioned, an additional data frame, dm_traindf, is available in the editor. This data frame contains the training data. If a Python or R model is built and needs to be assessed, the corresponding predictions or posterior probabilities are made available in the dm_scoredf data frame.

The Open Source Code node can be moved from the Miscellaneous group to the Supervised Learning group. To move the node, right-click the Open Source Code node and select Move to Supervised Learning. Do this to indicate that your Open Source Code node contains a model that should be merged with the input data and that model assessment should be performed. To successfully merge your Python or
R model, the number of observations in dm_inputdf and dm_scoreddf must be equal.

For an example using the Open Source Code node, see Open Source Code Node Example.

Note: This node does not support the operations Download score code, Register models, Publish models, Score holdout data, and other similar operations in the Pipeline Comparison tab. This node cannot be a member of the Ensemble node when used in a Supervised Learning group. These operations are not supported because the Open Source Code node does not generate SAS score code.

Note: The Open Source Code node cannot be run if lockdown is enabled or if you are working in a multi-tenant environment.

Open Source Code Properties

- **Code editor** — invokes the Open Source Code Editor. The next section will provide documentation on the Open Source Code Editor.
- **Language** — specifies the language of your code.
- **Generate data frame** — specifies whether data frames are created.
- **Data Sample**
  - **Sampling method** — specifies the method used to create the data sample. Options include None, Simple random, and Stratify. By default, Simple random is selected.
  - **Sample using** — specifies whether sampling is performed on a fixed number of observations or a fixed percentage of observations
  - **Number of observations** — specifies the fixed number of observations used in the sample.
  - **Percentage of observations** — specifies the fixed percentage of observations used in the sample.
  - **Include SAS formats** — specifies whether SAS formats are used in the data sample.

Open Source Code Editor User Interface

The following data items are defined and available for certain expected cases in your Python or R code. For R code, the setwd(dm_nodedir) function call is also added so that you can specify dm_nodedir as the current working directory.
<table>
<thead>
<tr>
<th>Data item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dm_nodedir</td>
<td>A variable that identifies the node’s working directory. That is, where the local files are created. The data type is string.</td>
</tr>
<tr>
<td>dm_dec_target</td>
<td>A variable that identifies the name of the target. The data type is string.</td>
</tr>
<tr>
<td>dm_partitionvar</td>
<td>A variable that identifies the name of the data partition. The data type is string.</td>
</tr>
<tr>
<td>dm_partition_train_val</td>
<td>A variable that identifies the value that identifies the training partition. The data type is string or integer.</td>
</tr>
<tr>
<td>dm_class_input</td>
<td>A list of names that identify the categorical (nominal and ordinal) variables. The data type is a list in Python and a vector in R.</td>
</tr>
<tr>
<td>dm_interval_input</td>
<td>A list of names that identify the interval variables. The data type is a list in Python and a vector in R.</td>
</tr>
<tr>
<td>dm_input</td>
<td>A list of names that identify the input (interval and categorical) variables. The data type is a list in Python and a vector in R.</td>
</tr>
<tr>
<td>dm_model_formula</td>
<td>An object that identifies the model formula in R of the form ~ input_var1 + input_var2. The data type is formula object. This object is available only in R.</td>
</tr>
<tr>
<td>dm_model</td>
<td>An object that identifies the mode. The usage of this object is optional but recommended for future use.</td>
</tr>
<tr>
<td>dm_inputdf</td>
<td>A data frame that contains the sampled input data. This is available only when the Generate data frame property is selected. The data type is a pandas data frame in Python or a data frame where categorical variables are encoded as factors in R.</td>
</tr>
<tr>
<td>dm_traindf</td>
<td>A data frame that contains the sampled training data. This is available only when the Generate data frame property is selected. The data type is a pandas data frame in Python or a data frame where categorical variables are encoded as factors in R.</td>
</tr>
<tr>
<td>dm_scoreddf</td>
<td>A data frame that you must create with model predictions when you execute the node in the Supervised Learning group. This item must be created when the Generate data frame property is selected and contain the same number of rows as dm_inputdf. The data type is a pandas data frame in Python or a data frame in R.</td>
</tr>
</tbody>
</table>
Data item | Description
--- | ---
node_data.csv | A file in comma-separated value format that identifies the sampled input data in the dm_nodedir directory.
node_scored.csv | A file in comma-separated value format that you must create with model predictions when you execute the node in the *Supervised Learning* group and model assessment is performed. This file must be created when *Generate data frame* is not selected and must contain the same number of rows as node_data.csv.

Open Source Code Results

After running the node, you can open the Results window by right-clicking the node and selecting *Results* from the pop-up menu.

The following results are available:

- Plots, tables, or an editor that display the output from the open-source program. These are displayed when the open-source program creates output files with the *rpt_* prefix and a csv, txt, jpg, jpeg, png, or GIF file extension. For example, *rpt_ForestVariableImportance.csv* will display a variable importance table. For a CSV file, the results are always a table and the first row must contain the column names.
- The Python code or R code that was executed is displayed.
- The standard Python output or R output is displayed.
- The node properties are displayed.
- When in *Supervised Learning* mode, the list of Predicted Variables expected by the project is displayed. In addition, if available, the top 10 observations from the input data and the scored data are displayed.
- The model assessment reports, such as Lift, ROC, and Fit Statistics, are displayed.

Using the Open Source Code Node

When running the *Open Source Code* node, additional Python or R code is generated and added before and after your code. The precursor code creates target, input, and partition variables; input handles; and other necessary data items. The posterior code, when applicable, converts the scored data frame to a CSV file.
You can use the Open Source Code node to produce a custom model that is incorporated into your pipeline. To indicate that your Code node will produce a model that should be assessed and processed by the Model Comparison node, complete the following steps:

1. Right-click the node, and select Move.
2. Select Supervised Learning.
3. The Open Source Code node will now be connected to a Model Comparison node.

In order for the model produced by the Open Source Code node to be assessed, it must either produce a scored data set with the appropriate prediction variables names. Otherwise, no assessment reports are generated. To ensure that reports are generated, ensure that the following criteria are met:

- If the Generate data frame property is selected, the scored data set must be saved to dm_scoreddf. Otherwise, it must be saved to node_scored.csv.
- The prediction variables in the scored data set must be named according to the following convention:
  - Interval targets must be named P_<targetVariableName>. For example, if MSRP is the target variable then the prediction variable must be P_MSBP.
  - Categorical targets must be named P_<targetVariableName><targetLevel>. All target level properties must be computed. For example, if BAD is the target variable with levels 0 and 1 then the posterior probabilities should be P_BAD0 and P_BAD1.
- The scored data set must contain the same number of observations as the input data set.

---

**Installation of Python or R Software**

The Python or R software must be installed on the same machine as the compute server microservice. The executable Python or Rscript file must be available in the system path. On Windows, the environment variables PYTHONHOME or RHOME specify the home directories of Python or R, where the executable file is expected to be located in %PYTHONHOME%/python or %RHOME%/bin/Rscript. On Linux, the path for the compute server microservice can be modified by editing the sasenv_local file under the /opt/sas/spre/home/SASFoundation/bin directory. The following line can be added to the sasenv_local file:

```bash
export PATH=path_to_your_python_or_R_bin_directory:$PATH
```

Consult your SAS Administrator to ensure that the above configurations are satisfied. You also need to install any necessary packages with sudo privileges so that they are accessible to all users. The Open Source Code node is agnostic to the version of Python or R that is installed. It acts to transfer the code to software package that is installed.

To run the Open Source Code node with the Language set to Python and Generate data frame selected, you must have pandas installed. You do not need pandas installed if you deselect Generate data frame.
Common Questions

What happens if you run the Open Source Code node when Python or R is not installed on the compute server?

The Open Source Code node fails if Python or R is not installed or if it is not configured correctly. The executable python or Rscript file must be available in the system path on Linux or their install directories can be specified with PYTHONHOME or RHOME on Windows.

When should you deselect Include SAS formats property?

The Include SAS formats property controls whether the downloaded data sent to the Python or R software should keep SAS formats. It is strongly recommended that you keep SAS formats and this should work in most cases. However, some numeric formats like DOLLARw.d add a dollar sign and change the data type of the variable when exporting to CSV. In such cases, these formats must be removed.

Note that this property will either keep or remove SAS formats for all variables in the data set. If input or target variables have SAS or user-defined formats that significantly modify the data, it is not recommended to deselect this option as the model built might not be comparable to other models.

How are missing values in input data handled in open-source software?

Both Python and R packages sometimes do not support missing values in data. It is your responsibility to prepare the data as necessary for these packages. It is highly recommended that you add an Imputation node before the Open Source Code node to handle missing values. If the training data does not contain missing values but if either the validation or test data does contain missing values, you should consider enabling the Impute non-missing variables property in the Imputation node.

Why do I get dm_traindf not found error when using the training data frame in code?

The data frame variable dm_traindf is created only when input data is partitioned. Ensure that there is a training partition in the project.

Can SWAT packages be installed and CAS actions invoked from Python or R installations on the compute server?

This scenario has not been tested and is not supported. Instead, if CAS actions need to be invoked, it is recommended that you use the SAS Code node and PROC CAS. There is no benefit in making a call to CAS via the Python or R SWAT packages when you can directly do that with PROC CAS in the SAS Code node.

Why does the Open Source Code node not have Assessment results even though it was successfully executed under the Supervised Learning group?

Assessment results might not be generated for one or more of the following reasons:

- When a scored data set (dm_scoreddf or nodescored.csv) is not generated. In this case, the node fails with an error in the Python or R code.
- The scored data set does not contain the expected predicted variable names. The node does not fail but does not produce assessment plots or statistics in the Results tab.
The scored data set does not contain the same number of observations as the input data set. The node does not fail but does not produce assessment plots or statistics in the **Results** tab. The following warning message is generated in the log: Data merge was skipped as observations in input data (<number of observations>) and scored data (<number of observations>) did not match. Hence, model assessment will not be performed.

If any of the above conditions are met, the **Open Source Code** node is not included in the **Model Comparison** node that follows.

**Do variable names have any length requirements?**
When using the R language, variable names are limited to 31 characters. When using Python, variable names are limited to 32 characters. Variable names longer than this are truncated, which causes the node to fail.

**Why does the **Open Source Code** node require pandas package when executing Python code?**
The node requires pandas package because the **Generate data frame** property creates a pandas data frame for input data. There is no pandas requirement when the **Generate data frame** property is deselected.

**Can the **Open Source Code** node be executed when there is no code entered in the editor?**
Yes, it is good practice to execute the node in an empty state to validate whether Python or R is correctly installed and configured. In addition, you can view the precursor code that is added as part of the executed code. The code that is added depends on combination of properties selected. The precursor code is part of the node results.

**Any other debugging tips?**
If the **Open Source Code** node fails with errors related to working with data frames, consider deselecting the **Generate data frame** property and working with CSV files.
Overview of Quantile Regression

The Quantile Regression node is a Supervised Learning node. A quantile regression attempts to fit the quantiles of an interval target as a linear function of one or more features. The Quantile Regression node requires an interval target variable and one or more effect variables. Effect variables can be interval variables, classification variables, interaction effects, or splines.

Quantile Regression Properties

General Options

Quantile — specifies the quantile level for the regression. You can specify any value between 0 and 1, exclusive. For example, specify 0.5 to perform a median regression.
Effects Options

- **Two-factor interactions** — specifies whether to include all two-factor interactions of nominal variables in the model. By default, this option is deselected.
- **Factor split** — specifies whether levels of a nominal variable can enter or leave a model independently. This is done automatically for LASSO selection. For other selection methods, this option is deselected.
- **Spline** — specifies whether to expand interval inputs into cubic B-spline bases with three equally spaced knots. (Expansion yields seven design matrix columns for each of the variables.) Spline and polynomial effects cannot be specified together. By default, this option is deselected.
- **Spline split** — specifies whether each individual column in the design matrix that corresponds to the spline effect is treated as a separate effect that can enter or leave the model. By default, this option is deselected. This option is available only if Spline is selected.
- **Polynomial** — specifies whether to use polynomial effects up to the specified **Polynomial degree** for all interval variables. Spline and polynomial effects cannot be specified together. By default, this option is deselected.
- **Polynomial degree** — specifies the polynomial degree when polynomial terms are included in the model. This option is available only if Polynomial has been selected. Possible values are 2 and 3. The default value is 2.
- **Suppress intercept** — specifies whether to suppress the intercept. By default, this is deselected.
- **Use missing as level** — specifies whether missing values should be treated as a separate level for all nominal inputs. By default, this is deselected.
- **Model missing values for inputs** — specifies whether to include observations with missing values. For interval variables, an indicator variable is created to indicate whether an observation contains a missing value is created and the original value is imputed with the mean. For categorical variables, missing values are treated as their own measurement level.

Selection Method Property

- **Selection method** — specifies a model selection method. Here are the possible values:
  - **Backward** — specifies that training is done by least squares regression that starts with all candidate features in the model. Features are removed one at a time until the stop criterion is met.
  - **Forward** — specifies that training is done by least squares regression that starts with no candidate features in the model. Features are added one at a time until the stop criterion is met.
  - **None** — specifies not to perform variable selection.
Stepwise — specifies that training is done by least squares regression that starts as in the forward model but might remove features already in the model.

The default value is Stepwise.

## Selection Options

- **Effect-selection criterion** — specifies the criterion used to determine how effects are added to the model. Here are the possible values:
  - **Adjusted R1 statistics** — specifies the Adjusted R1 statistic. The Adjusted R1 statistic is an adjustment of the tau statistic that adjusts for ties and can vary between -1 and 1, inclusive. Values closer to 1 or -1 indicate a stronger relationship.
  - **AIC** — specifies Akaike’s Information Criterion. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model. Smaller values indicate better models, and AIC values can become negative.
  - **AICC** — specifies Corrected Akaike’s Information Criterion. AICC adjusts the AIC value by accounting for sample size. Extra features penalize AICC more than AIC. As the sample size increases, AICC and AIC converge.
  - **Average check loss validation** — specifies average check loss for the validation data as the selection method.
  - **R1 statistic** — specifies the R1 statistic. The R1 statistic is also known as the tau statistic and can vary between -1 and 1, inclusive.
  - **SBC (BIC)** — specifies Schwarz’s Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). Unexplained variations in the response variable and the number of features increase the value of the SBC. Smaller values indicate better models. SBC penalizes total number of model parameters more strongly than AIC, and hence tends to choose more sparse models.
  - **Significance level** — specifies the standard statistical significance level criterion.

If the **Selection method** above is LASSO, this option is unavailable. The default value is **SBC (BIC)**.

- **Selection-process stopping criterion** — specifies the criterion to stop the selection process. Here are the possible values:
  - **Adjusted R1 statistics** — specifies the Adjusted R1 statistic. The Adjusted R1 statistic is an adjustment of the tau statistic that adjusts for ties and can vary between -1 and 1, inclusive. Values closer to 1 or -1 indicate a stronger relationship.
  - **AIC** — specifies Akaike’s Information Criterion. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model. Smaller values indicate better models, and AIC values can become negative.
  - **AICC** — specifies Corrected Akaike’s Information Criterion. AICC adjusts the AIC value by accounting for sample size. Extra features penalize AICC more than AIC. As the sample size increases, AICC and AIC converge.
- **Average check loss validation** — specifies average check loss for the validation data as the selection method.
- **None** — specifies no selection—process stopping criterion.
- **SBC (BIC)** — specifies Schwarz’s Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). Unexplained variations in the response variable and the number of features increase the value of the SBC. Smaller values indicate better models. SBC penalizes total number of model parameters more strongly than AIC, and hence tends to choose more sparse models.

- **Significance level** — specifies the standard statistical significance level criterion.

The default value is **SBC (BIC)**.

## Model–selection criterion
- Specifies the criterion to choose the model (from the list of models at each step of the selection process) that yields the best value of the specified criterion. If the optimal value of the specified criterion occurs for models at more than one step, then the model that has the smallest number of parameters is chosen. Here are the possible values:

- **Adjusted R² statistics** — specifies the Adjusted R² statistic. The Adjusted R² statistic is an adjustment of the tau statistic that adjusts for ties and can vary between -1 and 1, inclusive. Values closer to 1 or -1 indicate a stronger relationship.

- **AIC** — specifies Akaike’s Information Criterion. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model. Smaller values indicate better models, and AIC values can become negative.

- **AICC** — specifies Corrected Akaike’s Information Criterion. AICC adjusts the AIC value by accounting for sample size. Extra features penalize AICC more than AIC. As the sample size increases, AICC and AIC converge.

- **Average check loss validation** — specifies average check loss for the validation data as the selection method.

- **SBC (BIC)** — specifies Schwarz’s Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). Unexplained variations in the response variable and the number of features increase the value of the SBC. Smaller values indicate better models. SBC penalizes total number of model parameters more strongly than AIC, and hence tends to choose more sparse models.

The default value is **SBC (BIC)**.

- **Entry significance level** — specifies the significance level for adding variables in the forward and stepwise directions. The default value is 0.05.

- **Stay significance level** — specifies the significance level for removing variables in backward and stepwise directions. The default value is 0.05.

- **Maximum number of effects** — specifies the maximum number of effects in any model that is considered during the selection process. If a model at some step of the selection process contains the specified maximum number of effects, then no additional effects are considered. If **Maximum number of effects** is set to 0 (the default value), this option is ignored.

- **Minimum number of effects** — specifies the minimum number of effects in any model that is considered during the backward selection process. If **Minimum number of effects** is set to 0 (the default value), this option is ignored.
- **Maximum number of steps** — specifies the maximum number of selection steps that are performed. If **Maximum number of steps** is set to 0 (the default value), this option is ignored.

- **Hierarchy** — specifies whether and how the model hierarchy requirement is applied. Model hierarchy refers to the requirement that, for any term to be in the model, all model effects that are contained in the term must be present in the model. For example, in order for the interaction A*B to enter the model, the main effects A and B must also be in the model. Likewise, neither effect A nor effect B can leave the model while the interaction A*B is in the model. Here are the possible values:
  - **All variables** — specifies that both interval and nominal variables are subject to the model hierarchy requirement.
  - **Class variables** — specifies that only nominal variables are subject to the model hierarchy requirement.
  - **None** — specifies that the hierarchy requirement is never applied.

  The default value is **None**.

**Model Interpretability**

Specifies the method of model interpretation in order to show how the model responds to changing inputs.

- **Explain Model Inputs** — specifies the following parameters for the LIME algorithm:
  - **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.
  - **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.
  - **Cluster-based ICE plots** — the maximum number of variables (1-10).

- **Explain Individual Predictions** — specifies whether to include **Number of clusters**, **Number of perturbed instances**, and **Number of iterations**. You can specify the **Distance metric** as either **Cosine** or **K prototype**. You can specify the **Kernel width** and the **Number of inputs to use in the explanation**. You can specify whether to **Create a score code for assigning clusters**.

**Quantile Regression Results**

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Node**
- **t Values by Parameter** — displays bar charts for the t values in the final model. The bars are color-coded to indicate the algebraic signs of the coefficients.

- **Parameter Estimates** — displays a table of the various statistics related to estimates for the parameters. These statistics include the t value, sign, estimate, absolute estimate, the p-value, chi-square value, and standard error.

- **Selection Summary** — displays a table of the iterations, adding in and removing effects. This table also includes the SBC and optimal SBC values.

- **Regression Fit Statistics** — displays a table of data on the quantile regression fit statistics. The table includes the quantile level, statistics, a corresponding qualitative description, and the values.

- **Node Score Code** — displays the SAS score code that was created by the node. The SAS score code can be used outside of the Model Studio environment to score new data.

- **Score Inputs** — displays a table of data on the input variables for scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.

- **Score Outputs** — displays a table of data on the predicted response variable for scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.

- **Path Score Code** — displays the SAS score code that was created by the node, as well as all preceding nodes. The SAS model score code can be used outside of the Model Studio environment to score new data.

- **Train Code** — displays the SAS code that Model Studio used to train the node.

- **Properties** — specifies the various properties selected before running the node.

- **Output** — displays the SAS output of the quantile regression run.

**Assessment**

- **Predicted Reports** — displays the predicted and target mean as a function of the depth for the model. The predicted and target mean are given for each of the data roles. To examine the predicted mean as a function of the target mean, use the drop-down menu in the upper right corner. This result is displayed only if the target is an interval variable.

- **Fit Statistics** — displays a table of the fit statistics for the model, broken down by data role.

**Model Interpretability**

- **Input Relative Importance** — displays a table of the relative importance of the input variables, including variable level and label.

- **Partial Dependence** — displays the functional relationship between the input variable and the model prediction.

- **LIME Plots** — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed around a particular observation to create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when **Explain Individual Predictions** is selected.

- **Local Instances** — displays the values for each input variable of the cluster center.
- **Local Explanation** — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.

- **Local Model** — displays the parameter estimates of the local linear model that is used to predict the cluster center.

- **Prediction** — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

- **LIME Score Code** — displays the SAS code that Model Studio used to assign observations to clusters. This is displayed only when **Create score code for assigning clusters** is selected.

- **Individual Conditional Expectation** — displays the functional relationship between the input variable and the model prediction. The relationships for all cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.
Replacement

Overview of Replacement

The Replacement node is a Data Mining Preprocessing node. It is used to generate score code to replace outliers and unknown class levels with specified values. In some cases, you might want to reassign specified nonmissing values (trim your variable’s distribution) before performing imputation calculations for the missing values. This would be a typical task for the Replacement node.

Replacement Properties

Replacement Value Property

- Replacement value for unknown class levels — specifies the replacement method for unknown class levels encountered in the scoring process. Here are the possible values:
  - Ignore — specifies that variables that contain unknown levels are left unmodified.
  - Missing value — specifies that variables that contain unknown levels are replaced with SAS missing value notations.
  - Mode (most frequent level) — specifies that variables that contain unknown levels are replaced with the mode.

The default value is Ignore.
Interval Inputs

- **Default limits method** — specifies the default method by which the lower and upper limits are derived for interval variables. Here are the possible values:
  - Extreme percentiles
  - Median absolute deviation (MAD)
  - Metadata limits — specifies that any limits stored in the metadata are used for replacement.
  - None
  - Standard deviation from the mean

  The default value is **Standard deviation from the mean**.

- **Extreme percentile** — specifies the maximum and minimum upper and lower percentile to be used in deriving the upper and lower limits. The default value is 0.5.

- **MAD deviations** — specifies the number of MAD deviations from the median to be used in deriving the lower and upper limits. The default value is 9.

- **Alternate limits method** — specifies the method for calculating interval variable limits when metadata lacks stored limits. Here are the possible values:
  - Standard deviation from the mean
  - Median absolute deviation (MAD)
  - Extreme percentiles
  - None

- **Standard deviations** — specifies the number of Standard deviations from the mean to be used in deriving the lower and upper limits. The default value is 3.

- **Replacement value** — specifies the replacement value for lower and upper outlier values. Possible values are **Computed limits** and **Missing value**. The default setting is **Computed limits**.

Replacement Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Results**

- **Class Variables** — displays a list table of the replacement class variables and their replacement values for unknown levels.

- **Interval Variables** — displays a list table that summarizes the replaced variables, the limits method, the lower and upper limits, the replacement method, and the lower and upper replacement values.
- **Replacement Counts** — displays a list table that summarizes the variable replacement counts, their role, and level.

- **Node Score Code** — displays SAS score code that was created by the node. The SAS score code can be used outside of the Model Studio environment in custom user applications.

- **Properties** — specifies the various properties that you can select before running the node.

- **Output** — displays the SAS output of the replacement run.
Overview of SAS Code

The **SAS Code** node is a Miscellaneous node that enables you to incorporate new or existing SAS code into Model Studio pipelines. The **SAS Code** node extends the functionality of Model Studio by making other SAS System procedures available for use in your data mining analysis. You can also write SAS DATA steps to create customized scoring code, conditionally process data, or manipulate existing data sets. The **SAS Code** node is also useful for building predictive models, formatting SAS output, defining table and plot views in the user interface, and for modifying variables metadata. The **SAS Code** node can be placed at any location within a pipeline. By default, the **SAS Code** node does not require data. The exported data that is produced by a successful **SAS Code** node run can be used by subsequent nodes in a pipeline.

For Model Studio examples using the **SAS Code** node, see [SAS Code Node Examples](#) in the Model Studio user guide documentation.

SAS Code Properties

- **Code editor** — invokes the SAS Code Editor. The next section will provide documentation on the SAS Code Editor.
- **Train only data** — specifies whether the node should receive the training observations only if the data is partitioned. By default, this option is deselected.
Currently, this property is unavailable for this node. To specify that the node receive only training data, add the following where Clause to your code:

```
Where &dm_partitionvar.=1;
```

- **Model Interpretability** — specifies the method of model interpretation in order to show how the model responds to changing inputs. This property becomes available when the SAS code is used in a supervised learning pipeline.

- **Explain Model Inputs** — specifies the following parameters for the LIME algorithm:
  - **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.
  - **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.
  - **Cluster-based ICE plots** — the maximum number of variables (1-10).

- **Explain Individual Predictions** — specifies whether to include Number of clusters, Number of perturbed instances, and Number of iterations. You can specify the Distance metric as either Cosine or K prototype. You can specify the Kernel width and the Number of inputs to use in the explanation. You can specify whether to Create a score code for assigning clusters.

---

**Code Editor User Interface**

**Macros Table**

The following macros are defined and available for use in the SAS Code Editor.

<table>
<thead>
<tr>
<th>Macro Name</th>
<th>Macro Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dm_dec_target</td>
<td>A macro that identifies the name of the target variable.</td>
</tr>
<tr>
<td>dmcas_metachange</td>
<td>A macro that specifies changes to the metadata.</td>
</tr>
<tr>
<td>dm_unary_input</td>
<td>A macro that identifies unary input variables with missing values.</td>
</tr>
<tr>
<td>dm_interval_input</td>
<td>A macro that identifies interval input variables.</td>
</tr>
<tr>
<td>dm_text</td>
<td>A macro that identifies text variables.</td>
</tr>
<tr>
<td>Macro Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dmcas_copybinaryfile</td>
<td>A macro that copies a binary file.</td>
</tr>
<tr>
<td>dmcas_copyfile</td>
<td>A macro that copies a text file.</td>
</tr>
<tr>
<td>dm_nominal_input</td>
<td>A macro that identifies nominal input variables.</td>
</tr>
<tr>
<td>dmcas_checkMacroVar</td>
<td>A macro that checks the existence of a macro variable. A default value is assigned if the macro variable does not exist.</td>
</tr>
<tr>
<td>dm_offset</td>
<td>A macro that identifies the name of the offset variable.</td>
</tr>
<tr>
<td>dm_binary_input</td>
<td>A macro that identifies the binary input variables.</td>
</tr>
<tr>
<td>dm_ordinal_input</td>
<td>A macro that identifies the ordinal input variables.</td>
</tr>
<tr>
<td>dm_segment</td>
<td>A macro that identifies the segment variables.</td>
</tr>
<tr>
<td>dm_id</td>
<td>A macro that identifies the ID variables.</td>
</tr>
<tr>
<td>dmcas_varmacro</td>
<td>A macro used to generate a macro that contains a list of variables that satisfy a specified where clause. The macro also generates a macro variable that identifies the number of variables in the list. The macro arguments are:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
dmcas_register

A macro that registers files and data sets. The macro arguments are:
- DATASET — the data set name.
- FILE — the file name.
- PROPERTY — specifies (Y or N) property file. The default value is N.
- MODELMANAGER — specifies (Y or N) whether to export the file to model manager. The default value is N.

dmcas_report

A macro used to register and define reports. The macro arguments are:
- DATASET — the data set name.
- FILE — the file name.
- REPORTTYPE — the report type. Possible values are BandPlot, BarChart, CodeEditor, DecisionTree, Html, IciclePLot, NetworkDiagram, PieChart, ScatterPlot, SeriesPlot, and Table.
- DESCRIPTION — the report description.
- X — the category variable.
- Y — the response variable.
- GROUP — the group variables.
- SUBGROUP — the subgroup variables.
- FREQ — the frequency variable.
- BY — the by or lattice variables.
- WHERE — the where clause.
- AUTODISPLAY — specifies (Y or N) whether to auto display report when the result viewer is open.
- BLOCK — defines, if applicable, the group in which the report belongs to when opening the result viewer.

Macro Variables Table

The following macro variables are defined and available for use in the SAS Code Editor.

<table>
<thead>
<tr>
<th>Macro Variable</th>
<th>Macro Variable Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dm_partitionvar</td>
<td>A macro variable that identifies the name (blank or <em>partind</em>) of the partition variable.</td>
</tr>
<tr>
<td>dm_codebar</td>
<td>A macro variable that identifies a code separator.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dm_dsep</td>
<td>A macro variable that identifies the operating system file delimiter (for example, for Windows).</td>
</tr>
<tr>
<td>dm_projectData</td>
<td>A macro variable that identifies the project CAS source table.</td>
</tr>
<tr>
<td>dm_file astore</td>
<td>A macro variable that identifies the local analytical store.</td>
</tr>
<tr>
<td>dm_dec_target</td>
<td>A macro variable that identifies the name of the target variable.</td>
</tr>
<tr>
<td>dm_dec_level</td>
<td>A macro variable that identifies the measurement level of the target variable. Possible measurement levels are BINARY, INTERVAL, ORDINAL, and NOMINAL.</td>
</tr>
<tr>
<td>dm_key</td>
<td>A macro variable that identifies the name of the key variable.</td>
</tr>
<tr>
<td>dm_num_interval_input</td>
<td>A macro variable that identifies the number of interval input variables.</td>
</tr>
<tr>
<td>dm_data_outfit</td>
<td>A macro variable that identifies the fit statistics data set.</td>
</tr>
<tr>
<td>dm_datalib</td>
<td>A macro variable that identifies the libref associated with the training table.</td>
</tr>
<tr>
<td>dm_nodedir</td>
<td>A macro variable that identifies the folder where local files will be created.</td>
</tr>
<tr>
<td>dm_data</td>
<td>A macro variable that identifies the CAS training table.</td>
</tr>
<tr>
<td>dm_data rstore</td>
<td>A macro variable that identifies the remote analytical store.</td>
</tr>
<tr>
<td>dm_file_traincode</td>
<td>A macro variable that identifies the file that contains the training code.</td>
</tr>
<tr>
<td>dm_num_text</td>
<td>A macro variable that identifies the number of text variables.</td>
</tr>
<tr>
<td>dm_folder_scorevar</td>
<td>A macro variable that identifies the local folder that contains the score code files. The score code files contain individual variable transformations.</td>
</tr>
<tr>
<td>dm_memname</td>
<td>A macro variable that identifies the name of the training table.</td>
</tr>
<tr>
<td>dm_data caslib</td>
<td>A macro variable that identifies the CASLIB associated with the training table.</td>
</tr>
<tr>
<td>dm_num id</td>
<td>A macro variable that identifies the number of ID variables.</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dm_num_nominal_input</td>
<td>A macro variable that identifies the number of nominal input variables.</td>
</tr>
<tr>
<td>dm_dec_format</td>
<td>A macro variable that identifies the format of the target variable.</td>
</tr>
<tr>
<td>dm_dec_order</td>
<td>A macro variable that identifies the order of the target variable. Possible values are ASCFMT, ASC, DESC, and DESFMT. The default value is DESC.</td>
</tr>
<tr>
<td>dm_num_unary_input</td>
<td>A macro variable that identifies the number of unary input variables with missing values. Variables that are constant without missing values are assigned the role rejected.</td>
</tr>
<tr>
<td>dm_partition_statement</td>
<td>A macro variable that identifies the partition statement.</td>
</tr>
<tr>
<td>dm_nodeid</td>
<td>A macro variable that identifies the nodeID.</td>
</tr>
<tr>
<td>dm_num_binary_input</td>
<td>A macro variable that identifies the number of binary input variables.</td>
</tr>
<tr>
<td>dm_dec_label</td>
<td>A macro variable that identifies the label of the target variable.</td>
</tr>
<tr>
<td>dm_metadata</td>
<td>A macro variable that identifies the local data set that contains the node imparted metadata.</td>
</tr>
<tr>
<td>dm_file_deltacode</td>
<td>A macro variable that identifies the file that contains data step code that modifies the columnsmeta meta exported by the node.</td>
</tr>
<tr>
<td>dm_num_segment</td>
<td>A macro variable that identifies the number of segment variables.</td>
</tr>
<tr>
<td>dm_dec_type</td>
<td>A macro variable that identifies the type (C or N) of the target variable.</td>
</tr>
<tr>
<td>dm_data_scorevar</td>
<td>A macro variable that identifies the data set used to register individual variable transformations.</td>
</tr>
<tr>
<td>dm_lib</td>
<td>A macro variable that identifies the SAS library where components should save and create local tables.</td>
</tr>
<tr>
<td>dm_dec_event</td>
<td>A macro variable that identifies the event level of a class target variable. The event is the first level as defined by the order of the target variable.</td>
</tr>
<tr>
<td>dm_maxnamelen</td>
<td>A macro variable that identifies the maximum length for the name of a new variable.</td>
</tr>
</tbody>
</table>
### Code Editor Menu

- **Options** — specifies options for code editor settings. The following options are available:
  - **General** — specifies general property configurations:
    - **Enable code folding** — specifies whether to enable wraparound of code text. By default, this option is selected.
    - **Show line numbers** — specifies whether to display line numbers to the left of the code editor pane. By default, this option is selected.
    - **Enable line highlighting** — specifies whether to highlight the line the cursor is currently on. By default, this option is selected.
    - **Font size** — specifies the font size of the text inside the editor pane. Possible font sizes range from 8 to 72, 16 being the default value.

    To set all values back to default, click **Reset**.
  - **Editing** — specifies properties related to the editor. The following properties are available:
    - **Enable autocomplete** — specifies whether to enable autocomplete for SAS keywords, properties, and functions. By default, this option is selected.
    - **Enable color coding** — specifies whether to enable the color coding system that uses different colors to differentiate between different types of SAS keywords. By default, this option is selected.
    - **Tab width** — specifies the width of a Tab keystroke in spaces. The default value is 4.
    - **Substitute spaces for tabs** — specifies whether to substitute a Tab keystroke with the equivalent number of spaces. By default, this option is deselected.
    - **Enable auto indentation** — specifies whether to automatically indent at the beginning of new code blocks. By default, this option is selected.

    To set all values back to default, click **Reset**.
  - **Others** — specifies other properties for the code editor. **Print Line Numbers** displays line numbers when the code is printed. By default, this option is selected. To set this value back to default, click **Reset**.

    - **Undo** — reverses the most recent change to the code pane.
    - **Redo** — restores the most recent undone change to the code pane.
    - **Cut** — deletes the selected item and copies it to the clipboard.
    - **Copy** — copies the selected item to the clipboard.
SAS Code Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Node**
- **SAS Code** — displays the SAS code from the code editor.
- **Score Inputs** — displays a table of data on the input variables for scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.
- **Score Outputs** — displays a table of data on the predicted response variable for scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.
- **Path EP Score Code** — displays the SAS code that was created by the node. The score code can be used outside of the Model Studio environment to score new data.
- **DS2 Package Code** — displays the SAS DS2 package code that was created by the node. The score code can be used outside of the Model Studio environment to score new data.
- **Properties** — specifies the various properties selected before running the node.
- **Output** — displays the SAS output of the code node run—that is, the SAS output of the code in the Code Editor.

**Assessment**
- **Predicted Reports** — displays the predicted and target mean as a function of the depth for the model. The predicted and target mean are given for each of the data roles. To examine the predicted mean as a function of the target mean, use the drop-down menu in the upper right corner. This result is displayed only if the target is an interval variable.
- **Lift Reports** — displays the cumulative lift as a function of the depth for the model. The cumulative lift is given for each of the data roles. To examine other statistics as a function of depth, use the drop-down menu in the upper right corner. Other statistics include lift, gain, % captured response, cumulative % captured response, % response, and cumulative % response. This result is displayed only if the target is a class variable.
- **ROC Reports** — displays the ROC (receiver operating characteristic) chart for a model, giving the sensitivity as a function of 1-specificity. The sensitivity is given
for each of the data roles. To examine other statistics, use the drop-down menu in the upper right corner. Other statistics include accuracy and F1 score. This result is displayed only if the target is a class variable.

- **Fit Statistics** — displays a table of the fit statistics for the model, broken down by data role.

**Model Interpretability**

- **Input Relative Importance** — displays a table of the relative importance of the input variables, including variable level and label.

- **Partial Dependence** — displays the functional relationship between the input variable and the model prediction.

- **LIME Plots** — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed around a particular observation to create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when **Explain Individual Predictions** is selected.
  
  - **Local Instances** — displays the values for each input variable of the cluster center.
  
  - **Local Explanation** — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.
  
  - **Local Model** — displays the parameter estimates of the local linear model that is used to predict the cluster center.
  
  - **Prediction** — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

- **LIME Score Code** — displays the SAS code that Model Studio used to assign observations to clusters. This is displayed only when **Create score code for assigning clusters** is selected.

- **Individual Conditional Expectation** — displays the functional relationship between the input variable and the model prediction. The relationships for all cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.

---

**Using the SAS Code Node**

You can use your **SAS Code** node to produce a custom model that is incorporated into your pipeline. To indicate that your **SAS Code** node will produce a model that should be assessed and processed by the **Model Comparison** node, complete the following steps:

1. Right click the node, and select **Move**.
2. Select **Supervised Learning**.
3. The **SAS Code** node will now be connected to a **Model Comparison** node.
In order for the model produced by the SAS Code node to be assessed, it must either produce DATA step score code or an analytic store that will produce the expected predictor or posterior variables. The &dm_lib..dmcas_targetLevel data set provides the expected names as a macro variable.

To move your SAS Code node out of the Supervised Learning group, complete the following steps:

1. Right click the SAS Code node, and select Move.
2. Select Preprocessing.
Overview of Save Data

The Save Data node is a Miscellaneous node that enables you to save the training table that is produced by a predecessor node to a CAS library. This table could be partitioned into training, validation, or test sets based on the project settings. In that case, the table contains the _partind_ variable that identifies the partitions.

Save Data Properties

General Properties

- **Output library** — specifies the output CAS library where the table will be saved on disk. Use **Browse** to navigate to the proper library. If the user has specified an **Output Library** under **Project Settings**, then this library will be used by default.

- **Table name** — specifies the name for the CAS table being saved. The default value is **tmpSaveData**.

- **Replace existing table** — specifies whether to override an existing CAS table with the same name when saving. By default, this option is deselected.

- **Promote table** — specifies whether to load the table in memory and promote the table to global space. By default, this option is deselected.
Save Data Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Results**

- **Properties** — specifies the various properties selected prior to running the node. For the **Save Data** node, these include the output library, the table name, whether to replace or promote the table, and the CAS session ID.

- **Output** — displays the SAS output of the save data run.
Score Code Import

Overview of Score Code Import

The Score Code Import node enables you to import external models that are saved as SAS score code.

Using the Score Code Import Node

To use the Score Code Import node, perform the following steps:

1. Build a process flow diagram in SAS Enterprise Miner. Your process flow diagram must end in a Score node.

2. Right-click the Score node in your diagram and select Export Path as SAS Program. Ensure that Run this path is enabled in the Export Path as SAS Program window. Specify a location to save the score code and click OK.

To use analytic store code, save the ecode file and save the score data to a CAS library that the mid-tier can access, such as a public CAS library. The Model Studio mid-tier cannot access an analytic store that is located in a user’s personal CAS library.

3. In a Model Studio pipeline, add a Score Code Import node to your pipeline.

4. In the Score Code Import node properties pane, click Open and navigate to the file that you saved earlier. The score code that you created in SAS Enterprise Miner is opened in the code editor.

5. Click to save this code. Then click Close.

6. Continue building your pipeline. When you run your pipeline, the Score Code Import node runs alongside all other nodes.
Score Code Import Properties

- **Code editor** — invokes the Score Code Editor. For more information about the SAS Code Editor see Code Editor User Interface on page 142.
- **Model Interpretability** — specifies the method of model interpretation.
  - **Explain Model Inputs** — specifies the following parameters for the LIME algorithm:
    - **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.
    - **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.
    - **Cluster-based ICE plots** — the maximum number of variables (1-10).
  - **Explain Individual Predictions** — specifies whether to include **Number of clusters**, **Number of perturbed instances**, and **Number of iterations**. You can specify the **Distance metric** as either **Cosine** or **K prototype**. You can specify the **Kernel width** and the **Number of inputs to use in the explanation**. You can specify whether to **Create a score code for assigning clusters**.

Score Code Import Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Node**

- **Path Score Code** — provides the optimized SAS code from SAS Enterprise Miner.
- **Score Inputs** — displays a table of data on the input variables for scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.
- **Score Outputs** — displays a table of data on the predicted response variable for scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.
- **EM Score Code** — provides the score code that was created by SAS Enterprise Miner.

**Assessment**

- **Predicted Reports** — displays the predicted and target mean as a function of the depth for the model. The predicted and target mean are given for each of the
data roles. To examine the predicted mean as a function of the target mean, use the drop-down menu in the upper right corner. This result is displayed only if the target is an interval variable.

- **Lift Reports** — displays the cumulative lift as a function of the depth for the model. The cumulative lift is given for each of the data roles. To examine other statistics as a function of depth, use the drop-down menu in the upper right corner. Other statistics include lift, gain, % captured response, cumulative % captured response, % response, and cumulative % response. This result is displayed only if the target is a class variable.

- **ROC Reports** — displays the ROC (receiver operating characteristic) chart for a model, giving the sensitivity as a function of 1-specificity. The sensitivity is given for each of the data roles. To examine other statistics, use the drop-down menu in the upper right corner. Other statistics include accuracy and F1 score. This result is displayed only if the target is a class variable.

- **Fit Statistics** — displays a table of the fit statistics for the model, broken down by data role.

**Model Interpretability**

- **Input Relative Importance** — displays a table of the relative importance of the input variables, including variable level and label.

- **Partial Dependence** — displays the functional relationship between the input variable and the model prediction.

- **LIME Plots** — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed around a particular observation to a create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when **Explain Individual Predictions** is selected.
  
  - **Local Instances** — displays the values for each input variable of the cluster center.
  
  - **Local Explanation** — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.
  
  - **Local Model** — displays the parameter estimates of the local linear model that is used to predict the cluster center.
  
  - **Prediction** — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

- **LIME Score Code** — displays the SAS code that Model Studio used to assign observations to clusters. This is displayed only when **Create score code for assigning clusters** is selected.

- **Individual Conditional Expectation** — displays the functional relationship between the input variable and the model prediction. The relationships for all cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.
Overview of SVM

The SVM node is a Supervised Learning node. An SVM, or support vector machine, is a supervised machine-learning method that is used for binary pattern recognition problems.

Most problems in a finite dimensional space are not linearly separable. In this case, the original space needs to be mapped into a much higher dimensional space, which makes the separation easier. The node currently supports linear and low-degree polynomial kernels.

SVM Properties

<table>
<thead>
<tr>
<th>General Properties</th>
<th>157</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Autotuning</td>
<td>158</td>
</tr>
<tr>
<td>Binary Classification Cutoff</td>
<td>160</td>
</tr>
<tr>
<td>Model Interpretability</td>
<td>160</td>
</tr>
</tbody>
</table>

SVM Results

161

Overview of SVM

The SVM node is a Supervised Learning node. An SVM, or support vector machine, is a supervised machine-learning method that is used for binary pattern recognition problems.

Most problems in a finite dimensional space are not linearly separable. In this case, the original space needs to be mapped into a much higher dimensional space, which makes the separation easier. The node currently supports linear and low-degree polynomial kernels.

SVM Properties

<table>
<thead>
<tr>
<th>General Properties</th>
<th>157</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Autotuning</td>
<td>158</td>
</tr>
<tr>
<td>Binary Classification Cutoff</td>
<td>160</td>
</tr>
<tr>
<td>Model Interpretability</td>
<td>160</td>
</tr>
</tbody>
</table>

SVM Results

161

Overview of SVM

The SVM node is a Supervised Learning node. An SVM, or support vector machine, is a supervised machine-learning method that is used for binary pattern recognition problems.

Most problems in a finite dimensional space are not linearly separable. In this case, the original space needs to be mapped into a much higher dimensional space, which makes the separation easier. The node currently supports linear and low-degree polynomial kernels.

SVM Properties

<table>
<thead>
<tr>
<th>General Properties</th>
<th>157</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Autotuning</td>
<td>158</td>
</tr>
<tr>
<td>Binary Classification Cutoff</td>
<td>160</td>
</tr>
<tr>
<td>Model Interpretability</td>
<td>160</td>
</tr>
</tbody>
</table>

SVM Results

161

Overview of SVM

The SVM node is a Supervised Learning node. An SVM, or support vector machine, is a supervised machine-learning method that is used for binary pattern recognition problems.

Most problems in a finite dimensional space are not linearly separable. In this case, the original space needs to be mapped into a much higher dimensional space, which makes the separation easier. The node currently supports linear and low-degree polynomial kernels.

SVM Properties

<table>
<thead>
<tr>
<th>General Properties</th>
<th>157</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Autotuning</td>
<td>158</td>
</tr>
<tr>
<td>Binary Classification Cutoff</td>
<td>160</td>
</tr>
<tr>
<td>Model Interpretability</td>
<td>160</td>
</tr>
</tbody>
</table>

SVM Results

161

Overview of SVM

The SVM node is a Supervised Learning node. An SVM, or support vector machine, is a supervised machine-learning method that is used for binary pattern recognition problems.

Most problems in a finite dimensional space are not linearly separable. In this case, the original space needs to be mapped into a much higher dimensional space, which makes the separation easier. The node currently supports linear and low-degree polynomial kernels.

SVM Properties

<table>
<thead>
<tr>
<th>General Properties</th>
<th>157</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Autotuning</td>
<td>158</td>
</tr>
<tr>
<td>Binary Classification Cutoff</td>
<td>160</td>
</tr>
<tr>
<td>Model Interpretability</td>
<td>160</td>
</tr>
</tbody>
</table>

SVM Results

161

Overview of SVM

The SVM node is a Supervised Learning node. An SVM, or support vector machine, is a supervised machine-learning method that is used for binary pattern recognition problems.

Most problems in a finite dimensional space are not linearly separable. In this case, the original space needs to be mapped into a much higher dimensional space, which makes the separation easier. The node currently supports linear and low-degree polynomial kernels.

SVM Properties

<table>
<thead>
<tr>
<th>General Properties</th>
<th>157</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Autotuning</td>
<td>158</td>
</tr>
<tr>
<td>Binary Classification Cutoff</td>
<td>160</td>
</tr>
<tr>
<td>Model Interpretability</td>
<td>160</td>
</tr>
</tbody>
</table>

SVM Results

161

Overview of SVM

The SVM node is a Supervised Learning node. An SVM, or support vector machine, is a supervised machine-learning method that is used for binary pattern recognition problems.

Most problems in a finite dimensional space are not linearly separable. In this case, the original space needs to be mapped into a much higher dimensional space, which makes the separation easier. The node currently supports linear and low-degree polynomial kernels.

SVM Properties

<table>
<thead>
<tr>
<th>General Properties</th>
<th>157</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Autotuning</td>
<td>158</td>
</tr>
<tr>
<td>Binary Classification Cutoff</td>
<td>160</td>
</tr>
<tr>
<td>Model Interpretability</td>
<td>160</td>
</tr>
</tbody>
</table>

SVM Results

161

Overview of SVM

The SVM node is a Supervised Learning node. An SVM, or support vector machine, is a supervised machine-learning method that is used for binary pattern recognition problems.

Most problems in a finite dimensional space are not linearly separable. In this case, the original space needs to be mapped into a much higher dimensional space, which makes the separation easier. The node currently supports linear and low-degree polynomial kernels.

SVM Properties

<table>
<thead>
<tr>
<th>General Properties</th>
<th>157</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Autotuning</td>
<td>158</td>
</tr>
<tr>
<td>Binary Classification Cutoff</td>
<td>160</td>
</tr>
<tr>
<td>Model Interpretability</td>
<td>160</td>
</tr>
</tbody>
</table>

SVM Results

161

Overview of SVM

The SVM node is a Supervised Learning node. An SVM, or support vector machine, is a supervised machine-learning method that is used for binary pattern recognition problems.

Most problems in a finite dimensional space are not linearly separable. In this case, the original space needs to be mapped into a much higher dimensional space, which makes the separation easier. The node currently supports linear and low-degree polynomial kernels.

SVM Properties

<table>
<thead>
<tr>
<th>General Properties</th>
<th>157</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Autotuning</td>
<td>158</td>
</tr>
<tr>
<td>Binary Classification Cutoff</td>
<td>160</td>
</tr>
<tr>
<td>Model Interpretability</td>
<td>160</td>
</tr>
</tbody>
</table>

SVM Results

161

Overview of SVM

The SVM node is a Supervised Learning node. An SVM, or support vector machine, is a supervised machine-learning method that is used for binary pattern recognition problems.

Most problems in a finite dimensional space are not linearly separable. In this case, the original space needs to be mapped into a much higher dimensional space, which makes the separation easier. The node currently supports linear and low-degree polynomial kernels.

SVM Properties

<table>
<thead>
<tr>
<th>General Properties</th>
<th>157</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Autotuning</td>
<td>158</td>
</tr>
<tr>
<td>Binary Classification Cutoff</td>
<td>160</td>
</tr>
<tr>
<td>Model Interpretability</td>
<td>160</td>
</tr>
</tbody>
</table>

SVM Results

161

Overview of SVM

The SVM node is a Supervised Learning node. An SVM, or support vector machine, is a supervised machine-learning method that is used for binary pattern recognition problems.

Most problems in a finite dimensional space are not linearly separable. In this case, the original space needs to be mapped into a much higher dimensional space, which makes the separation easier. The node currently supports linear and low-degree polynomial kernels.

SVM Properties

<table>
<thead>
<tr>
<th>General Properties</th>
<th>157</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Autotuning</td>
<td>158</td>
</tr>
<tr>
<td>Binary Classification Cutoff</td>
<td>160</td>
</tr>
<tr>
<td>Model Interpretability</td>
<td>160</td>
</tr>
</tbody>
</table>

SVM Results

161
- **Penalty** — specifies the penalty value. The default value is 1.
- **Kernel** — specifies the kernel type that the support vector machine uses. Possible values are **Linear** and **Polynomial**. The default value is **Linear**.
- **Polynomial Degree** — specifies the degree of the polynomial that is used. This option is available only if the **Kernel** type is **Polynomial**. Possible values are 2 and 3. The default value is 2.
- **Tolerance** — specifies the minimum number at which the iteration stops. The default value is 0.000001.
- **Maximum iterations** — specifies the maximum number of iterations allowed with each try. The default value is 25.

### Perform Autotuning

Specifies whether to perform autotuning of any SVM parameters. Warning: Performing autotuning can substantially increase run time. If **Perform Autotuning** is selected, the following options are available:

- **Penalty** — specifies whether to autotune the penalty value. If selected, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the penalty parameter. The default value is 1. Use the **From** and **To** options to specify the range. The default **From** value is 0.000001, and the default **To** value is 100.

- **Polynomial degree** — specifies whether to autotune the polynomial degree for the SVM model. If selected, the following option is available:
  - **Initial value** — specifies the initial value for autotuning the polynomial degree. The default value is 1. Use the **From** and **To** options to specify the range. The default **From** value is 1, and the default **To** value is 3.

- **Search Options** — specifies the options for autotuning searching. The following options are available:
  - **Search method** — specifies the tuning search method. Here are the possible values:
    - **Bayesian** — specifies the Bayesian method. This method uses priors to seed the iterative optimization.
    - **Genetic algorithm** — specifies the genetic algorithm method. This method uses an initial Latin Hypercube sample that seeds a genetic algorithm to generate a new population of alternative configurations at each iteration.
    - **Grid** — specifies the grid search method. This method performs an exhaustive search of all configurations. Each hyperparameter of interest is discretized into a desired set of values to be studied. Models are trained and assessed for all combinations of values across all the hyperparameters (thus forming a multi-dimensional “grid”).
    - **Latin hypercube sample** — specifies the Latin Hypercube method. This method performs an optimized grid search that is uniform in each tuning parameter, but random in combinations.
- **Random** — specifies the Random method. This method generates a single sample of purely random configurations.

The default value is **Genetic algorithm**.

- **Number of evaluations per iteration** — specifies the number of tuning evaluations in one iteration. This option is available only if the **Search method** is **Genetic algorithm** or **Bayesian**. The default value is 10.

- **Maximum number of evaluations** — specifies the maximum number of tuning evaluations. This option is available only if the **Search method** is **Genetic algorithm** or **Bayesian**. The default value is 50.

- **Maximum number of iterations** — specifies the maximum number of tuning iterations. This option is available only if the **Search method** is **Genetic algorithm** or **Bayesian**. The default value is 5.

- **Maximum number of points in model** — specifies the maximum number of points in the model. This option is available only if the **Search method** is **Bayesian**. The default value is 100.

- **Sample size** — specifies the sample size. This option is available only if the **Search method** is **Random** or **Latin hypercube sample**. The default value is 50.

- **General Options** — specifies the general properties for autotuning. The following options are available:

  - **Validation method** — specifies the validation method for finding the objective value. Note that if your data is partitioned, then that partition is used and **Validation method**, **Validation data proportion**, and **Cross validation number of folds** are all ignored. Here are the possible values:
    - **Partition** — specifies using the partition validation method. With partition, you will specify proportions to use for randomly assigning observations to each role.
    - **K-fold cross validation** — specifies using the cross validation method. In cross validation, each model evaluation requires \( k \) training executions (on \( k-1 \) data folds) and \( k \) scoring executions (on one holdout fold). This increases the evaluation time by approximately a factor of \( k \).

  For small to medium data tables, cross validation provides, on average, a better representation of error across the whole data table. **Partition** is the default value.

  - **Training data proportion** — specifies the proportion of data to be used for training in the **Partition** validation method. The default value is 0.7.

  - **Validation data proportion** — specifies the proportion of data to be used for the **Partition** validation method. The default value is 0.3.

  - **Cross validation number of folds** — specifies the number of partition folds in the cross validation process (the \( k \) defined above). Possible values range from 2 to 20. The default value is 5.

  - **Class target objective function** — specifies the objective function to optimize for tuning parameters for a class target. Here are the possible values:
    - **Area under the curve**
    - **Average squared error**
    - **F0.5 score**
- F1 score
- Gamma
- Gini coefficient
- Kolmogorov-Smirnov statistic
- Misclassification rate
- Multi-class log loss
- Root average squared error
- Tau

The default value is **Kolmogorov-Smirnov statistic**.

- **Maximum time (minutes)** — specifies the maximum time (in minutes) for the optimization tuner. The default value is 60.

- **Maximum training time for single model (minutes)** — specifies the maximum time (in minutes) for a single model train. To specify no maximum, do not enter a value for **Maximum training time for single model (minutes)**.

### Binary Classification Cutoff

- **Specify node binary classification cutoff** — specifies whether to use the binary classification cutoff specified below for the node. If deselected, the project binary classification cutoff is used for determining the predicted value for a binary target based on the posterior probabilities. By default, this value is deselected.

- **Node binary classification cutoff** — specifies the cutoff to use in the node for determining the predicted value for a binary target based on the posterior probabilities. This option is available only if **Specify node binary classification cutoff** is selected. The default value is 0.5.

### Model Interpretability

Specifies the method of model interpretation in order to show how the model responds to changing inputs.

- **Explain Model Inputs** — specifies the following parameters for the LIME algorithm:
  - **Input relative importance table** — a ranked list of the most important model inputs. The most important input is assigned a relative importance of 1, and unimportant inputs have relative importance values near zero.
  - **Partial dependence plots** — the maximum number of variables for the partial dependence plot (1-10), and the selection of observations as either a sample or a cluster and their number. The default number of sample observations is 500. The default number of clusters is 100.
  - **Cluster-based ICE plots** — the maximum number of variables (1-10).
Explain Individual Predictions — specifies whether to include Number of clusters, Number of perturbed instances, and Number of iterations. You can specify the Distance metric as either Cosine or K prototype. You can specify the Kernel width and the Number of inputs to use in the explanation. You can specify whether to Create a score code for assigning clusters.

SVM Results

After running the node, you can open the Results window by right-clicking the node and selecting Results from the pop-up menu.

Node

- **Fit Statistics** — displays the SVM fit statistics for the model. These include accuracy, error, sensitivity, and specificity, and their value for each data role (Train, Validate, or Test).
- **Training Results** — displays a table of data on the SVM training statistics. The table includes the statistic, a corresponding qualitative description, and the value for the statistic.
- **Iteration Report** — displays a table of data on model accuracy after every iteration of the training process for each data role. This chart will not be displayed if Include iterations report is not selected.
- **Autotune Best Configuration** — displays a table of the resulting values of parameters that were autotuned. This table will not be displayed if autotuning is not selected.
- **Autotune Results** — displays the iterative results of the autotuning process, showing how each of the parameter values evolve as the model iterates. This table will not be displayed if autotuning is not selected.
- **Score Inputs** — displays a table of input variables used in scoring calculations. The table includes the name, role, and the variable level, type, label, format, and length.
- **Score Outputs** — displays a table of predicted response variables generated during scoring calculations. The table includes the name, role, type, the variable format and length, creator, creator GUID, and function.
- **Path EP Score Code** — displays the SAS score code that was created by the node. The score code can be used outside of the Model Studio environment to score new data.
- **DS2 Package Code** — displays the SAS DS2 package code that was created by the node. The score code can be used outside of the Model Studio environment to score new data.
- **Train Code** — displays the SAS code that Model Studio used to train the node.
- **Properties** — specifies the various properties selected before running the node.
- **Output** — displays the SAS output of the SVM run.

Assessment

- **Lift Reports** — displays the cumulative lift as a function of the depth for the model. The cumulative lift is given for each of the data roles. To examine other
statistics as a function of depth, use the drop-down menu in the upper right corner. Other statistics include lift, gain, % captured response, cumulative % captured response, % response, and cumulative % response. This result is displayed only if the target is a class variable.

- **ROC Reports** — displays the ROC (receiver operating characteristic) chart for a model, giving the sensitivity as a function of 1-specificity. The sensitivity is given for each of the data roles. To examine other statistics, use the drop-down menu in the upper right corner. Other statistics include accuracy and F1 score. This result is displayed only if the target is a class variable.

- **Fit Statistics** — displays a table of the fit statistics for the model, broken down by data role.

**Model Interpretability**

- **Input Relative Importance** — displays a table of the relative importance of the input variables, including variable level and label.

- **Partial Dependence** — displays the functional relationship between the input variable and the model prediction.

- **LIME Plots** — The LIME algorithm calculates a more easily interpretable linear model around a particular observation. During this process the data is perturbed around a particular observation to create a cluster that is centered on that particular observation. The linear model is created on the perturbed data set. These plots are available only when **Explain Individual Predictions** is selected.
  - **Local Instances** — displays the values for each input variable of the cluster center.
  - **Local Explanation** — displays the contributing weights for each predictor variable. Larger weights indicate that the input variable contributed more to the predicted outcome.
  - **Local Model** — displays the parameter estimates of the local linear model that is used to predict the cluster center.
  - **Prediction** — displays the predicted value of the cluster center for interval targets, and the predicted probability of each target outcome for class targets.

- **LIME Score Code** — displays the SAS code that Model Studio used to assign observations to clusters. This is displayed only when **Create score code for assigning clusters** is selected.

- **Individual Conditional Expectation** — displays the functional relationship between the input variable and the model prediction. The relationships for all cluster centers are displayed, which can help you determine whether there are any significant differences between the clusters or determine any input variable interactions.
Overview of Text Mining

The Text Mining node is a Data Mining Preprocessing node that enables you to process text data in a document collection. This data can be used to build predictive models in a distributed computing environment. Data is processed in two phases: text parsing and transformation. Text parsing processes textual data into a term-by-document frequency matrix. Transformations such as singular value decomposition (SVD) alter this matrix into a data set that is suitable for data mining purposes. A document collection with thousands of documents and terms can be represented in a compact and efficient form.

The Text Mining node supports Arabic, Chinese, Croatian, Czech, Danish, Dutch, English, Farsi, Finnish, French, German, Greek, Hebrew, Hindi, Hungarian, Indonesian, Italian, Japanese, Korean, Norwegian, Polish, Portuguese, Romanian, Russian, Slovak, Slovene, Spanish, Swedish, Tagalog, Thai, Turkish, and Vietnamese.

Text Mining Properties

Language Property

- Language — specifies the language to use for text parsing. Although all languages are displayed, the node fails if you attempt to use a language that you do not currently license. The default value is English.
Parsing Options

- **Include parts of speech** — specifies whether to use part-of-speech tagging in parsing. By default, this option is selected.
- **Extract noun groups** — specifies whether to extract noun groups in parsing. By default, this option is selected.
- **Extract entities** — specifies whether to extract entities in parsing. By default, this option is deselected.
- **Stem terms** — specifies whether to treat different terms with the same root as equivalent. By default, this option is selected.
- **Minimum number of documents** — specifies the threshold for the number of documents in which a term must occur to appear in the text analysis. Possible values range from 1 to 100. The default value is 4.

Topic Discovery

- **Automatically determine number of topics** — specifies whether to automatically determine the number of topics. When selected, topic generation is calculated based on a log function of the number of documents. By default, this option is selected.
- **Maximum topics** — specifies the maximum number of topics to discover. This option is available only if **Automatically determine number of topics** is deselected.

Text Mining Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Results**

- **Kept Terms** — displays 2,000 rows from a list of the terms that are kept in the document collection. This list table includes information about the terms, such as their role, attribute, and frequency.
- **Dropped Terms** — displays 2,000 rows from a list of the terms that have been dropped in the document collection. This list table includes information about the terms, such as their role, attribute, and frequency.
- **Terms: Role by Frequency** — displays a bar chart giving the total frequency of each occurrence of parsed terms in the document collection, broken down by term role.
- **Topics** — displays the topics created by the node and the unique ID for each topic. The **Text Mining** node creates topics based on groups of terms that occur
together in several documents. Each term-document pair is assigned a score for every topic in which it appears. Thresholds are then used to determine whether the association is strong enough to consider if that document or term belongs in the topic. As a result of this, terms and documents can belong to multiple topics.

- **Score Code** — displays SAS score code that was created by the node. The SAS score code can be used outside of the Model Studio environment to score new data.
- **Properties** — specifies the various properties selected before running the node.
- **Output** — displays the SAS output for the text mining run.
Overview of Transformations

The **Transformations** node is a Data Mining Preprocessing node. It enables making transformations by replacing an input variable with a function of that variable.

Such deterministic mathematical functions can be used to stabilize variances, remove nonlinearity, and correct non-normality in variables to improve the fit of your model.

**Note:** When you specify a transformation method for a variable on the Data pane, that method is always used, regardless of the property settings for this node.

Transformations Properties

Interval Inputs

- **Default interval inputs method** — specifies a default transformation method for all interval input variables that had no method assigned in the metadata table in the project data tab. Here are the possible values:
  - Best
  - Bucket binning
  - Centering
Ranking Criterion for Best Transformation

- **Criterion for interval target** — specifies the selection criterion that is used when you select **Best** in the **Default interval inputs method** property. Here are the possible values:
  - Average quantile kurtosis
  - Average quantile skewness
  - Moment kurtosis
  - Moment skewness
  - Pearson correlation with target

The default value is **Moment skewness**.

- **Criterion for binary target** — specifies the selection criterion that is used when you select **Best** in the **Default interval inputs method** property. Here are the possible values:
  - Anderson-Darling statistic with target
  - Average quantile kurtosis
  - Average quantile skewness
  - Cramer-Von Mises statistic with target
  - Kolmogorov-Smirnov statistic with target
  - Moment kurtosis
  - Moment skewness

The default value is **Moment skewness**.
Criterion for nominal target — specifies the selection criterion that is used when you select Best in the Default interval inputs method property. Here are the possible values:

- Average quantile kurtosis
- Average quantile skewness
- Moment kurtosis
- Moment skewness

The default value is Moment skewness.

Binning

If Bucket binning, Quantile binning, or Tree-based binning are selected as the Default interval inputs method, the following Binning configurations are available:

- Number of bins — specifies the number of nonmissing bins to create for interval inputs when applying a binning method. Possible values range from 2 to 50. The default value is 4.
- Create extra bins for values outside training range — specifies whether two extra bins, for values below the training minimum and above the training maximum, should be created when bucket or quantile binning. If not selected, these values are placed in the first or last bin, respectively. By default, this is deselected.
- Minimum tree bin size — specifies the minimum number of observations in a tree bin. Bins that do not meet or exceed the minimum number of observations are not split out. This option is available only when the Interval inputs method is Tree-based binning. The default value is 5.

Class Inputs

- Default class inputs method — specifies the default transformation method to use for class inputs. Possible values are Bin rare nominal levels, Level encoding, and None. The default value is None.
- Rare cutoff value percentage — specifies the cutoff percentage to determine which levels are considered rare. This option is available only when Bin rare nominal levels is selected. The default value is 0.5.
- Missing values binning treatment — specifies how missing values should be treated when binning. Possible values are Ignore and Separate. When Ignore is specified, the bin value is set to missing. When Separate is specified, missing values are assigned to their own bin. The default value is Separate.
- Reject original variables — specifies whether the original, non-transformed variables should be rejected, and thus unavailable for usage in proceeding nodes.
- Summary statistics — specifies whether summary statistics are generated for the transformed variables.
Transformations Results

After running the node, you can open the Results window by right-clicking the node and selecting Results from the pop-up menu.

Results
- **Input Variable Statistics** — displays information about input variables, including variable level, number of missing values, percent missing, minimum and maximum value, mean, midrange, importance standard deviation, and skewness.
- **Binning Summary** — displays information about the binned variables.
- **Transformed Variables Summary** — displays information about the transformed variables, including how they were transformed, the corresponding input variable, the formula applied, the variable level, type, and variable label.
- **Transformation Summary** — displays a table containing additional statistics on the results of the transformations applied to the training data. This table gives the following properties:
  - Transformed Variable
  - Method
  - Input Variable
  - Formula
  - Number of Missing Values
  - Minimum
  - Maximum
  - Mean
  - Standard Deviation
  - Skewness
  - Kurtosis
  - Variable Label
  
  This result is displayed only if you select Summary statistics. This result replaces Transformed Variables Summary.
- **Node Score Code** — displays SAS score code that was created by the node. The SAS score code can be used outside the Model Studio environment in custom user applications.
- **Properties** — specifies the various properties selected before running the node.
- **Output** — displays the SAS output of the transformation run. For the Transformations node, this includes the input variable statistics.
Overview of Variable Clustering

The Variable Clustering node is a Data Mining Preprocessing node. This node is a useful tool for data reduction because it finds the best variables for analysis. Variable clustering removes collinearity, decreases variable redundancy, and helps reveal the underlying structure of the input variables in a data set.

Variable clustering divides numeric variables into disjoint or hierarchical clusters. Variables in different clusters are conditionally independent given their own clusters. For each cluster that contains more than one variable, the variable that contributes the most to the variation in that cluster is chosen as the representative variable. All other variables are rejected.

Class variables are handled differently in Model Studio than they are in SAS Enterprise Miner. In SAS Enterprise Miner, the original Class variables are replaced by individual binary Class level variables, with one variable per Class level. And those variables are handled separately in the selection process. In Model Studio, the individual binary Class level variables are used in the clustering process, but the original Class variables are kept or dropped in the selection process. This selection depends on the variables that are included in a cluster, and the variable or variable level that is selected from each cluster.
Variable Clustering Properties

General Properties

- **Cluster representation** — specifies the identifier that is used to represent each cluster in proceeding nodes. Select **Best variable** to refer to the cluster by the name of the variable that best describes the cluster. Select **Cluster component** to refer to the cluster by a feature of the cluster.

- **Include class variables** — specifies whether to include class variables in the clustering process.

- **Export class level indicators** — specifies that the class level indicators are available to proceeding nodes.

- **Optimal cluster selection method** — specifies the method used to select the optimal cluster configuration. Select **Penalized log-likelihood** to calculate the Penalized value at each clustering step. The optimal cluster configuration determined is the clustering step with the lowest Penalized value. The default value is **None**.

- **Cluster Stopping Criteria**
  
  - **Number of clustering steps** — specifies the maximum number of clustering steps that are performed. Possible values range between 1 and 50. The default value is 6.
  
  - **Number of clusters lower threshold** — specifies the number of clusters below which the clustering process is stopped. There is one variable per cluster at the start of the variable clustering process. The number of clusters decreases with additional clustering steps. The default value is 1.
  
  - **Use default maximum number of variables per cluster** — specifies to use all available variables as the number of variables per cluster upper threshold. By default, this is selected.
  
  - **Number of variables per cluster upper threshold** — specifies the number of variables in a cluster above which the clustering process is stopped. There is one variable per cluster at the start of the variable clustering process. The number of variables per cluster increases with additional clustering steps. This option is available only if the **Use default maximum number of variables per cluster** option is deselected. The default value is 20.

- **Clustering RHO value** — specifies the value of RHO for determining the sequence of regularization parameters used in sequential clustering steps. The default value is 0.8.
Advanced Options

- **Maximum number of iterations** — specifies the maximum number of coordinate descent iterations for estimating the sparse precision covariance matrix. The default value is 1000.
- **Iteration convergence criterion** — specifies the minimal absolute tolerance at which an iteration stops. The default value is 0.0001.

Variable Clustering Results

After running the node, you can open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

**Results**

- **Clustered Variables Network** — displays a spatial map that gives the orientation and relative distance of clusters and cluster members. Cluster members with a stronger link are connected by a thicker line.
- **Clustered Variables Table** — displays a table that contains all of the clustered variables, listing their cluster IDs, variable labels, first principal components, and whether they were selected.
  
  *Note:* For class variables, the principal component might be blank. This is valid and expected.

- **Class Variable Mapping** — displays a table that describes how individual class variable levels are named in the clustering process.

- **Output Variable Metadata** — displays the metadata table for the output variables. This table contains the role of each output variable and an explanation of why that role was assigned.

- **Node Score Code** — displays the SAS score code that was created by the node. The SAS score code can be used outside of the Model Studio environment in custom user applications.

- **Properties** — displays the various properties that you can select before running the node.

- **Output** — displays the SAS output of a variable clustering run.
Variable Selection

Overview of Variable Selection

Many data mining databases have hundreds of potential model inputs (independent or explanatory variables) that can be used to predict the target (dependent or response variable). The Variable Selection node can assist you in reducing the number of inputs by rejecting input variables based on the selection results. Although rejected variables are passed to subsequent nodes in the pipeline, these variables are not used as model inputs by a successor modeling node.

The Variable Selection node is a Data Mining Preprocessing node that quickly identifies input variables, which are useful for predicting the target variable. The information-rich inputs can then be evaluated in more detail by one of the modeling nodes.

Variable Selection Properties

General Properties

- **Pre-screen Input Variables** — specifies whether to pre-screen input variables. If selected, this option will check the percentage of missing values for all
variables and the number of levels for the class variables. It rejects a variable if it exceeds the Maximum level or Maximum missing percent properties that are set below. By default, Pre-screen Input Variables is deselected.

- **Maximum level** — specifies the maximum number of levels for class variables. The default value is 50.
- **Maximum missing percent** — specifies the maximum percent for missing values. The default value is 50.
- **Combination criterion** — specifies the criterion used in keeping an input when you are using multiple variable selection methods. Here are the possible values:
  - Selected by a majority
  - Selected by a tie or majority
  - Selected by all
  - Selected by at least 1
  The default value is Selected by at least 1.

---

**Unsupervised Selection**

- **Unsupervised Selection** — specifies whether to perform unsupervised variable selection. By default, Unsupervised Selection is deselected. If selected, Unsupervised selection has the following options:
  - **Maximum steps** — specifies the maximum number of selection steps that are performed.
  - **Maximum variables** — specifies the maximum number of selected variables.
  - **Correlation statistics** — specifies Pearson correlation statistics. Possible values are Correlation, Covariance, and Sum of squares and crossproducts. The default value is Correlation.
  - **Cumulative variance cutoff** — specifies the cutoff value for the cumulative variance, which is explained by selected variables. The selection process stops when this fraction of the total variance can be explained by the selected variables. The default value is 1.
  - **Incremental variance cutoff** — specifies the cutoff value for incremental variance. The selection process stops when the increment of the explained variance is less than this fraction of the total variance. The default value is 0.001.
  - **Selection process** — specifies the process used for selecting variables when you are performing both unsupervised and supervised methods. Here are the possible values:
    - Combine with supervised method(s) — specifies to combine unsupervised selection with supervised methods via the Combination criterion.
    - Perform sequential selection — specifies to use unsupervised selection before one supervised method or combination of methods.

This property is ignored if no supervised methods are selected. The default value is Perform sequential selection.
Fast Supervised Selection — specifies whether to perform fast supervised selection. By default, Fast Supervised Selection is selected. If selected, Fast Supervised Selection has the following options:

- **Stop criterion** — specifies the criterion for stopping the selection process. Here are the possible values:
  - **AIC** — specifies Akaike’s Information Criterion. Smaller values indicate better models, and AIC values can become negative. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model.
  - **AICC** — specifies Corrected Akaike’s Information Criterion. This version of AIC adjusts the value to account for sample size. The result is that extra effects penalize AICC more than AIC. As the sample size increases, AICC and AIC converge.
  - **BIC** — specifies the Bayesian Information Criterion (BIC), also known as Schwarz’s Bayesian Criterion (SBC). BIC is an increasing function of the model’s residual sum of squares and the number of effects. Unexplained variations in the response variable and the number of effects increase the value of the BIC. As a result, a lower BIC implies either fewer explanatory variables, better fit, or both. BIC penalizes free parameters more strongly than AIC.
  - **None** — specifies that no stop criterion is used and that the selection process continues until all effects are in the model, or that a limit, described by one of the options below, is achieved.

  The default value is **BIC**.

- **Maximum steps** — specifies the maximum number of selection steps that are performed. The default value is 200.

- **Maximum variables** — specifies the maximum number of selected variables. The default value is 200.

- **Correlation statistics** — specifies Pearson correlation statistics. Possible values are Correlation, Covariance, and Sum of squares and crossproducts. The default value is Correlation.

- **Cumulative variance cutoff** — specifies the cutoff value for the cumulative variance, which is explained by selected variables. The selection process stops when this fraction of the total variance can be explained by the selected variables. The default value is 1.

- **Incremental variance cutoff** — specifies the cutoff value for the incremental variance. The selection process stops when the minimum increment of the explained variance is less than the fraction of the total variance. The default value is 0.001.
Linear Regression Selection

Linear Regression Selection — specifies whether to perform linear regression selection. By default, Linear Regression Selection is deselected. If selected, Linear Regression Selection has the following options:

- **Selection method** — specifies a model selection method. Here are the possible values:
  - Adaptive LASSO — specifies that training uses adaptive weights when applying L1 regularization to the regression methods coefficients. By default, ordinary least squares estimates of the model parameters are used as adaptive weights.
  - Backward — specifies that training is done by starting with all candidate effects in the model and removing effects until the Stay significance level or the stop criterion is met.
  - Forward — specifies that training is done by starting with no candidate effects in the model and adding effects until the Entry significance level or stop criterion is met.
  - Forward-swap — specifies that training is done by determining whether removing one effect and replacing it with the other improves the selection criterion.
  - LASSO — specifies that training is done using the group LASSO method—that is, adding and removing effects by a sequence of LASSO steps.
  - Stepwise — specifies that training is done as in the forward model but might remove effects already in the model.

The default value is Stepwise.

- **Effect-selection criterion** — specifies the criterion that the procedure uses to determine the order in which effects enter or leave at each step of the selection method. Here are the possible values:
  - Adjusted R-square — specifies the Adjusted R-squared value. The Adjusted R-Squared value attempts to account for the addition of more effect variables. Values can range from 0 to 1. Values closer to 1 are preferred.
  - AIC — specifies Akaike’s Information Criterion. Smaller values indicate better models, and AIC values can become negative. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution that is specified by the model.
  - AICC — specifies Corrected Akaike’s Information Criterion. This version of AIC adjusts the value to account for sample size. The result is that extra effects penalize AICC more than AIC. As the sample size increases, AICC and AIC converge.
  - Mallows’ Cp — specifies Mallows’ Cp value. Smaller values indicate better models, and Cp values can become negative.
  - R-square — specifies the R-squared value. R-squared value is an indicator of how well the model fits the data. R-squared values can range from 0 to 1. Values closer to 1 are preferred.
SBC (BIC) — specifies Schwarz's Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). SBC is an increasing function of the model’s residual sum of squares and the number of effects. Unexplained variations in the response variable and the number of effects increase the value of the SBC. As a result, a lower SBC implies either fewer explanatory variables, better fit, or both. SBC penalizes free parameters more strongly than AIC.

Significance level — specifies the standard statistical significance level criterion.

If the Selection method above is LASSO or Adaptive LASSO, this option is unavailable. The default value is SBC (BIC).

Selection-process stopping criterion — specifies the criterion to stop the selection process. Here are the possible values:

- Adjusted R-square — specifies the Adjusted R-squared value. The Adjusted R-Squared value attempts to account for the addition of more effect variables. Values can range from 0 to 1. Values closer to 1 are preferred.
- AIC — specifies Akaike’s Information Criterion. Smaller values indicate better models, and AIC values can become negative. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model.
- AICC — specifies Corrected Akaike’s Information Criterion. This version of AIC adjusts the value to account for sample size. The result is that extra effects penalize AICC more than AIC. As the sample size increases, AICC and AIC converge.
- Mallows' Cp — specifies Mallows’ Cp value. Smaller values indicate better models, and Cp values can become negative.
- None — specifies that the selection process continues until all the effects are in the model, or if a limit based on one of the options below is achieved.
- Predicted RSS — specifies that the selection process continues until the predicted residual sum of squares (RSS) starts to increase. This option is not valid with LASSO selection.
- SBC (BIC) — specifies Schwarz’s Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). SBC is an increasing function of the model’s residual sum of squares and the number of effects. Unexplained variations in the response variable and the number of effects increase the value of the SBC. As a result, a lower SBC implies either fewer explanatory variables, better fit, or both. SBC penalizes free parameters more strongly than AIC.
- Significance level — specifies the standard statistical significance level criterion.
- Validation ASE — average square error (ASE) of the model is computed using the validation data, and selection stops at the step where the ASE starts to increase. This method requires partitioned data.

The default value is SBC (BIC).

Model-selection criterion — specifies the criterion to choose the model from the list of models (at each step of the selection process) that yields the best value of the specified criterion. If the optimal value of the specified criterion occurs for models at more than one step, then the model that has the smallest number of parameters is chosen. Here are the possible values:
- **Adjusted R-square** — specifies the Adjusted R-squared value. The Adjusted R-Squared value attempts to account for the addition of more effect variables. Values can range from 0 to 1. The model that has the maximal value is chosen.

- **AIC** — specifies Akaike’s Information Criterion. Smaller values indicate better models, and AIC values can become negative. AIC is based on the Kullback-Leibler information measure of discrepancy between the true distribution of the response variable and the distribution specified by the model. The model that has the minimal AIC value is chosen.

- **AICC** — specifies Corrected Akaike’s Information Criterion. This version of AIC adjusts the value to account for sample size. The result is that extra effects penalize AICC more than AIC. As the sample size increases, AICC and AIC converge.

- **Mallows' Cp** — specifies Mallows' Cp value. Smaller values indicate better models, and the model with the minimal Cp is chosen.

- **Predicted RSS** — specifies that the model that has the minimal predicted residual sum of squares (RSS) is chosen. This option is not valid with LASSO selection.

- **SBC (BIC)** — specifies Schwarz’s Bayesian Criterion (SBC), also known as the Bayesian Information Criterion (BIC). SBC is an increasing function of the model’s residual sum of squares and the number of effects. Unexplained variations in the response variable and the number of effects increase the value of the SBC. As a result, a lower SBC implies either fewer explanatory variables, better fit, or both. SBC penalizes free parameters more strongly than AIC. The model that has the minimal SBC value is chosen.

- **Validation ASE** — average square error (ASE) of the model is computed using the validation data, and the model that has the minimal ASE is chosen. This method requires partitioned data.

If the **Selection method** is Forward-swap, this option is not available. The default value is SBC (BIC).

- **Entry significance level** — specifies the significance level for adding variables in forward and stepwise directions. The default value is 0.05.

- **Stay significance level** — specifies the significance level for removing variables in backward and stepwise selections. The default value is 0.05.

- **Maximum number of effects** — specifies the maximum number of effects in any model considered during the selection process. This option is ignored with the backward regression. If a model at some step of the selection process contains the specified maximum number of effects, then no candidate effects for addition are considered. Use the default value (0) to ignore this option.

- **Minimum number of effects** — specifies the minimum number of effects in any model considered during the backward selection process. The selection process terminates if a model at some step of the selection process contains the specified minimum number of effects. Use the default value (0) to ignore this option.

- **Maximum number of steps** — specifies the maximum number of selection steps that are performed. Use the default value (0) to ignore this option.

- **Suppress intercept** — specifies whether to suppress the intercept. By default, this is deselected.
Decision Tree Selection

- **Decision Tree Selection** — specifies whether to perform decision tree selection. By default, **Decision Tree Selection** is deselected. If selected, **Decision Tree Selection** has the following options:
  - **Splitting Options** — specifies the splitting criterion configurations:
    - **Class target criterion** — specifies the splitting criterion to use for determining best splits on inputs given a class target. Here are the possible values:
      - ChaID
      - Chi-square
      - Entropy
      - Gini
      - Information gain ratio
      The default value is **Entropy**.
    - **Interval target criterion** — specifies the splitting criterion to use for determining the best splits on inputs given an interval target. Here are the possible values:
      - ChaID
      - F test
      - Variance
      The default value is **Variance**.
    - **Significance level** — specifies the significance level for splitting criteria ChaID, Chi-Square, and F Test. The default value is 0.2.
    - **Bonferroni** — specifies whether a Bonferroni adjustment is applied to the top p-values for splitting criteria ChaID, Chi-Square, and F Test. By default, this is deselected.
    - **Maximum number of branches** — specifies the maximum number of branches that a splitting rule produces. The default value of 2 results in binary splits. Possible values range from 2 to 6.
    - **Maximum depth** — specifies the maximum number of generations in nodes. The original node, generation 0, is called the root node. The children of the root node are the first generation. Possible values range from 1 to 50. The default value is 10.
    - **Minimum leaf size** — specifies the smallest number of training observations that a leaf can have. The default value is 5.
    - **Missing values** — specifies how a splitting rule handles an observation with missing values. Here are the possible values:
      - Largest branch
      - Most correlated branch
      - Separate branch
- **Use in search**
  Missing values are either used as a value during the split search or assigned to a node based on this selection. The default value is **Use in search**.

- **Minimum missing use in search** — specifies the minimum number of missing values for a splitting variable to have for missing values to be treated as a separate level. This option is enabled when **Use in search** is specified for the **Missing values** property. The default value is 1.

- **Number of interval bins** — specifies the number of bins used for interval inputs. Bin size is \((\text{maximum value} - \text{minimum value})/\text{IntervalBins}\). The default value is 20.

- **Interval bin method** — specifies the method used to bin the interval input variables. The default value is **Bucket**.

- **Surrogate rules** — specifies the number of surrogate rules. The default value is 0.

- **Use input once** — if selected, specifies that no splitting rule will be based on an input variable used in a splitting rule of an ancestor node. This option is enabled when surrogate rules are not in effect. By default, this is deselected.

- **Pruning options** — specifies the pruning configurations:
  - **Subtree method** — specifies how to construct the subtree in terms of subtree pruning methods. Possible values include the following:
    - **C4.5** — the pruning is done with a C4.5 algorithm.
    - **Cost complexity** — the subtree with a minimum leaf-penalized ASE is chosen.
    - **Reduced error** — the smallest subtree with the best assessment value is chosen.

C4.5 is available only for class targets. For **Reduced error**, the assessment measure for class targets is Misclassification Rate. For interval targets, the assessment measure is ASE. The default value is **Cost complexity**.

- **Selection method** — specifies how to construct the subtree in terms of selection methods. Possible values include the following:
  - **Automatic** — specifies the appropriate subtree for the specified subtree pruning method.
  - **Largest** — specifies the full tree.
  - **N** — specifies the largest subtree with at most \(N\) leaves.

The default value is **Automatic**.

- **Confidence** — specifies the binomial distribution confidence level to use to determine the error rates of merged and split nodes. The default value is 0.25. This option is available only when **C4.5** is the pruning method.

- **Number of leaves** — specifies the number of leaves that are used in creating the subtree when the subtree selection method is set to **N**. The default value is 1.

- **Cross validation folds** — specifies the number of cross validation folds to use for cost-complexity pruning when there is no validation data. This option is enabled when the pruning selection method is **Automatic**, and the Create Validation from Training option is deselected. Possible values range from 2 to 20. The default value is 10.
- **1–SE rule** — specifies whether to perform the 1–SE rule when performing cross validated cost-complexity pruning. This option is enabled when the pruning selection method is **Automatic**, and the **Create Validation from Training** option is deselected. By default, this is deselected.

- **Seed** — specifies the random number seed used for cross validation cost-complexity. This option is enabled when the pruning selection method is **Automatic**, and the **Create Validation from Training** option is deselected. The default value is 12345.

- **Relative importance cutoff** — specifies the relative importance cutoff used to determine whether an input is rejected. Any input with a relative importance below this cutoff is rejected. The default value is 0.25.

---

**Forest Selection**

**Forest Selection** — specifies whether to perform forest selection. By default, **Forest Selection** is deselected. If selected, **Forest Selection** has the following options:

- **Tree-splitting Options** — specifies the tree-splitting configurations:
  
  - **Class target criterion** — specifies the splitting criterion to use for determining the best splits on inputs given a class target. Here are the possible values:
    
    - CHAID
    - Chi-square
    - Entropy
    - Gini
    - Information gain ratio

    The default value is **Information gain ratio**.

  - **Interval target criterion** — specifies the splitting criterion to use for determining the best splits on inputs given an interval target. Here are the possible values:
    
    - CHAID
    - F test
    - Variance

    The default value is **Variance**.

  - **Maximum number of branches** — specifies the maximum number of branches that a splitting rule produces. The default value of 2 results in binary splits. Possible values range from 2 to 5.

  - **Maximum depth** — specifies the maximum depth for each generated tree within the forest. Possible values range from 1 to 50. The default value is 20.

  - **Minimum leaf size** — specifies the smallest number of training observations that a new branch can have, expressed as a count of the available observations. Possible values range from 1 to 64. The default value is 5.

  - **Minimum missing use in search** — specifies a threshold for using missing values in the split search. The default value is 1.
- **Number of interval bins** — specifies the number of bins used for interval inputs. Bin size is defined as (maximum value – minimum value)/IntervalBins. The default value is 20.

- **Interval bin method** — specifies the method used to bin the interval input variables. The default value is **Bucket**.

- **In-bag sample proportion** — specifies the proportion of training observations with which to train a tree (the “in-bag” proportion). The default value is 0.6.

- **Use default number of inputs per split** — specifies whether to use the default number of inputs per split, which is the square root of the number of inputs. By default, this is selected.

- **Number of inputs per split** — specifies the number of input variables randomly sampled to use per split. This option is enabled when option **Use default number of inputs per split** is deselected. The default value is 100.

- **Number of inputs to subset with LOH** — specifies the number of input variables to further sample using LOH (where LOH is the method developed by Loh). When set to 0, no further sampling of inputs is performed. The default value is 0.

- **Seed** — specifies the seed for generating random numbers. This is used to select training observations for each tree and to select candidate variables in each node to split on. The default value is 12345.

  - **Number of trees** — specifies the number of trees in the forest. The default value is 100.

  - **Relative importance cutoff** — specifies the relative importance cutoff used to determine whether an input is rejected. Any input with a relative importance below this cutoff is rejected. The default value is 0.25.

---

**Gradient Boosting Selection**

**Gradient Boosting Selection** — specifies whether to perform gradient boosting. By default, **Gradient Boosting Selection** is deselected. If selected, **Gradient Boosting Selection** has the following options:

- **Tree-splitting Options** — specifies the tree-splitting configurations:
  - **Maximum number of branches** — specifies the maximum number of branches to consider at each node split in the tree. Possible values range from 2 to 5. The default value is 2.
  - **Maximum depth** — specifies the maximum number of generations of nodes. The original node, generation 0, is called the root node. The children of the root node are the first generation. Possible values range from 1 to 50. The default value is 6.
  - **Minimum leaf size** — specifies the smallest number of training observations that a leaf can have. The default value is 5.
  - **Minimum missing use in search** — specifies a threshold for using missing values in the split search.
  - **Number of interval bins** — specifies the number of bins in which to bin the interval input variables.
- **Interval bin method** — specifies the method used to bin the interval input variables. The default value is **Bucket**.
- **Use the default number of inputs per split** — specifies whether to use the default number of inputs per split, which is all of the inputs. By default, this option is selected.
- **Number of inputs per tree** — specifies the number of input variables randomly sampled to use per split. This option is used only if **Use the default number of inputs per split** is deselected. The default value is 100.
- **Seed** — specifies the seed for generating random numbers. The subsample rate property uses this value to select a training sample at each iteration. The default value is 12345.
  - **Number of trees** — specifies the number of terms in the boosting series. For interval and binary targets, the number of iterations equals the trees. For a nominal target, a separate tree is created for each target category in each iteration. The default value is 100.
  - **Learning rate** — specifies how much to reduce the prediction of each tree. The default value is 0.1.
  - **Subsample rate** — specifies the proportion of training observations with which to train a tree. A different training sample is taken in each iteration. Trees trained in the same iteration have the same training data. The default value is 0.5.
  - **L1 regularization** — specifies the L1 regularization parameter. The default value is 0.
  - **L2 regularization** — specifies the L2 regularization parameter. The default value is 0.
  - **Perform Early Stopping** — specifies whether to perform early stopping—that is, stopping training when the model starts to overfit. The training stops if the relative error in each of the \(N\) consecutive iterations (stagnation) is less than a threshold (tolerance). Early stopping cannot be used if there is no validation partition. By default, this is deselected.
    - **Stagnation** — specifies the number of consecutive iterations (\(N\)) for early stopping. The default value is 5.
    - **Tolerance** — specifies the threshold for early stopping. The default value is 0.
  - **Relative importance cutoff** — specifies the relative importance cutoff used to determine whether an input is rejected. Any input with a relative importance below this cutoff is rejected. The default value is 0.25.

**Create Validation from Training**

**Create Validation Sample from Training Data** — specifies whether a validation sample should be created from the incoming training data. This is recommended even if the data has already been partitioned so that only the training partition is used for variable selection, and the validation partition can be used for modeling. By default, this is selected. When selected, it has the following options:
  - **Validation proportion** — specifies the probability of a given record being selected for the validation set. The default value is 0.3.
Partition seed — specifies the partition seed to generate the sample that will be used for validation. The default value is 12345.

Variable Selection Results

After running the node, you can open the Results window by right-clicking the node and selecting Results from the pop-up menu.

Note: The results available for this node will vary based on the variable selection method.

Results

- **Variable Selection** — displays a table of variable metadata information about the data set submitted to the node. The table includes columns for variable name, label, level, role, and reason for rejection.

- **Variable Selection Combination Summary** — displays a table of input variables, whether they were selected or rejected by each method, and the final output role from the Combination criterion option. This result table is available when more than one model is selected, including the unsupervised model if combined with supervised models.

- **Cumulative Variance Explained** — displays a bar chart giving the proportion of variance explained as each variable is considered. The parameters are ordered by decreasing incremental variance. This result table is available for both Unsupervised Selection and Fast Selection.

- **Variance Explained Values** — displays the variance explained values at each iteration, as another variable or class variable level (parameter) is considered. This result table is available for both Unsupervised Selection and Fast Selection.

- **Parameter Estimates** — displays the statistical values obtained as a result of linear regression. This table includes effect, parameter, t-value, estimate, the p-value, and the standard error.

- **Tree Variable Importance** — displays a bar chart giving the relative importance of variables obtained via the decision tree model. This result table is available when Decision Tree is the only model selected.

- **Tree Variable Importance Values** — displays importance statistics of variables obtained via the decision tree method. This table includes variable name, train importance, importance standard deviation, relative importance, and count.

- **Forest Variable Importance** — displays a bar chart giving the relative importance of the variables obtained via the forest model. This result table is available when Forest is the only model selected.

- **Forest Variable Importance Values** — displays importance statistics of variables obtained via the forest method. This table includes variable name, train importance, importance standard deviation, and relative importance.

- **Gradient Boosting Variable Importance** — displays a bar chart giving the relative importance of the variables obtained via the gradient boosting model. This is available when Gradient Boosting is the only model selected.

- **Gradient Boosting Variable Importance Values** — displays importance statistics of variables obtained via the gradient boosting model. This table
includes variable name, train importance, importance standard deviation, and relative importance.

- **Properties** — specifies the various properties selected before running the node.
- **Output** — displays the SAS output of the variable selection run. The output given will vary based on the chosen variable selection method.