SAS® GLOBAL FORUM 2020 #SASGF

Paper SAS4485-2020

Automation in SAS® Visual Data Mining and Machine Learning

Wendy Czika, Christian Medins, and Radhikha Myneni, SAS Institute Inc.

ABSTRACT

Automated machine learning can help every data scientist, from the novice to the most experienced practitioner. This paper demonstrates the different levels of automation available in the Model Studio environment of SAS® Visual Data Mining and Machine Learning software. You can choose to have features automatically constructed or to automate the process of algorithm selection and hyperparameter tuning by using dedicated Model Studio nodes in the pipeline that represents your machine learning process. You can build on or edit a pipeline that includes these nodes, inserting your domain expertise into the process. Alternatively, you can ask the software to automatically build an entire pipeline that includes various feature engineering steps and predictive models, optimized for your specific data according to the assessment criterion of your choice. The included models are determined using hyperparameter tuning across multiple modeling algorithms. Not only do these automation techniques aid and accelerate the modeling process for beginning users, but they also relieve expert data scientists of the burden of iterating through various feature engineering steps, model hyperparameter values, and modeling algorithms, enabling them to focus on solving the problem at hand.

INTRODUCTION

Automated machine learning (commonly referred to as AutoML) involves, as you would expect, automating the tasks that are required for building a predictive model based on machine learning algorithms. These tasks include data cleansing and preprocessing, feature engineering, feature selection, model selection, and hyperparameter tuning, which can be tedious to perform manually. Platforms that provide this capability offer many benefits, including empowering data analysts or "citizen" data scientists by giving them a start at a machine learning workflow, as well as enabling advanced data scientists to spend more time on solutions to a problem rather than being bogged down with the repetitive tasks that they must do to determine the best workflow for their data. Note, however, that automation is not intended to replace the work of data scientists; ideally, there should be support for manual intervention by experts in these systems to allow the performance of tasks such as domain-specific feature engineering, which can be a critical component of predictive modeling. These systems should also be transparent with regard to the algorithms being used under the covers, so that users can be aware of and understand (and thus trust) the models being generated.

Although *Forbes* listed the growing prominence of automated machine learning as an artificial intelligence trend to watch for in 2019 (Janakiram MSV 2018), this is not new territory for SAS® software. SAS® Rapid Predictive Modeler software was first released in 2010, empowering users to develop predictive models by providing a process flow of automated data preparation and data mining tasks as part of SAS® Enterprise Miner™ software. The Model Studio visual interface for SAS Visual Data Mining and Machine Learning in SAS® Viya® is a modernized version of SAS Enterprise Miner that leverages the cloud-enabled, in-memory analytics engine of SAS® Cloud Analytics Services (CAS) to run algorithms in a distributed fashion (Wujek, Haller, and Wexler 2018). Model Studio provides automation like that of SAS Rapid Predictive Modeler, as well as more sophisticated automated machine learning that is presented in this paper. And automating your entire pipeline isn't the only degree of automation that is available. The paper presents various levels of automation that you can include in your machine learning pipeline-building process. You can do any combination of automated tasks, such as having the system determine variable roles and levels for you, create the best transformation for numeric features, generate multiple new features based on inputs and their profiles, and perform hyperparameter tuning. Alternatively, the whole process can be automated, through the Model Studio interface as well as using a REST API. The Machine Learning Pipeline Automation API can be integrated into your own applications to automatically build a pipeline, run it, and return the champion model, which can then be deployed. All these tasks are performed using SAS Visual Data Mining and Machine Learning under the hood.

AUTOMATION FROM THE MODEL STUDIO USER INTERFACE

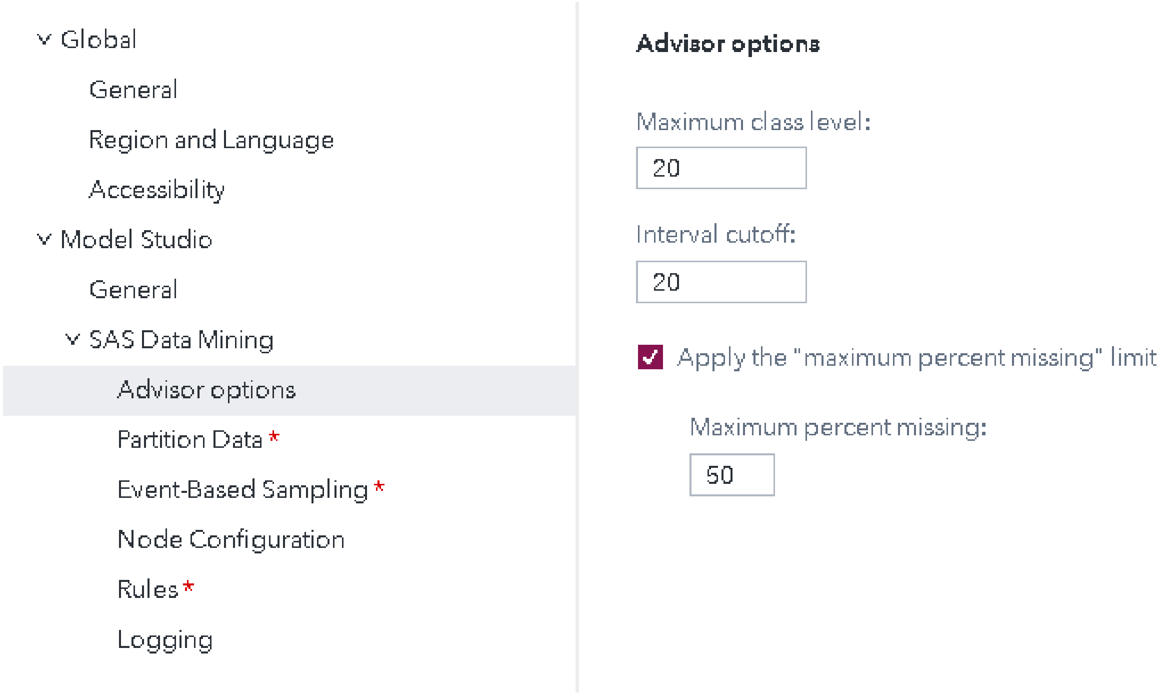
DATA PREPROCESSING AND FEATURE ENGINEERING

Model Studio provides plenty of flexibility for including your data preprocessing steps in your machine learning pipeline so that you can address data issues and make your inputs more effective in your model. These steps include nodes for imputing and transforming variables, filtering or replacing outliers, and reducing dimensionality. Even if you are manually building your pipeline with these nodes, there is automation taking place that you might be unaware of.

Data Advisor

One benefit that Model Studio provides to users is the data advisor. The data advisor serves to both prescreen your inputs and determine their measurement level—that is, whether an input should be treated as a categorical (nominal) input or a continuous (interval) input. Prescreening and measurement level determination are based on the following values, which you can control in the Advisor options (see Figure 1):

- Maximum class level – the maximum number of levels for a nominal input. This prescreening option serves to exclude a nominal input whose cardinality exceeds the specified value. Nominal inputs that exceed 254 class levels are automatically excluded regardless of the value specified here. Nominal inputs are excluded by setting their role in metadata to either "Rejected" or "ID." Character variables that exceed 254 levels are set to "ID," and all other nominal variables (character and numeric) that don't exceed 254 levels but are greater than the specified value are set to "Rejected."
- Interval cutoff – the minimum number of levels that a numeric variable must have in order to be classified as an interval input. A numeric variable that is below this number of levels is classified with one of three class levels (Nominal, Binary, Unary). Further analysis of nominal variables via the "Maximum class level" option serves to screen out high-cardinality variables.
- Maximum percent missing – the cutoff value for the missingness percentage allowed in a variable. This prescreening option serves to exclude inputs whose missingness percentage exceeds the specified value. These inputs are excluded by setting their role in metadata to "Rejected." You can choose not to apply this prescreening option by deselecting **Apply the "maximum percent missing"** limit.



Advisor options

Maximum class level: 20

Interval cutoff: 20

☑ Apply the "maximum percent missing" limit

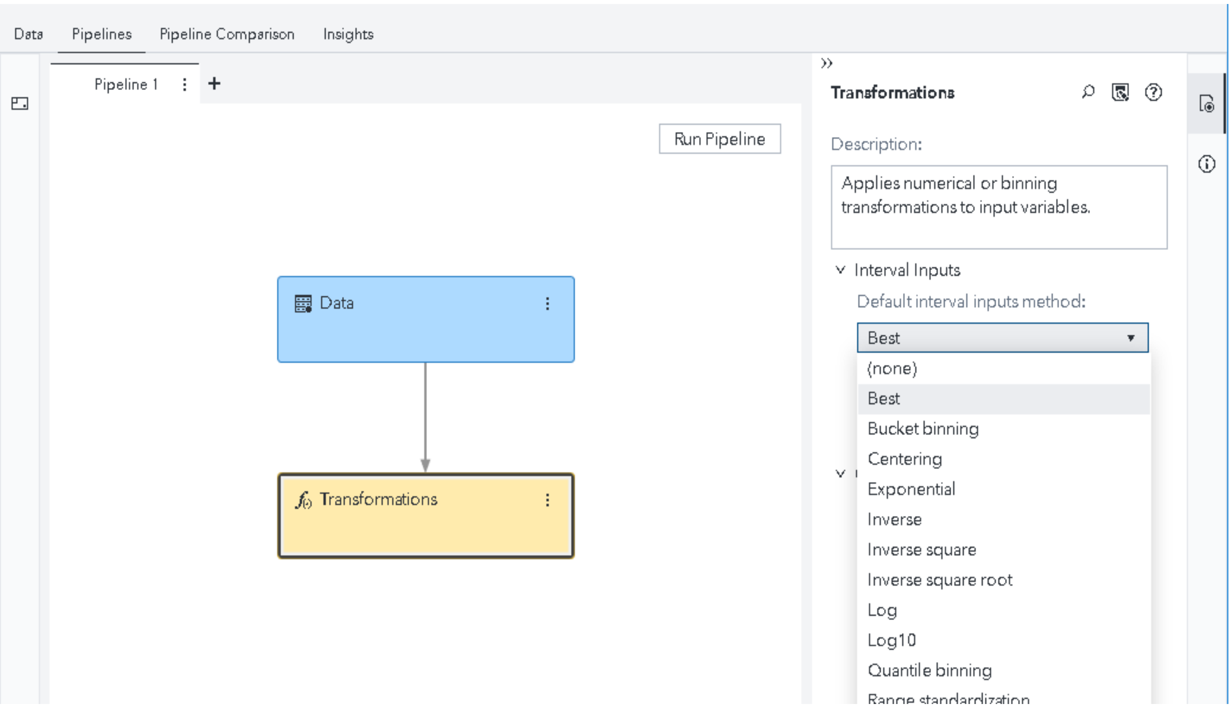
Maximum percent missing: 50

Figure 1. Advisor Options in User Settings

You can access the Advisor options globally via the User settings, which are the default settings for all projects, or locally for a specific project via the Advanced button in the New Project dialog box for that project. You can use all the decisions that the advisor automatically determines, or you can override any number of them to insert your business logic into what variables are used in your predictive models and how they are used. Don't want to override the same decisions in future projects? You can save any of the information or settings for any variable in the global metadata repository. This allows any new project that contains variables with the same names to automatically inherit these settings.

Best Transformation in Transformations Node

Another feature of Model Studio is the "Best" transformation method, available in the Transformations node. You might have inputs that you know are skewed and could benefit from a transformation, but you aren't sure which transformation would best normalize them. The Best transformation method includes typical Box-Cox transformations (x^{-2}, x^{-1}, $x^{-1/2}$, $x^{1/2}$, x^{2}) and others, such as Centering, Log, Log10, and Standardization. You can specify which ranking criterion to use to determine the best transformation, including target-based assessments for either binary or interval targets. In this ranking process, the transformations are applied to each input, the resulting values are analyzed according to the selected ranking criterion, and the top-ranked transformation is selected. Thus, the best transformation is automatically chosen for you. In the Transformations node (Figure 2), you access this method by selecting Best in the Default interval inputs method property. This selection is applied to all interval inputs that come into the node, unless the action is overridden by specific variable transformations identified via the Data tab or the Manage Variables node.



Description:

Applies numerical or binning transformations to input variables.

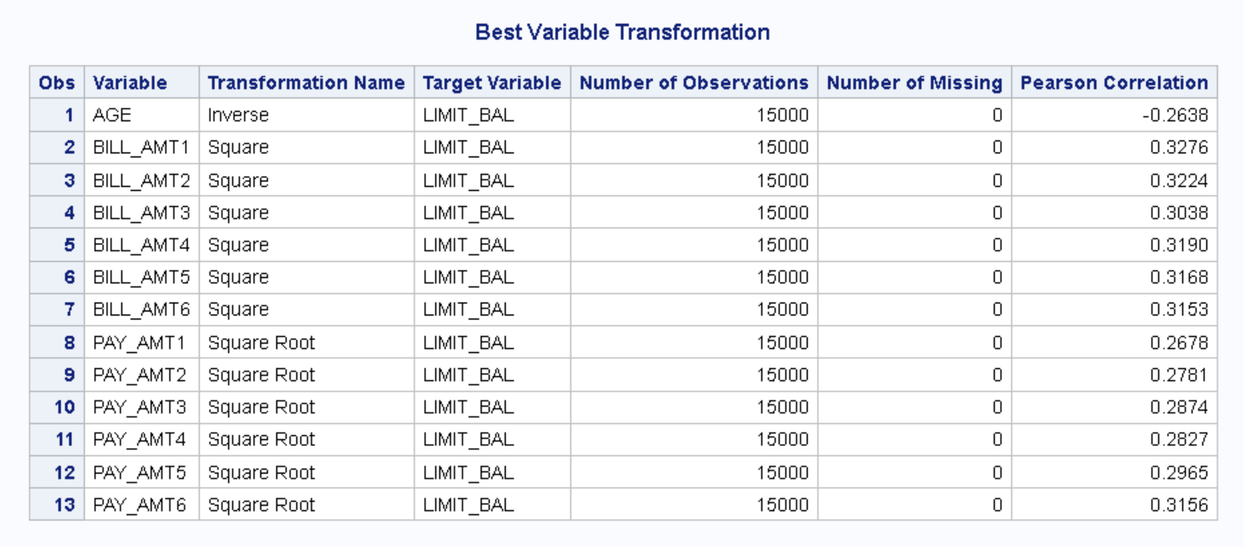
Default interval inputs method: Best

Figure 2. Best Transformation Method in the Transformations Node

This is the list of transformations available through the Best method:

- None – no transformation
- Centering – variable's value minus its mean
- Log
- Log10
- Square
- Square root
- Inverse square root
- Inverse
- Inverse square
- Range standardization – variable's value transformed onto the range 0 to 1
- Standardization – variable's value standardized using its mean and standard deviation

The example report shown in Figure 3 presents the list of input variables with their selected transformations. In this example, the Pearson correlation coefficient is the ranking criterion.



Best Variable Transformation

Obs	Variable	Transformation Name	Target Variable	Number of Observations	Number of Missing	Pearson Correlation
1	AGE	Inverse	LIMIT_BAL	15000	0	-0.2638
2	BILL_AMT1	Square	LIMIT_BAL	15000	0	0.3276
3	BILL_AMT2	Square	LIMIT_BAL	15000	0	0.3224
4	BILL_AMT3	Square	LIMIT_BAL	15000	0	0.3038
5	BILL_AMT4	Square	LIMIT_BAL	15000	0	0.3190
6	BILL_AMT5	Square	LIMIT_BAL	15000	0	0.3168
7	BILL_AMT6	Square	LIMIT_BAL	15000	0	0.3153
8	PAY_AMT1	Square Root	LIMIT_BAL	15000	0	0.2678
9	PAY_AMT2	Square Root	LIMIT_BAL	15000	0	0.2781
10	PAY_AMT3	Square Root	LIMIT_BAL	15000	0	0.2874
11	PAY_AMT4	Square Root	LIMIT_BAL	15000	0	0.2827
12	PAY_AMT5	Square Root	LIMIT_BAL	15000	0	0.2965
13	PAY_AMT6	Square Root	LIMIT_BAL	15000	0	0.3156

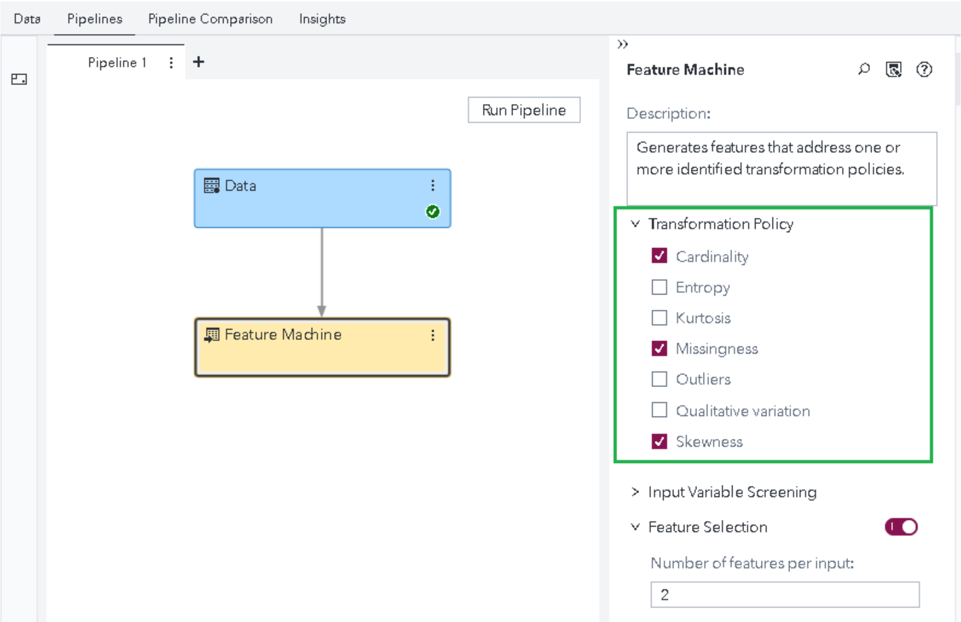
Figure 3. Best Transformation Example Output

For more detailed information about the Best transformation method, see "Appendix A: Best Transformation."

Feature Machine Node

Whereas you access the Best transformation method from within the Transformations node, an entire set of transformed features is automatically generated by the Feature Machine node. This node uses a three-step process to generate features. First, it explores the data such that input variables are grouped into categories that share the same statistical profile. This profile uses many variable attributes, including cardinality, coefficient of variation, entropy, qualitative variation, skewness, kurtosis, missingness, and outlier percentage. Next, the node screens input variables to identify variables to be excluded from feature generation or to be transformed in a specific way. Finally, the variables that survive the screening process are used to generate features, based on the exploration groupings and as required by the transformation policies specified in the node properties. Seven transformation policies are available for selection in the node (Figure 4), as shown in the following list. The features that are generated for each policy are designed to treat the data issue ascribed to that policy. Policies that are marked with an asterisk are enabled by default.

- *Cardinality – treatment of high cardinality
- Entropy – treatment of low entropy
- Kurtosis – treatment of high kurtosis
- *Missingness – treatment of missing values
- Outliers – treatment of outliers
- Qualitative variation – treatment of low indices of qualitative variation
- *Skewness – treatment of high skewness



Feature Machine

Description:

Generates features that address one or more identified transformation policies.

Transformation Policy

☑ Cardinality

☐ Entropy

☐ Kurtosis

☑ Missingness

☐ Outliers

☐ Qualitative variation

☑ Skewness

> Input Variable Screening

Feature Selection ●

Number of features per input: 2

Figure 4. Transformation Policy in the Feature Machine Node

The following large pool of feature transformations is used to address the transformation policies in the preceding list:

- Mode or median imputation
- Binary missing indicator
- Target encoding
- Level count encoding
- Level count rank encoding
- Rare level grouping
- Nominal level grouping using tree-based binning
- Tree-based binning
- Box-Cox power transformations
- Tukey’s ladder of power transformations
- Yeo-Johnson power transformations
- Winsorization

Multiple features can be generated per input variable, with the type and number of features determined by the transformation policies that are selected. The name of each feature defines the transformation steps that are applied for that feature. As an example, using the UCI Machine Learning Repository data (Dua and Graff 2019) on credit card client defaults (Yeh and Lien 2009), for a feature named "ho_winsor_BILL_AMT2," the input variable is BILL_AMT2, "ho" means that this feature is generated to address high outliers in the input, and "winsor" means that its distribution is transformed using the Winsor method. The features are subset (by default) with a feature selection process in which two features are kept (by default) per input. For any input variable that has a feature that is output, the input variable is dropped (rejected in metadata) by default. For more information about the Feature Machine node, see "Appendix B: Feature Machine Node."

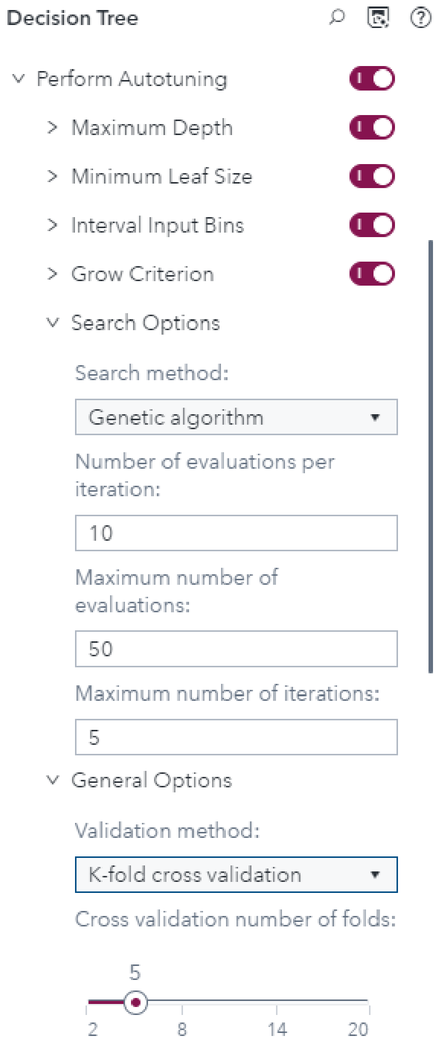
MODELING

When you have your data ready for modeling, there are multiple modeling algorithms to choose from, and within each one of them is an abundance of options and hyperparameters to set. Hyperparameters are properties that affect the training process, and thus they affect the quality of the resulting predictive model. Examples of hyperparameters include learning rate, regularization parameters, and the number of trees in a forest. Finding the optimal values for the hyperparameters for each model can involve a lot of manual trial and error. To find the optimal values in an automated manner, automatic hyperparameter tuning (autotuning) can perform an algorithmic search to determine the best model settings for your data. Autotuning (Koch et al. 2017) seeks to minimize or maximize a chosen objective function (typically a measure of model error) by using search methods such as Bayesian kriging, genetic algorithm, grid search, Latin hypercube sampling, and random search.

Supervised Learning Nodes: Autotuning Properties

To automate the identification of the hyperparameter values that give you the most accurate model, you can enable the Perform Autotuning property available in six of the Supervised Learning nodes in Model Studio: the Bayesian Network, Decision Tree, Forest, Gradient Boosting, Neural Network, and SVM nodes. Enabling this property requests that the optimal values of the available hyperparameters be chosen automatically. You can control which hyperparameters to autotune, the range of values to tune across, and the hyperparameter value to start with.

Figure 5 shows the autotuning properties of the Decision Tree node. You can see that there are four hyperparameters that can be autotuned for this modeling algorithm: Maximum Depth, Minimum Leaf Size, Interval Input Bins, and Grow Criterion. There are also other options related to the search method, and then a subset of general autotuning options are also shown. These options control the method of partitioning validation data, the objective function to optimize, and the limits for the time spent autotuning.



Decision Tree

Perform Autotuning

Maximum Depth

Minimum Leaf Size

Interval Input Bins

Grow Criterion

Search Options

Search method: Genetic algorithm

Number of evaluations per iteration: 10

Maximum number of evaluations: 50

Maximum number of iterations: 5

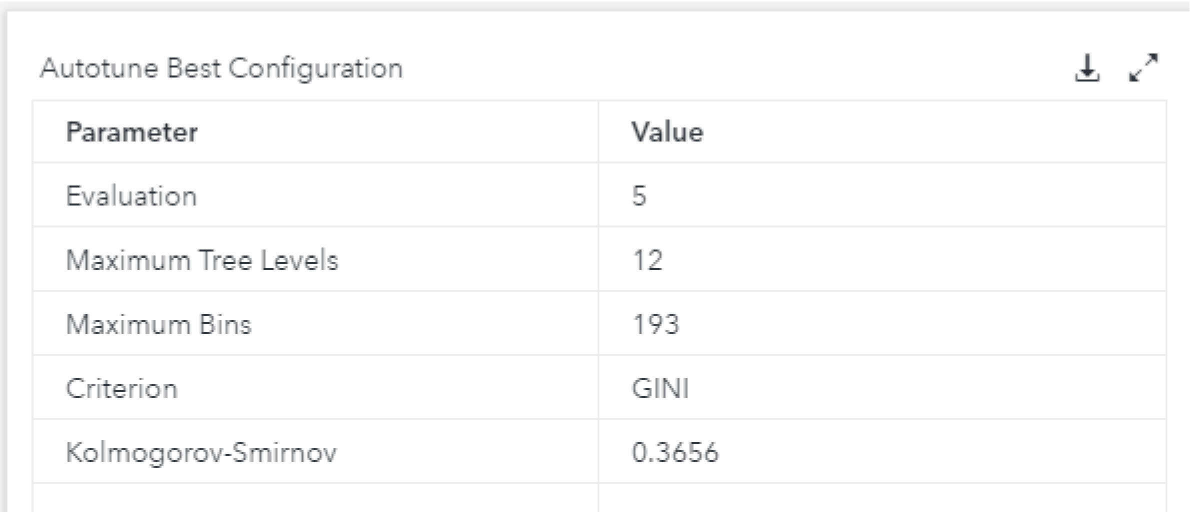
General Options

Validation method: K-fold cross validation

Cross validation number of folds: 5

Figure 5. Autotuning Properties of the Decision Tree Node

When the optimal set of hyperparameters has been determined on the basis of these properties, the model is trained using those hyperparameter values, giving you the results and assessment that you would get if you had set the values manually. The table in Figure 6 is included in the node results and shows the optimal values that were found for the hyperparameters, the evaluation at which they were used, and the value of the objective function (the Kolmogorov-Smirnov statistic in this example). Also included in the results for the node (not shown) is the "Autotune Results" table, which shows the sets of values of hyperparameters that were tried, the evaluation, the resulting objective function, and the evaluation time.



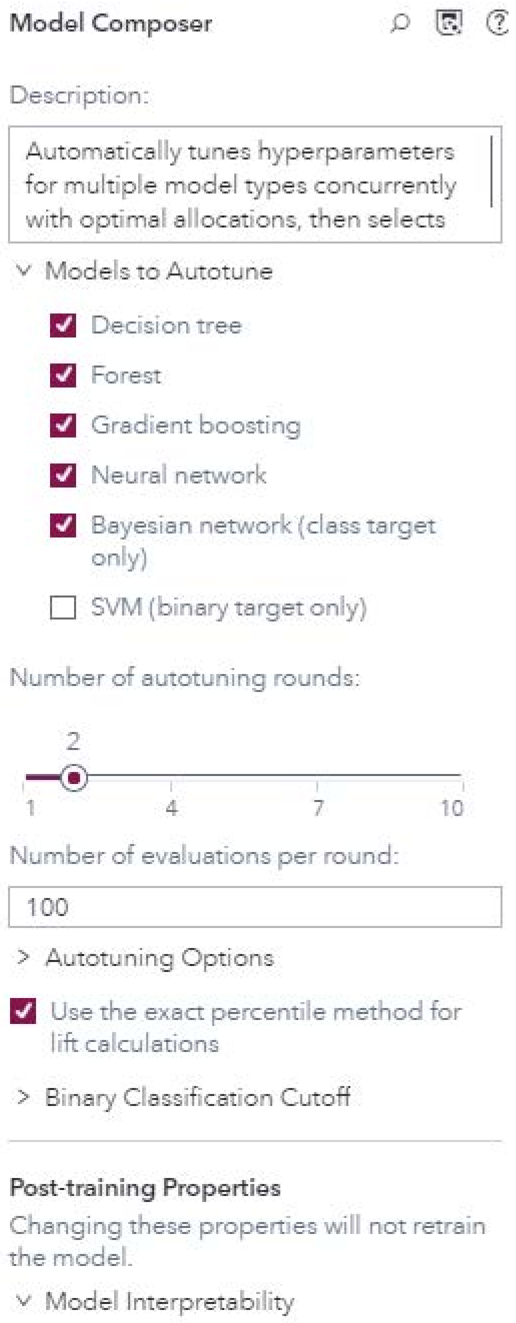
Autotune Best Configuration

Parameter	Value
Evaluation	5
Maximum Tree Levels	12
Maximum Bins	193
Criterion	GINI
Kolmogorov-Smirnov	0.3656

Figure 6. Optimal Values for Hyperparameters for the Decision Tree Node

Model Composer Node

New in SAS Visual Data Mining and Machine Learning 8.5 is the Model Composer node for performing what is known as combined algorithm selection and hyperparameter tuning (CASH). This feature adds another level of automation to autotuning, which can further enhance your productivity. It enables you to autotune various model types in parallel and performs multiple rounds of autotuning. The number of evaluations that are allocated to each model type in the rounds after the first round (where they are equally allocated) is determined by the accuracy and computational cost of each model type in previous rounds. When the rounds of autotuning are complete, the overall best model (according to your objective function) across model types that has the optimal set of hyperparameters is selected. The properties for the Model Composer node are shown in Figure 7, including the autotuning properties that are common to the Supervised Learning nodes mentioned previously.



Model Composer

Description:

Automatically tunes hyperparameters for multiple model types concurrently with optimal allocations, then selects

Models to Autotune

☑ Decision tree

☑ Forest

☑ Gradient boosting

☑ Neural network

☑ Bayesian network (class target only)

□ SVM (binary target only)

Number of autotuning rounds: 2

Number of evaluations per round: 100

Autotuning Options

☑ Use the exact percentile method for lift calculations

Binary Classification Cutoff

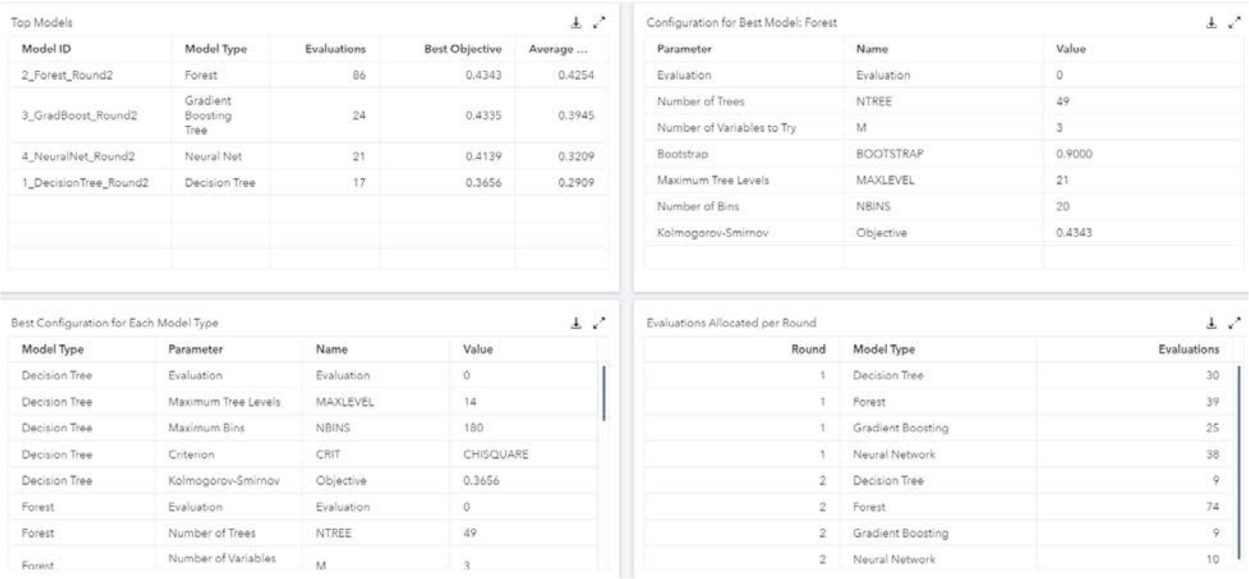
Post-training Properties

Changing these properties will not retrain the model.

Model Interpretability

Figure 7. Properties of the Model Composer Node

The Model Composer node determines the best modeling algorithm and its best configuration of hyperparameters based on your data and the properties you set. The score code from that particular model is provided, in either DATA step or analytic store (astore) form. If you request any of the model interpretability plots, they are included in the node results for the best overall model as well. You can also see the best configurations for the other model types. Figure 8 shows four of the tables that the node results include: the top models for each model type and their objective function value, the configuration for the top overall model, the best configuration for each model type, and the evaluations that are used per round for each model type.



Top Models

Model ID	Model Type	Evaluations	Best Objective	Average ...
2_Forest_Round2	Forest	86	0.4343	0.4254
3_GradBoost_Round2	Gradient Boosting Tree	24	0.4335	0.3945
4_NeuralNet_Round2	Neural Net	21	0.4139	0.3209
1_DecisionTree_Round2	Decision Tree	17	0.3656	0.2909

Configuration for Best Model: Forest

Parameter	Name	Value
Evaluation	Evaluation	0
Number of Trees	NTREE	49
Number of Variables to Try	M	3
Bootstrap	BOOTSTRAP	0.9000
Maximum Tree Levels	MAXLEVEL	21
Number of Bins	NBINS	20
Kolmogorov-Smirnov	Objective	0.4343

Best Configuration for Each Model Type

Model Type	Parameter	Name	Value
Decision Tree	Evaluation	Evaluation	0
Decision Tree	Maximum Tree Levels	MAXLEVEL	14
Decision Tree	Maximum Bins	NBINS	180
Decision Tree	Criterion	CRIT	CHISQUARE
Decision Tree	Kolmogorov-Smirnov	Objective	0.3656
Forest	Evaluation	Evaluation	0
Forest	Number of Trees	NTREE	49
Forest	Number of Variables	M	3

Evaluations Allocated per Round

Round	Model Type	Evaluations
1	Decision Tree	30
1	Forest	39
1	Gradient Boosting	25
1	Neural Network	38
2	Decision Tree	9
2	Forest	74
2	Gradient Boosting	9
2	Neural Network	10

Figure 8. Results of the Model Composer Node

AUTOMATING THE ENTIRE PIPELINE

So far, the paper has discussed different pieces of a machine learning pipeline that can be automated individually. For a fully automated experience, there are tools available to you in Model Studio to have an entire pipeline built for you, either using a static, prebuilt template or having a pipeline dynamically generated that is specifically tailored to your data.

Templates

Model Studio offers a broad array of prebuilt pipeline templates for predictive modeling, along with a template for feature engineering. These "getting started" tools give you the means to apply classical and modern modeling and feature engineering techniques to your data so that you can quickly integrate predictive analytic capabilities earlier in the decision-making cycle. You can choose from basic, intermediate, advanced, and advanced-with-autotuning predictive modeling templates. There are two versions of each template—one where the target is a class (categorical) variable, and one where the target is an interval variable.

- Basic template – includes a regression model (logistic for a class target, linear for an interval target) with imputation.
- Intermediate template – includes the Basic template and adds a decision tree model and a stepwise regression model (logistic for a class target, linear for an interval target) with imputation and variable selection.
- Advanced template – includes the Intermediate template and adds a gradient boosting model, a forest model, a neural network model with imputation and variable selection, and an ensemble model. (See Figure 9.)
- Advanced template with autotuning – includes the Advanced template and adds hyperparameter autotuning to the nonregression models.

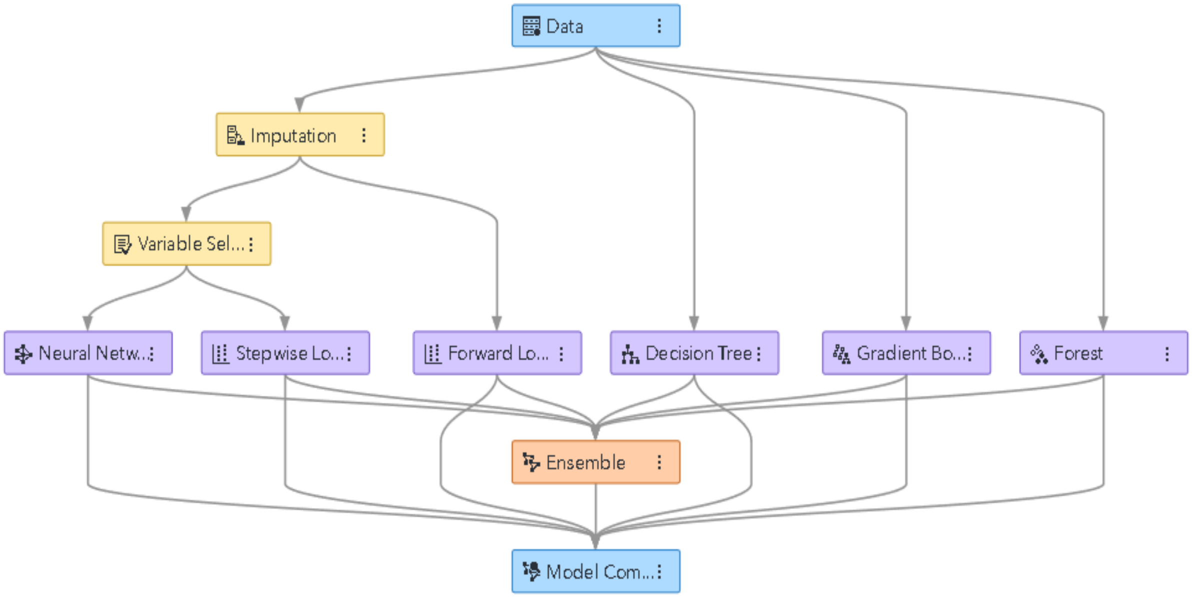


Figure 9. Advanced Template for Class Target

Also available is the feature engineering template. This template gives you the means to execute a prebuilt pipeline that transforms your data, producing different types of features with the goal of improving predictive modeling performance. The feature engineering template treats your data to handle high cardinality, high dimensionality, missingness, skewness, and variable selection, generating three different feature result sets that use a variety of feature engineering methods. (See Figure 10.)

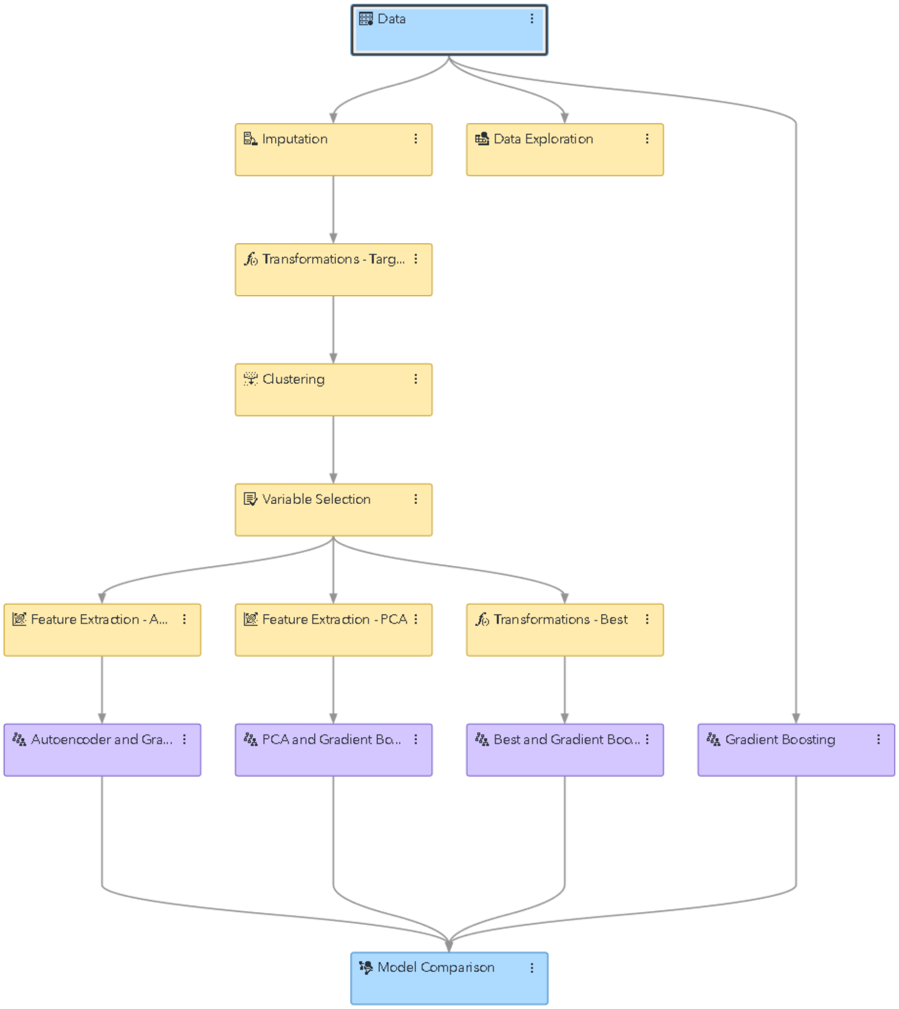


Figure 10. Feature Engineering Template

In the template pipeline, the first three nodes use the following feature engineering techniques:

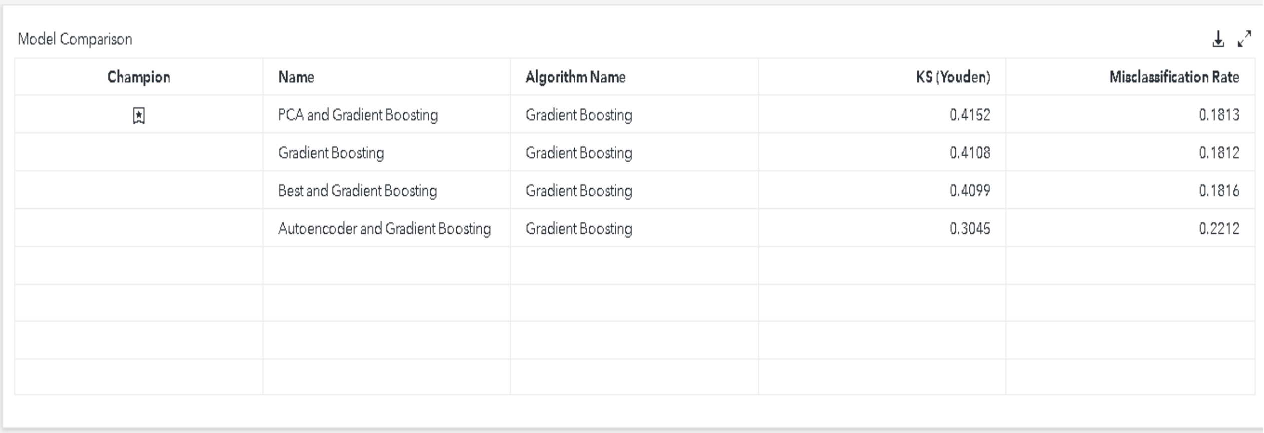
- Median imputation for interval inputs – treatment of missingness
- Target encoding for class inputs – class to interval feature mapping for treatment of high cardinality
- Observational clustering – clustering by using the *k*-prototypes algorithm to extract a cluster feature that can provide more information for predictive modeling

Next, a variable selection node takes as input the original input variables and the generated features from the three feature engineering methods in the preceding list, applying two supervised methods to select the features that provide the most information about the target. This subset of features is the source for the following three downstream feature engineering nodes, each applying a different feature engineering technique to generate a feature set:

- Transformations – Best: This node transforms all interval variables by using the Best transformation method. See the "Data Preprocessing and Feature Engineering" section of this paper for more information about this method.

- Feature Extraction – Autoencoder: This node uses the autoencoder technique to create new features from all input variables, both nominal and interval. Using the neural network algorithm, an autoencoder is an unsupervised learning technique whose objective is to learn a set of lower dimensional features that can be used to reconstruct the input data. Ten lower-dimensional features are extracted here. You can find the white paper that includes a discussion of this technique at https://support.sas.com/resources/papers/proceedings16/SAS3100-2016.pdf.
- Feature Extraction – PCA: This node implements principal component analysis (PCA) to extract five lower-dimensional features (principal components) from the set of 500 or fewer interval inputs. If there are more than 500 interval inputs in the data, this node uses the singular value decomposition (SVD) technique instead of PCA to extract five lower-dimensional features.

Each of these three feature engineering branches, along with the original untransformed data, is used as input to a gradient boosting modeling node so that you can compare the predictive performance among the four sets of data. After running this template against your data, view the Model Comparison node for the assessment results. In Figure 11, you can see that the PCA and Gradient Boosting model, whose source is the PCA feature set, has the best KS assessment result.



Model Comparison

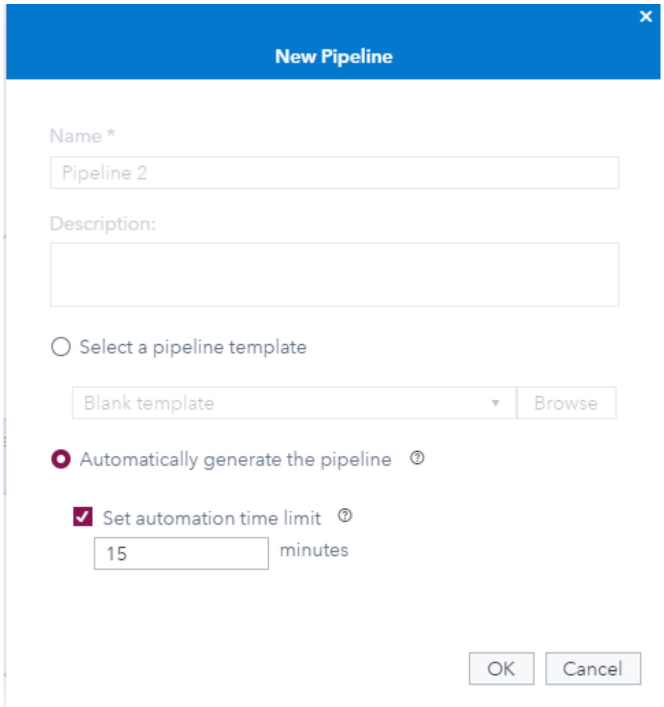
Champion	Name	Algorithm Name	KS (Youden)	Misclassification Rate
★	PCA and Gradient Boosting	Gradient Boosting	0.4152	0.1813
	Gradient Boosting	Gradient Boosting	0.4108	0.1812
	Best and Gradient Boosting	Gradient Boosting	0.4099	0.1816
	Autoencoder and Gradient Boosting	Gradient Boosting	0.3045	0.2212

Figure 11. Model Comparison Results from Feature Engineering Template

Automatically Generated Pipelines

One of the most exciting new features in SAS Visual Data Mining and Machine Learning 8.5 is the option to automatically generate the pipeline when you add a pipeline to your Model Studio project, as shown in Figure 12. This combines some of the automation concepts mentioned previously with intelligence being used behind the scenes to dynamically create the optimal pipeline for your data. It takes the Advanced templates a step further by attempting to improve on their champion models by using techniques such as these:

- applying the Best transformation method to interval inputs
- binning and/or encoding variables to create new features
- imputing and/or creating missing value indicators
- generating a set of new features for each input based on its characteristics
- selecting important inputs
- creating an ensemble model of two or more of the top branches of the pipeline



New Pipeline

Name *

Pipeline 2

Description:

○ Select a pipeline template

Blank template | Browse

● Automatically generate the pipeline

☑ Set automation time limit

15 minutes

OK | Cancel

Figure 12. New Pipeline Dialog Box with Option to Automatically Generate the Pipeline

Selecting this option gives you the pipeline that contains the top five models, in the time allotted if a time limit is set, based on the Model Comparison selection statistic and partition that you specify in Model Studio Project Settings. The top five models include a regression model (logistic for a class target or linear for an interval target), so you always have an interpretable model in case you need one for regulatory or other such purposes.

When the pipeline has been generated, the nodes that the pipeline includes and the properties that are set for them provide details of the data preprocessing steps that are being performed and the supervised learning algorithms that are being used; there is no "black box" aspect of this process. You can then run the pipeline in its locked state to get the overall pipeline champion model, shown in the Model Comparison node, based completely on the automation. Alternatively, you can unlock the pipeline so you can edit it to include your domain knowledge by adding, deleting, or modifying nodes in the pipeline. Other subsequent tasks for the pipeline that is built could include the following:

- performing autotuning in the Supervised Learning nodes, by turning on that property, if you have time to run and potentially improve your model by tuning its hyperparameters. This can be done automatically when you use the Enhanced modeling mode with the REST API (see the next section for more information).
- turning on model interpretability properties in the Supervised Learning nodes to get a better understanding of the inputs that drive the predictions in general, by using variable importance and partial dependency (PD) plots, or a better understanding of the predictions for specific observations in your data, by using ICE, LIME, and/or Kernel SHAP techniques
- comparing the champion from this pipeline with a pipeline that you have manually built, or with a pipeline from a template that you had previously saved or a colleague shared with you

Again, using the credit card clients data from the UCI Machine Learning Repository, in order to predict probability of default on credit card payments, Figure 13 presents an example of the automatically generated pipeline that uses the default Project Setting values. The best models that are found include a forest model that uses the Best transformation method for each interval input and the binning of rare levels for class inputs that are then target-encoded; a forest model with two new features generated per input by using the Feature Machine node; a forest model with no preprocessing, along with a logistic regression model included for interpretability; and an ensemble model that averages the posterior probabilities of these four models.

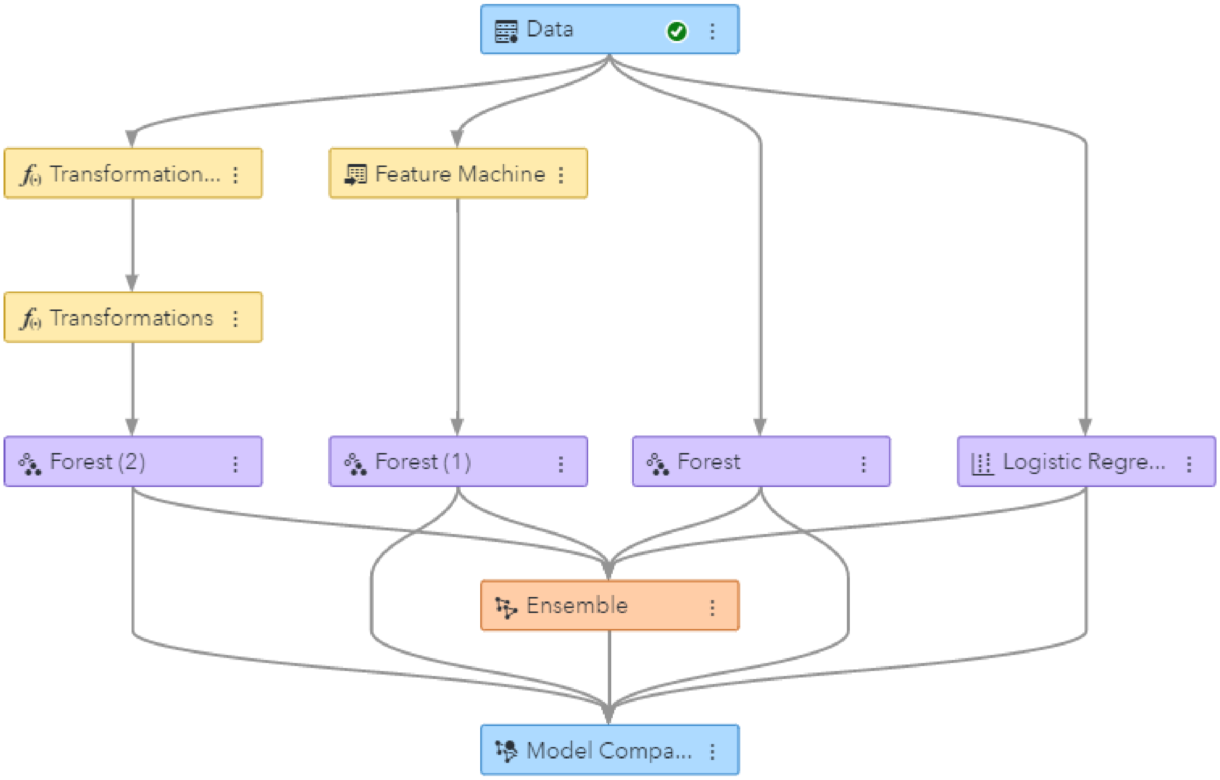


Figure 13. Automatically Generated Pipeline Using Default Project Settings

Just by changing the Class selection statistic option in Project Settings from the default of Kolmogorov-Smirnov statistic (KS) to Multiclass log loss and generating a new pipeline, you can see in Figure 14 how the pipeline contains different data preprocessing steps represented by the nodes, based on the different selection statistic. For example, weight-of-evidence (WOE) encoding is applied to high-cardinality inputs, different types of imputation are performed, and the best ensemble model includes only the forest models, not the logistic regression.

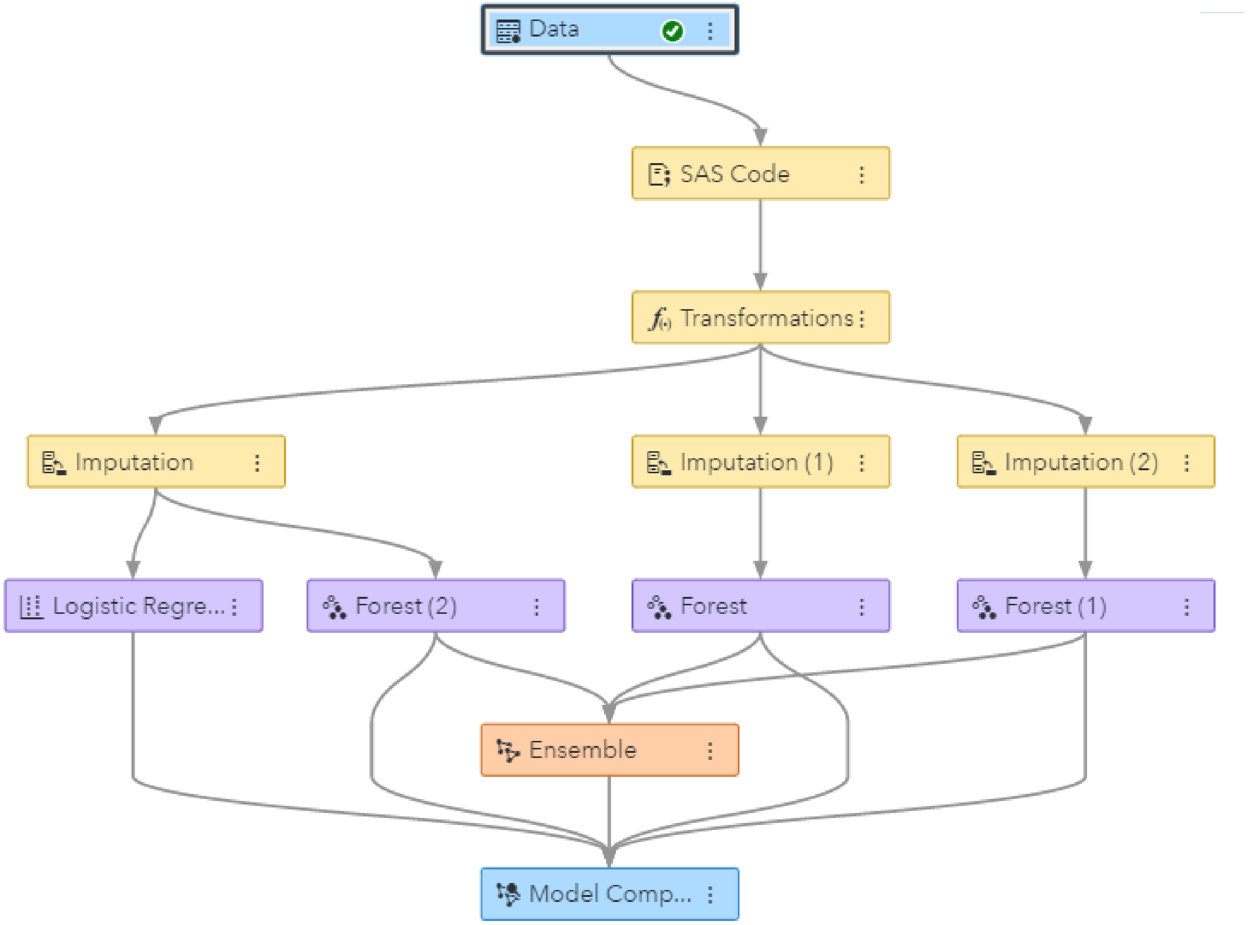


Figure 14. Automatically Generated Pipeline Using Modified Project Settings

AUTOMATICALLY GENERATED PIPELINES USING REST API

Another way to automatically generate pipelines in Model Studio is through RESTful interfaces. These interfaces enable you to embed this capability into your custom applications and drive it with a few clicks. The Machine Learning Pipeline Automation REST API is a set of endpoints that enables you to control more parameters than those available through the Model Studio user interface.

A REST request to create a project requires only the input data and a target variable in order to start the pipeline automation process. The process first creates a Model Studio project, runs the Advisor (to determine metadata), partitions and samples the input data (based on the parameters in the request or from Model Studio project settings), tries various models, and finally builds a pipeline by using top *n* (*n*=5 by default) models from the previous step.

When you create a project, you can modify the parameters shown in Table 1 and Table 2, which appear under settings and analyticsProjectAttributes in the request payload of the REST API.

Table 1. Parameters in Create Project Request under *Settings*

Parameter Name	Parameter Description
autorun	Specifies whether to run the pipeline after it is automatically built. Valid values are True and False. The default is True.
maxModelingTime	Specifies the modeling time in minutes. Valid values include any integer greater than zero.
modelingMode	Specifies the modes of operation. Valid values are Standard and Enhanced. If time permits, the Enhanced mode proceeds to model selection and hyperparameter optimization after running Standard mode. The default is Standard.
numberOfModels	Specifies the maximum number of final candidate models to create in the pipeline. Valid values include any integer greater than zero. The default is 5.

Table 2. Parameters in Create Project Request under *analyticsProjectAttributes*

Parameter Name	Parameter Description
classSelectionStatistic	Specifies the selection statistic for binary or nominal target. Valid values are ks, ase, c, response, cumulative_response, cumulative_lift, f1, fdr, fpr, gain, gini, lift, misclassification_event, mce, mcll, ks2, rase, and misclassificiation_cutoff. The default is ks (Kolmogorov-Smirnov statistic).
cutoffPercentage	Specifies the cutoff value to use for classifying binary target predictions. Valid values are 5, 10, 15, 20, 25, 30, . . . , 90, and 95. The default is 50.
intervalSelectionStatistic	Specifies the selection statistic for the interval target. Valid values are ase, rase, rmae, and rmsle. The default is ase (average square error).
numberofCutoffValues	Specifies the number of cutoff values to use in computing ROC-based measures. Valid values are 10, 20, 50, 100, 500, and 1000. The default is 20.
partitionEnabled	Specifies whether data should be partitioned. Valid values are True and False. The default is True. The data are partitioned according to Model Studio project settings.
samplingEnabled	Specifies whether to enable event-based sampling for a binary target. Event-based sampling undersamples a majority event and can be used when the event of interest is rare. Valid values are the strings AUTO, TRUE, and FALSE. The default is AUTO. When set to AUTO, event-based sampling is enabled when the target event rate is less than 5% or the gradient boosting model produces a misclassification rate greater than the target event rate.

samplingPercentage	Specifies the percentage of minority events in the final sample when event-based sampling is enabled using the samplingEnabled parameter. Valid values include integers between 1 and 99, inclusive. The default is 50.
selectionDepth	Specifies the depth to use for the class target in computing lift-based measures. Valid values are 5, 10, 15, and 20. The default is 10.
selectionPartition	Specifies the data partition to use for selecting the champion model. Valid values are default, train, validate, and test. The default is default. By default, the partition chosen in Model Studio project settings is used, which is test if unchanged by the user.
targetEventLevel	Specifies the target event level of interest. The default value is based on the project settings in Model Studio.
targetVariable	Specifies the target variable in input data. This is a required parameter.

An end-to-end example in Python to automatically generate pipelines using this REST API is provided in the example1.py file in the sas-viya-dmml-pipelines GitHub repository. Other languages or REST clients can also be used in place of Python.

In the example code, modify the setup parameters urlPrefix (host name of the microservice), authUser (user ID), authPw (password), datasetName (name of the input data loaded in CAS), target (target variable in data), and publicUri (URI of CAS library name where data reside) in the beginning section.

The example code performs the following steps:

1. Creates a Model Studio project for the automated pipeline.
2. Polls every 5 seconds to wait until the project state goes to completion. The project state starts in pending state and transitions to waiting, ready, modeling, constructingPipeline, runningPipeline, and finally to completed state.
3. Retrieves and prints the champion model information.
4. Publishes the champion model to the SAS® Micro Analytic Service destination.
5. Scores new data.
6. Retrains the project (when new training data become available).

Before you execute the example, the input data set (uci_default_credit_card) should be loaded into the corresponding CAS library that is specified in the publicUri parameter, and the location of analytic store model files needs to be configured for the SAS Micro Analytic Service destination as described in the documentation for SAS Viya Administration: Models.

For more information about the REST API and the authentication tokens needed for access, see the SAS developer site at https://developer.sas.com/apis/rest/MachineLearningPipeline. Information about configuring publishing destinations like Hadoop, Teradata, and CAS can be found in the documentation for SAS Viya Administration: Publishing Destinations.

CONCLUSION

The automation of key tasks that are involved in the building of predictive models is an integral part of Model Studio. These tasks, which include assigning variable roles, feature transformation and engineering, model hyperparameter tuning, combined model selection and hyperparameter tuning, using templates to create pipelines, and automatically generating pipelines, enable the data analyst and data scientist to hit the ground running when they are developing predictive models to solve their data problems. The automation of these complex and time-consuming tasks aids in democratizing machine learning and significantly reduces the time it takes to put models into production.

REFERENCES

Dua, D., and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Janakiram MSV (2018). "5 Artificial Intelligence Trends to Watch Out For in 2019." *Forbes*. Available at https://www.forbes.com/sites/janakirammsv/2018/12/09/5-artificial-intelligence-trends-to-watch-out-for-in-2019.

Koch, P., Wujek, B., Golovidov, O., and Gardner, S. (2017). "Automated Hyperparameter Tuning for Effective Machine Learning." In *Proceedings of the SAS Global Forum 2017 Conference*. Cary, NC: SAS Institute Inc. Available at https://support.sas.com/resources/papers/proceedings17/SAS0514-2017.pdf.

Wujek, B., Haller, S., and Wexler, J. (2018). "Navigating the Analytics Life Cycle with SAS Visual Data Mining and Machine Learning on SAS Viya." In *Proceedings of the SAS Global Forum 2018 Conference*. Cary, NC: SAS Institute Inc. Available at https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2246-2018.pdf.

Yeh, I.-C., and Lien, C.-H. (2009). "The Comparisons of Data Mining Techniques for the Predictive Accuracy of Probability of Default of Credit Card Clients." *Expert Systems with Applications* 36:2473–2480.

ACKNOWLEDGMENTS

The authors would like to thank Ed Huddleston for editing this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Wendy Czika	Christian Medins	Radhikha Myneni
wendy.czika@sas.com	christian.medins@sas.com	radhikha.myneni@sas.com

APPENDIX A: BEST TRANSFORMATION

This appendix contains additional information about the Best transformation method.

As mentioned in the main body of this paper, when you are processing the Best transformation for an input variable, a list of available transformations is applied, and the resulting values are analyzed to determine the best transformation according to a ranking criterion. Here is a list of the available ranking criteria, divided into three groups, followed by a brief description of each group:

Univariate statistics (target not used)

- Moment skewness
- Average quantile skewness
- Moment kurtosis
- Average quantile kurtosis

Empirical distribution comparison statistics (binary target)

- Anderson-Darling statistic with target
- Cramér–von Mises statistic with target
- Kolmogorov-Smirnov statistic (K-S) with target

Correlation statistics (interval target)

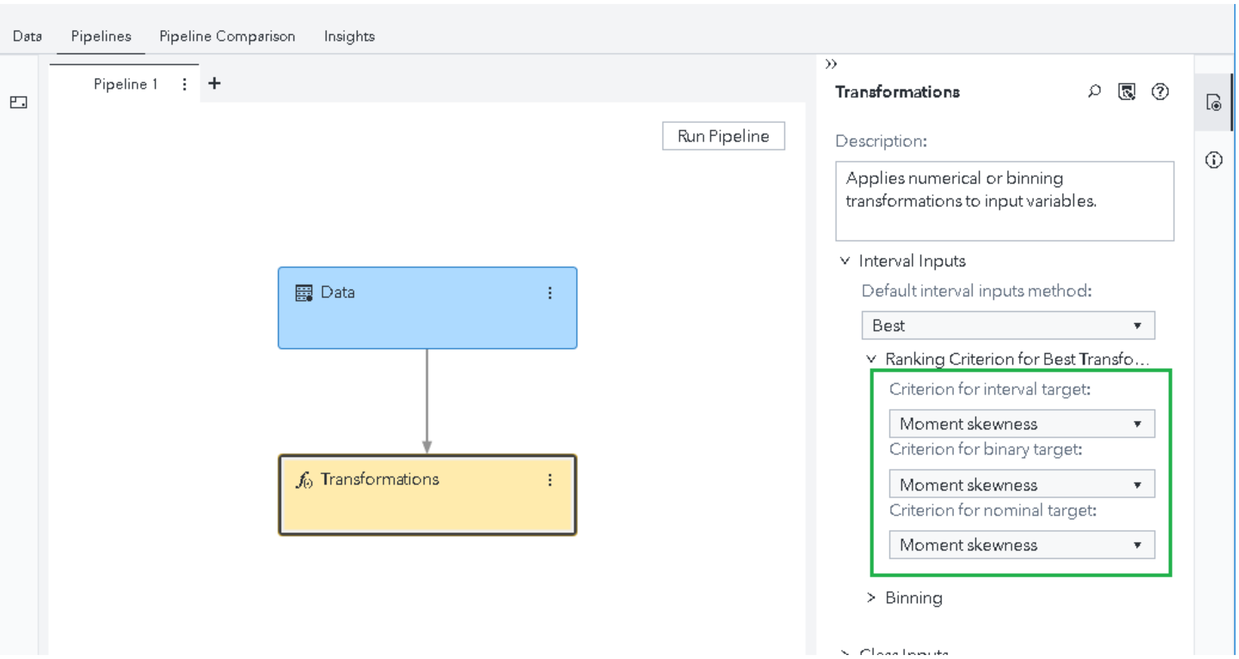
- Pearson correlation with target

Univariate statistics. You can choose one of these statistics to select the transformation that maximizes the normality of an input. Moment skewness and moment kurtosis are the standard skewness and kurtosis statistics. For average quantile skewness and average quantile kurtosis, ratios of the average quantile values in their formulas are used. They are considered robust, because they are significantly less sensitive to extreme outliers. For all four statistics, the transformation whose absolute value of the chosen statistic is closest to zero is selected. For more documentation about the average quantile statistics, see the article at http://www.cirano.qc.ca/realisations/grandes_conferences/methodes_econometriques/white.pdf. From the article, the average quantile skewness is $SK_3 = ((m - Q_2)/(AAD\_median))*3$ and the average quantile kurtosis is $KR_3 = ((U_{0.05} - L_{0.05})/(U_{0.5} - L_{0.5})) - 2.59$.

Empirical distribution comparison statistics. You can choose one of these statistics to compare the empirical distributions of a transformed input between the binary target groups. The predictive power of an input variable increases when there is greater distribution variation between target groups. For all three statistics, the transformation that has the greatest distribution variation is the one that has the maximum statistic value, and this is the transformation that is selected.

Correlation statistics. The Pearson correlation coefficient measures the linear correlation between each transformed input and the target. The transformation that has the highest correlation statistic is selected.

There are three properties in the Transformations node, within the Ranking Criterion for Best Transformation group, which enable you to select the ranking criterion: Criterion for interval target, Criterion for binary target, and Criterion for nominal target. (See Figure 15.)



Transformations

Description:

Applies numerical or binning transformations to input variables.

Interval Inputs

Default interval inputs method: Best

Ranking Criterion for Best Transfo…

Criterion for interval target: Moment skewness

Criterion for binary target: Moment skewness

Criterion for nominal target: Moment skewness

> Binning

> Class Inputs

Figure 15. Ranking Criterion for Best Transformation

The criteria are divided into these three properties. Which property to use depends on the level of the target variable (interval, binary, or nominal) in your data. Note that univariate statistics are always available as ranking criteria because they do not use the target variable.

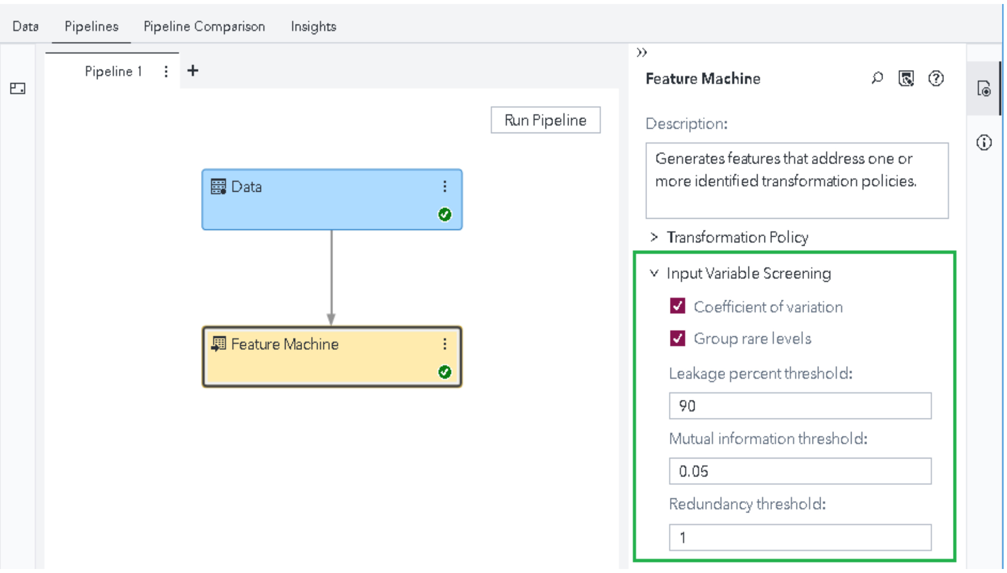
See this SAS Communities post for another resource about the Best transformation method.

APPENDIX B: FEATURE MACHINE NODE

This appendix contains additional information about the Feature Machine node.

For input variable screening, a number of options are available in the node (see Figure 16):

- Coefficient of variation – Identifies interval variables that have a low coefficient of variation (close to constant value). These variables are excluded from feature processing. This property is enabled by default.
- Group rare levels – Identifies nominal variables that have rare levels. These variables are transformed by rare level grouping. This property is enabled by default.
- Leakage percent threshold – Identifies variables that have a very high level of information about the target (leakage variables). Variables that exceed this threshold (target entropy reduction) are excluded from feature processing. The default value is 90.
- Mutual information threshold – Identifies variables that have a low level of information about the target (not informative). Variables that fall below this threshold are excluded from feature processing. The default value is 0.05.
- Redundancy threshold – Identifies variables that are redundant (highly correlated). If the symmetric uncertainty coefficient, a measure of nominal association for two variables, exceeds this threshold, the variable that has less information about the target is excluded from feature processing. The default value is 1. With this default value, redundancy screening is not enabled. You can enable this property by specifying a value less than 1.



Description:

Generates features that address one or more identified transformation policies.

> Transformation Policy

∨ Input Variable Screening

☑ Coefficient of variation

☑ Group rare levels

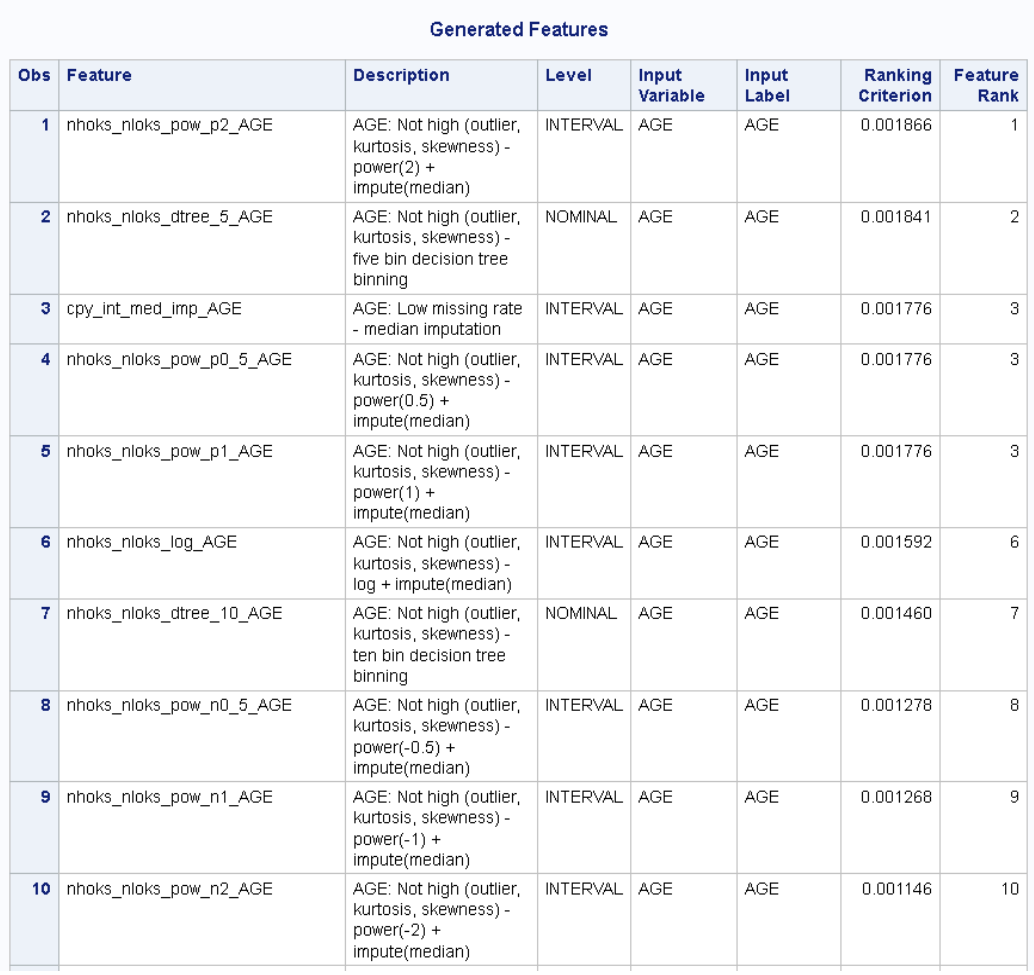
Leakage percent threshold: 90

Mutual information threshold: 0.05

Redundancy threshold: 1

Figure 16. Input Variable Screening in the Feature Machine Node

Feature Selection is enabled by default to subset the list of multiple generated features. All the features for an input variable are ranked using the symmetric uncertainty (SU) coefficient, and the top-ranked features (per input) are selected and output from the node. When it is not enabled, all generated features are output from the node. For Feature Selection, you specify the number of selected features per input by using the Number of features per input property (Default=2). This value is compared to the ranking values to determine the selected features. If the ranking of features results in a tie (that is, two or more features have the same SU value), this can result in more features being selected for an input than specified. In Figure 17, which lists the generated features for the input variable AGE, the third, fourth, and fifth features are tied at Feature Rank 3. In this example, if the specified number of features per input is 1, the first feature is kept. If the specified number of features is 2, the first and second features are kept. However, if the specified number of features is 3, the first through fifth features are kept, since features 3, 4, and 5 all have Feature Rank 3.



Generated Features

Obs	Feature	Description	Level	Input Variable	Input Label	Ranking Criterion	Feature Rank
1	nhoks_nloks_pow_p2_AGE	AGE: Not high (outlier, kurtosis, skewness) - power(2) + impute(median)	INTERVAL	AGE	AGE	0.001866	1
2	nhoks_nloks_dtree_5_AGE	AGE: Not high (outlier, kurtosis, skewness) - five bin decision tree binning	NOMINAL	AGE	AGE	0.001841	2
3	cpy_int_med_imp_AGE	AGE: Low missing rate - median imputation	INTERVAL	AGE	AGE	0.001776	3
4	nhoks_nloks_pow_p0_5_AGE	AGE: Not high (outlier, kurtosis, skewness) - power(0.5) + impute(median)	INTERVAL	AGE	AGE	0.001776	3
5	nhoks_nloks_pow_p1_AGE	AGE: Not high (outlier, kurtosis, skewness) - power(1) + impute(median)	INTERVAL	AGE	AGE	0.001776	3
6	nhoks_nloks_log_AGE	AGE: Not high (outlier, kurtosis, skewness) - log + impute(median)	INTERVAL	AGE	AGE	0.001592	6
7	nhoks_nloks_dtree_10_AGE	AGE: Not high (outlier, kurtosis, skewness) - ten bin decision tree binning	NOMINAL	AGE	AGE	0.001460	7
8	nhoks_nloks_pow_n0_5_AGE	AGE: Not high (outlier, kurtosis, skewness) - power(-0.5) + impute(median)	INTERVAL	AGE	AGE	0.001278	8
9	nhoks_nloks_pow_n1_AGE	AGE: Not high (outlier, kurtosis, skewness) - power(-1) + impute(median)	INTERVAL	AGE	AGE	0.001268	9
10	nhoks_nloks_pow_n2_AGE	AGE: Not high (outlier, kurtosis, skewness) - power(-2) + impute(median)	INTERVAL	AGE	AGE	0.001146	10

Figure 17. Feature Ranking Example in the Feature Machine Node

The following lists present the set of available feature transformations, grouped by transformation policy. For more information about these transformations, see the SAS Visual Data Mining and Machine Learning: Programming Guide.

Cardinality (Nominal to Interval Transformations)

- hc_tar_mean – Mean target encoding
- hc_tar_min – Minimum target encoding
- hc_tar_max – Maximum target encoding
- hc_tar_frq_rat – Frequency ratio target encoding
- hc_tar_woe – Weight-of-evidence target encoding
- hc_tar_evt_prob – Event probability target encoding
- hc_lbl_cnt – Level count rank
- hc_cnt – Level count
- hc_cnt_log – Level count followed by log transformation

Entropy, Qualitative Variation

- grp_rare1 – Mode imputation and group rare levels
- grp_rare2 – Missing level and group rare levels
- lchehi_lab – Label encoding
- lcnhenhi_grp_rare – Group rare levels
- lcnhenhi_rtree5 – Five-bin regression tree binning
- lcnhenhi_rtree10 – Ten-bin regression tree binning
- lcnhenhi_dtree5 – Five-bin decision tree binning
- lcnhenhi_dtree10 – Ten-bin decision tree binning
- lcnhenhi_woe5 – Five-bin weight-of-evidence binning
- lcnhenhi_woe10 – Ten-bin weight-of-evidence binning

Kurtosis

- hk_yj – Yeo-Johnson power transformations with parameters −2, −1, 0, 1, 2
- hk_dtree_disct5 – Five-bin decision tree binning
- hk_dtree_disct10 – Ten-bin decision tree binning
- hk_rtree_disct5 – Five-bin regression tree binning
- hk_rtree_disct10 – Ten-bin regression tree binning

Missingness

- cpy_int_med_imp – Median imputation
- cpy_nom_mode_imp_lab – Mode imputation and label encoding
- cpy_nom_miss_lev_lab – Missing level and label encoding
- miss_ind – Missing indicator

Outliers

- ho_winsor – Winsorization
- ho_quan_disct5 – Five-bin quantile binning
- ho_quan_disct10 – Ten-bin quantile binning
- ho_dtree_disct5 – Five-bin decision tree binning
- ho_dtree_disct10 – Ten-bin decision tree binning
- ho_rtree_disct5 – Five-bin regression tree binning
- ho_rtree_disct10 – Ten-bin regression tree binning

Skewness

- hs_bc – Box-Cox power transformations with parameters −2, −1, 0, 1, 2
- hs_dtree_disct5 – Five-bin decision tree binning
- hs_dtree_disct10 – Ten-bin decision tree binning
- hs_rtree_disct5 – Five-bin regression tree binning
- hs_rtree_disct10 – Ten-bin regression tree binning

Kurtosis, Outliers, Skewness (Low- or Medium-Rated Values)

- nhoks_nloks_pow, nhoks_nloks_log – Tukey's ladder of power transformations with parameters −2, −1, −0.5, 0, 0.5, 1, 2
- nhoks_nloks_dtree5 – Five-bin decision tree binning
- nhoks_nloks_dtree10 – Ten-bin decision tree binning
- nhoks_nloks_rtree5 – Five-bin regression tree binning
- nhoks_nloks_rtree10 – Ten-bin regression tree binning

Kurtosis, Outliers, Skewness (Low-Rated Values)

- all_l_oks_dtree5 – Five-bin decision tree binning
- all_l_oks_dtree10 – Ten-bin decision tree binning
- all_l_oks_rtree5 – Five-bin regression tree binning
- all_l_oks_rtree10 – Ten-bin regression tree binning

After running the Feature Machine node, click to open the results from the context pop-up menu. When Feature Selection is enabled, the Selected Features report is displayed. This report contains the list of selected features, sorted by input variable, feature rank, and feature, that are output by the node. Downstream nodes receive only these features. The Description column, which describes the feature, includes the input variable followed by a colon and the data quality issue, which is followed by a hyphen and the transformation method. An example feature description is "AGE: Not high (outlier, kurtosis, skewness) – power(2) + impute(median)." Its expanded meaning is as follows: This feature is for the input variable AGE. It addresses the data quality where one or more of the statistical measures for outliers, kurtosis, and skewness have a medium value but none of them have a high value. It is transformed by taking the square and imputing the median value. When Feature Selection is not enabled, the Generated Features report is displayed instead. This report contains the list of all generated features, which are output by the node for input into downstream nodes. These are sorted by input variable and feature. The Output report contains all features that the Feature Machine node generates, regardless of whether Feature Selection is enabled.