Paper SAS4572-2020

What's New in SAS® Visual Data Mining and Machine Learning: From a Programmer's Perspective

Tao Wang, SAS Institute Inc.

ABSTRACT

The latest releases of SAS® Visual Data Mining and Machine Learning software provide a comprehensive set of data mining and machine learning capabilities. The latest features in this product suite include enhanced principal component analysis through the new Kernel PCA action set, the new SEMISUPLEARN procedure to "impute" missing target labels in your training data automatically, and the new SPARSEML procedure for discovering insights from sparse data that now exist in many business domains. There are also many new features in existing action sets and procedures, such as the Image action set and the SVMACHINE procedure, and much more. This paper provides an overview of some hot topics and the latest features of the SAS Visual Data Mining and Machine Learning product suite and includes use cases and examples that demonstrate how to take full advantage of product capabilities from the programmer's perspective.

INTRODUCTION

Analytics starts with data and follows the data. The latest releases of SAS® Visual Data Mining and Machine Learning provides many new features that can help data engineers, data scientists, and machine learning developers better group and label their data, build better models, and make better decisions. This paper provides an overview of many of the new SAS Visual Data Mining and Machine Learning programming features so that programmers like you can be more familiar with these new features and leverage them for your business needs.

SOME CONCEPTS

This section defines some important concepts that will be used in this paper.

SAS PLATFORM

The SAS Platform [1] is a powerful analytics platform that is used to orchestrate your entire analytics journey. With the SAS Platform you can access data of any format from any source to build better models and get a better return on your analytics investments.

SAS VIYA®

SAS Viya [3] is an extension of the SAS Platform. SAS Viya has unified environment management, advanced scheduling, and fast and distributed in-memory processing capabilities. Its resilient architecture with guaranteed failover can automatically detect server failure across clusters. This ensures high availability for uninterrupted processing and automated recovery with self-healing mechanisms.

SAS CAS (CLOUD ANALYTICS SERVICES)

SAS CAS [2] is a cloud-based analytic engine running on top of SAS Viya. The CAS server is fault tolerant and has interfaces with third-party software such as Python, Java, Lua, and REST API.

SAS VISUAL DATA MINING AND MACHINE LEARNING

SAS Visual Data Mining and Machine Learning [4] runs in SAS Viya and includes a web-based software package called Model Studio. It combines data wrangling, data sampling, feature engineering, and exploration techniques with modern statistical, data mining, machine learning, predictive modeling, and model assessment techniques in a single, scalable in-memory processing environment. SAS Visual Data Mining and Machine Learning enables you to solve complex analytical problems with a comprehensive visual interface that handles all tasks in the analytics life cycle.

THE SAS CODE NODE IN SAS VISUAL DATA MINING AND MACHINE LEARNING

In Model Studio, the visual interface of SAS Visual Data Mining and Machine Learning, there is a SAS Code node that is a programming environment. You can execute the sample code that is presented in this paper in this node.

CAS ACTIONS AND PROCEDURES

CAS actions [5] perform a single task in the CAS server. From a SAS programmer's perspective, CAS actions are the building blocks that are used to build complex SAS programs. For example, CAS procedures running in the client can call one or more CAS actions to perform multiple or complex tasks. In this paper, the sample code is written with CAS procedures; but you can always translate a CAS procedure to its equivalent CAS action calls.

Figure 1 shows the visual interface (left) and the SAS Code node (right) of SAS Visual Data Mining and Machine Learning.

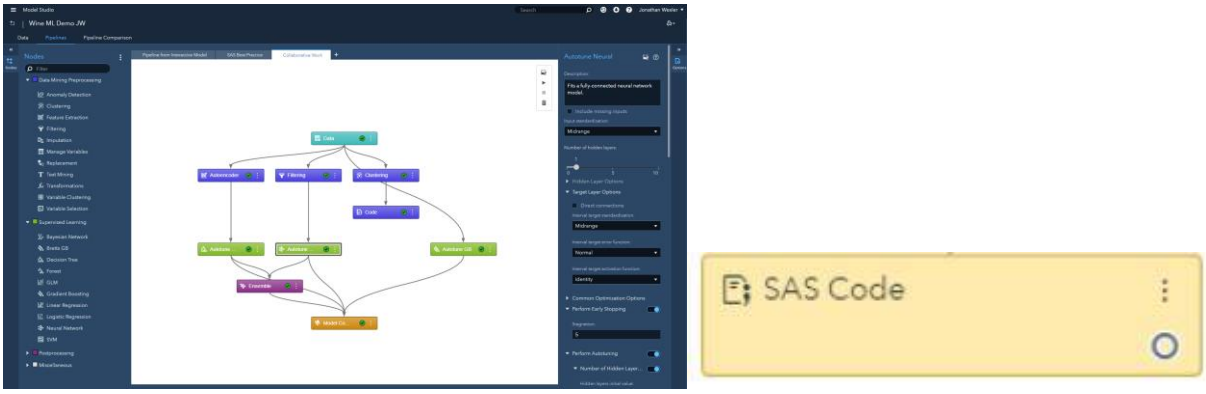

Figure 1. (left) The Visual Interface and (right) the SAS Code Node of SAS Visual Data Mining and Machine Learning

NEW FEATURES IN SAS VISUAL DATA MINING AND MACHINE LEARNING 8.5

SAS Visual Data Mining and Machine Learning 8.5 has the following new features [4]:

- A public API for automated modeling, including model development and deployment. You can also use this API to train and deploy your own custom predictive modeling applications.
- The SPARSEML procedure that was designed for analyzing data in sparse data formats.
- The Feature Machine node that enables you to automatically cleanse, transform, and select features for models.

- The Model Composer node that enables you to automatically select the best model across multiple techniques.
- The Kernel SHAP interpretability report that helps you to determine the relative importance of each variable to a given observation's prediction.
- Batch Reinforcement Learning that helps you to develop a smart system or agent that can optimize its outcome in a complex environment.

NEW FEATURES **FROM A PROGRAMMER'S** PERSPECTIVE

This paper focuses on the new features that are available to SAS programmers. As a SAS programmer, you can access these new features by directly calling CAS actions or procedures in the SAS Code node of SAS Visual Data Mining and Machine Learning. In this paper, the sample code is written with CAS procedures; but you can also use their equivalent CAS actions. From a programmer's perspective, the new features in SAS Visual Data Mining and Machine Learning 8.5 can be classified into four categories: a new action set, enhancements to existing action sets, new procedures, or enhancements to existing procedures.

NEW ACTION SET

SAS Visual Data Mining and Machine Learning 8.5 has the following new CAS action set:

- Kernel Principal Component Analysis Action Set

ENHANCEMENTS TO EXISTING ACTION SETS

SAS Visual Data Mining and Machine Learning 8.5 has enhancements to the following CAS action sets:

- Analytic Store Scoring Action Set
- Autotune Action Set
- BioMedImage Action Set
- Data Science Pilot Action Set
- Explain Model Action Set
- Factorization Machine Action Set
- Graph-Based Semisupervised Learning Action Set
- Image Action Set
- Network Action Set
- Support Vector Machine Action Set

NEW PROCEDURES

SAS Visual Data Mining and Machine Learning 8.5 has the following new CAS Procedures:

- The KPCA Procedure
- The SPARSEML Procedure

ENHANCEMENTS TO EXISTING PROCEDURES

SAS Visual Data Mining and Machine Learning 8.5 has enhancements to the following CAS Procedures:

- The FACTMAC Procedure

- The FOREST Procedure
- The GRADBOOST Procedure
- The SEMISUPLEARN Procedure
- The SVMACHINE Procedure

SAS Visual Data Mining and Machine Learning 8.5 also has enhancements to the AUTOTUNE Statement that is used in some procedures.

THREE NEW FEATURES FROM A **PROGRAMMER'S PERSPE**CTIVE

SAS Visual Data Mining and Machine Learning enables you to better process your data and make better decision. This section selects and presents three new features in SAS Visual Data Mining and Machine Learning 8.5 from a programmer's perspective.

BETTER GROUPING YOUR DATA WITH PROC KPCA

Principal component analysis (PCA) [6] uses orthogonal transformations to convert observations of possibly correlated variables into a set of values of linerally uncorrelated variables known as principal components. PCA is often used for data exploration, data visualization, and supervised machine learning. For supervised machine learning, it is desirable to use principal components to distinguish different groups of observations. When the principal components generated from linear PCA cannot distinguish different groups of observations, nonlinear techniques can be helpful.

Kernel principal component analysis (KPCA) is a nonlinear form of principal component analysis [6]. Just like PCA, KPCA aims to project the set of data onto a low-dimensional subspace that captures the highest possible amount of variance in the data. Whereas PCA projects the original data linearly onto a subset of the space, KPCA maps the original data to a high-dimensional space by a linear dimensionality reduction through the "kernel trick" in that space. Different mapping functions are implemented with different kernels. As a result, a nonlinear manifold that contains the original data can be generated. In many applications, we found that the principal components generated from KPCA can distinguish different groups of observations of the original data that is impossible for linear PCA.

To enable fast and memory efficient training for big data, kernel PCA uses techniques such as low-rank matrix approximation [7] and k-means clustering [8]. Also, the low-rank approximation method provides a fast and memory efficient way for scoring. The applications of kernel PCA include nonlinear dimensionality reduction, nonlinear data classification, kernel principal component regression, image denoising, novelty detection, and so on.

PROC KPCA is a new procedure in SAS Visual Data Mining and Machine Learning 8.5. It performs kernel principal component analysis, stores the results in output tables, and saves the projection of the training data into some kernel principal components. PROC KPCA offers the following features:

- supports training and scoring big data
- supports SMP (symmetric multiprocessing) and MPP (massively parallel processing)
- provides a variety of kernels such as linear, polynomial, and radial basis function (Gaussian) kernels.
- supports both fast training and fast scoring with the low-rank matrix approximation and k-means clustering techniques
- enables you to project the training or new data onto either linear or nonlinear principal components

This section shows how to use PROC KPCA to project a dataset that contains two groups (circles) of data points to a space so that the principal components generated from KPCA can distinguish the two groups (circles). The code below is a modified version from the PROC KPCA documentation [6]. It was modified to better fit the paper format and for better presentation. The results can be found in Figure 2. Figure 2 (left) shows the visualization of the original data points with one group marked as red circles and another group marked as blue circles. Figure 2 (right) shows that the first two kernel principal components generated from KPCA can linearly distinguish the two groups.

```
proc iml;
start Linspace(a, b, n);
  if n<2 then return( b ); incr = (b-a) / (n-1); return( do(a, b, incr) );
finish;
start makecircles(X, y, n_samples, noise, random_state, factor);
  pi = constant("pi");
  lins=Linspace(0, 2*pi, floor(n_samples/2)+1);
  n_2=floor(n_samples/2); lins=remove(lins,n_2+1);
  outer_circ_x=cos(lins); outer_circ_y=sin(lins);
  inner_circ_x = outer_circ_x#factor; inner_circ_y = outer_circ_y#factor;
  X=(outer_circ_x||inner_circ_x)`||(outer_circ_y||inner_circ_y)`;
  y=(j(1,n_2,0)||j(1,n_2,1))`; Xy=X||y;
  call randseed(random_state);
  obsIdx = sample(1:nrow(Xy), nrow(Xy), "NoReplace");
      X=Xy[obsIdx,1:ncol(Xy)-1]; y=Xy[obsIdx,ncol(Xy)]; Xn=j(nrow(X),2);
      if noise^=. Then call randgen(Xn, "Normal", 0, noise); X=X+Xn;
finish;
n_samples=600; noise=0.08; random_state=0; factor=0.3;
run makecircles(X, y, n_samples, noise, random_state, factor);
circles=X||y;
create circles from circles[colname={"x" "y" "group"}];
append from circles; close circles; quit;
proc sgplot data=circles; styleattrs datasymbols=(Circle X);
  scatter x=x y=y / group=group markerattrs=(size=7px); run;
cas; caslib _all_ assign;
data casuser.circles (replace=yes); set circles; run;
proc KPCA data=casuser.circles method=exact; input x y;
  output out=casuser.scored copyvar=group npc=2; run;
proc sgplot data=casuser.scored; styleattrs datasymbols=(Circle X);
  scatter x=_PCS_1 y=_PCS_2/group=group markerattrs=(size=8px); run; quit;
```

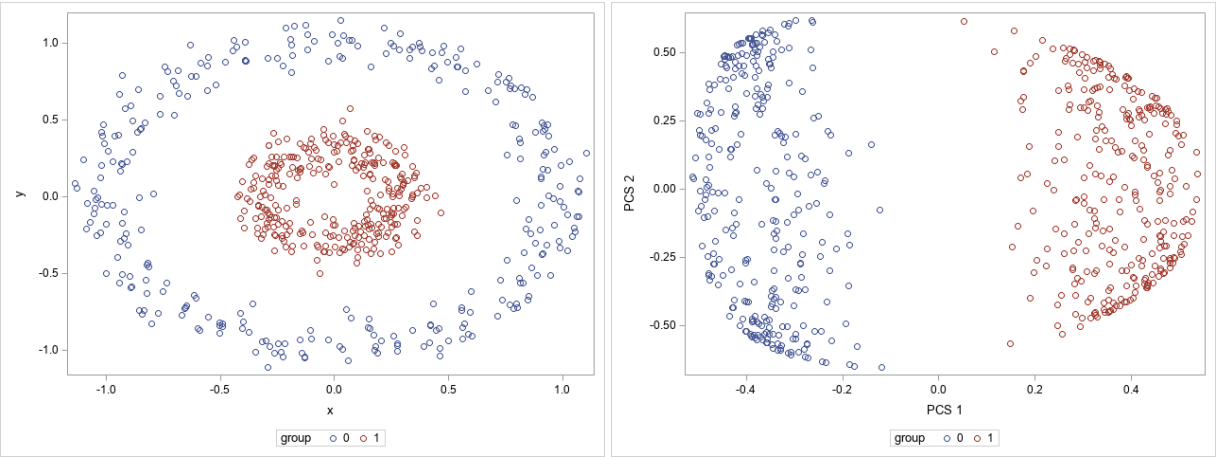

Figure 2. (left) The Original Data Points (right) The First Two Kernel Principal Components Generated from KPCA

BETTER LABELING YOUR DATA WITH PROC SEMISUPLEARN

Supervised machine learning or predictive modeling requires the target (dependent) variable Y to be labeled in training data so that a model can be built to predict the label of new or unseen data. However, labeling training data usually requires a physical experiment or a statistical trial, and human labor is often required. Therefore, it can be expensive to have the entire training data set labeled. Additionally, some already labeled observations could become missing due to various reasons such as human errors or packet loss from network data transmissions. It is very common that the target variable Y is not fully labeled in the training data set.

Semi-supervised learning [9, 10, 11] can play an important role in such situations. Semi-supervised learning (SSL) can use a small amount of labeled data together with a large amount of unlabeled data to help build better predictive models. Because it requires less human effort and produces better results, semi-supervised learning has been very popular in both academia and industry. Traditionally, semi-supervised learning algorithms have the following categories [10]: co-training, self-training, generative models, semi-supervised SVMs (S3VMs) and graph-based methods.

Co-training algorithms assume that features in the training data can be divided into two categories that are conditionally independent to each other. Another assumption is that each sub-feature set is sufficient to train a good model. The co-training algorithms first use the labeled data to train two separate models on the two sub-feature categories independently. Each model then independently gives predictions by assigning labels to the unlabeled data. After that, the newly labeled data are used by each model to teach the other model with high confidence. The usage of co-training has been limited because in many applications a sub-feature set is not sufficient to train a good model.

Unlike co-training, self-training does not depend on another model. The small amount of labeled data is first used to train a model. After the training is complete, the model then gives predictions to the unlabeled data. The newly labeled data with high confidence is added back to the training set. The model is then re-trained, and this process iterates until some pre-defined conditions are met. As its name indicates, the self-training algorithm teaches itself with its own predictions from its previous iterations. Self-training has been a popular choice in applications such as natural language processing. One of the problems of self-training is the error propagation issue, which means a classification error in a previous iteration can reinforce itself and hurt the final result.

Generative models assume the underlying machine learning model is a certain mixture of components, each of which can be identified. Gaussian mixture models have been a popular choice and the expectation-maximization (EM) algorithm has been used to construct such mixture models. If the mixture model assumption holds true, unlabeled data can be used to improve prediction accuracy. Unfortunately, if this assumption does not hold true, unlabeled data is very likely to hurt predictions [10].

Semi-supervised support vector machines (S3VMs) are also called transductive SVMs (TSVMs) in the literature [10]. It is an extension of the support vector machines (SVM) that is designed to handle both labeled and unlabeled data. The basic idea of SVM is to find a hyperplane to separate different classes of data points with the maximum margin. Due to this reason, SVM is also called maximum-margin-classifier. However, SVM algorithms usually discard training data with unlabeled target values. S3VM uses the unlabeled data to improve its classification performance. The goal of S3VM is to assign labels to the unlabeled data to find the hyperplane to separate different classes with maximum margin. Unfortunately, it is a NP-hard problem [10] to find the exact S3VM hyperplane. Finding good approximation algorithms has been the focus in the literature. However, existing S3VM algorithms cannot handle large unlabeled data. S3VM can be viewed as SVM with an additional regularization term on the unlabeled data. This is a non-convex problem, which

makes the optimization process very difficult. S3VM is also non-deterministic which means even for the same training data it can produce quite different results from one run to another run. In big data applications, this can become even worse because the data are distributed in different nodes and the distribution can be random. Therefore, S3VM is not the best choice for big data problems.

The basic idea of graph-based methods is to construct a graph connecting similar observations; labels propagate through the graph edges from labeled to unlabeled nodes. Graph based algorithms are becoming popular due to their capability of handling big data. PROC SEMISUPLEARN implements a graph-based semi-supervised learning algorithm. It was first released in 2018. In the 2019 releases, the KNN kernel and the AUTOTUNE statement were added. PROC SEMISUPLEARN offers the following features and advantages:

- handles big data
- supports SMP (symmetric multiprocessing) and MPP (massively parallel processing)
- supports both RBF and KNN kernels
- uses AUTOTUNE statement to find best hyperparameters for your data automatically

This section shows how to apply PROC SEMISUPLEARN to "impute" unlabeled target values. The dataset is the German credit benchmark dataset, which is available at SAMPSIO.DMAGECR. The binary target variable is named GOOD_BAD. The code below is a modified version from the PROC SEMISUPLEARN documentation [9]. It was modified to better fit the paper format and for better presentation. The results can be found in Figure 3. Figure 3 (left) shows the first ten observations of unlabeled data. Figure 3 (right) shows the first ten observations of data after PROC SEMISUPLEARN assigns labels to them.

```
cas; caslib _all_ assign;
data casuser.dmagecrunlabel(drop=good_bad);
   set sampsio.dmagecr(obs=800); id=_N_; run;
data casuser.dmagecrlabel; set sampsio.dmagecr(obs=50); run;
proc semisuplearn data=casuser.dmagecrunlabel label=casuser.dmagecrlabel;
   input duration amount installp resident existcr;
   kernel KNN/k=5;
   output out = casuser.out copyvars=(amount installp id);
   target good_bad;
   autotune TUNEPARMS=(GAMMA(lb=1 ub=10) k(lb=2 ub=20));
run;
proc print data=casuser.dmagecrunlabel(obs=10 keep=duration amount);run;
proc print data=casuser.out(obs=10 drop=_warn_ id);run;
```

Obs	duration	amount
1	6	1169
2	48	5951
3	12	2096
4	42	7882
5	24	4870
6	36	9055
7	24	2835
8	36	6948
9	12	3059
10	30	5234

Obs	duration	amount	I_good_bad
1	6	1169	good
2	48	5951	good
3	12	2096	good
4	42	7882	good
5	24	4870	bad
6	36	9055	good
7	24	2835	good
8	36	6948	good
9	12	3059	good
10	30	5234	bad

Figure 3. (left) The First Ten Observations of Unlabeled Data (right) The First Ten Observations of Data After PROC SEMISUPLEARN Assigns Labels to Them

BETTER DECISIONING FOR SPARSE DATA WITH PROC SPARSEML

A sparse data set is a data set with many zeros and very few non-zero elements. Figure 4 shows an example of sparse dataset. As you can see in Figure 4, there are only four non-zero elements in this dataset and all the other elements are zeros.

_1	_2	_3	_4	_5	_6	_7	_8	_9
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0.3195505541	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0.5543968899	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.3692914593
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0.3195505541	0	0	0	0	0	0	0	0

Figure 4. An Example of a Sparse Dataset

Sparse data can be found in many applications such as text analytics, genomics, point-of-sale, IOT, banking, retail, and government. It is a waste of storage and time to store and process the zeros in a sparse data set if the zeros do not carry any information. When the sparse data set is large, computers can run out of memory trying to store and retrieve the zeros, and supervised machine learning models can run for a long time processing those zeros and building models. In those applications, PROC SPARSEML [13] can be useful.

PROC SPARSEML is a new procedure in SAS Visual Data Mining and Machine Learning 8.5. It is a supervised machine learning model that was designed to handle sparse data sets. It supports the LIBLINEAR sparse data format [14] and the parallel coordinate descent optimization algorithm [12]. The LIBLINEAR sparse data format is a COO format [14] that stores a list of (row, column, value) tuples without storing the zeros. This is a sparse data format that is good [14] for sparse matrix constructions. The parallel dual coordinate descent algorithm [12] is one of the most effective approaches for large-scale sparse data classification in multi-core environments. By combining the power of both the sparse data format and the parallel dual coordinate descent algorithm, PROC SPARSEML offers the following features and advantages:

- supports training and scoring big data in sparse format
- supports high dimensions of variables (wide data table)
- supports SMP (symmetric multiprocessing) and MPP (massively parallel processing)
- requires little storage space when the dataset is sparse and in sparse format
- fast training and scoring

This section shows how to build a predictive model with PROC SPARSEML to handle sparse data. The data set used here is the SASHELP.JUNKMAIL data set. The SASHELP.JUNKMAIL data set comes from a study that classifies whether an email is junk email or not. The data was collected by Hewlett-Packard Labs and was donated by George Forman. The target variable, CLASS, is a binary indicator of whether an email is considered junk (coded as 1) or not (coded as 0). The code below is a modified version from the PROC SPARSEML documentation [13]. It was modified to better fit the paper format and for better presentation. The results can be found in Figure 5. Figure 5 (left) shows the training data information that demonstrates how sparse the training dataset is. Figure 5 (right) shows fit statistics of the predictive model.

```
data JunkMail;
  length sparseString $600; length oneVariable  $60; set Sashelp.JunkMail;
  array vars{*} _NUMERIC_;
  if Class=0 then do; sparseString = put(-1, 2.);
  end; else do; sparseString = put(+1, 2.); end;
  do i=2 to 58;
    if vars{i} ne 0 then do;
      oneVariable = strip(put(i-1, 5.))||':'||strip(put(vars{i}, 10.3));
        sparseString = strip(sparseString)||' '||oneVariable;
      end;
  end;
  keep Test Class sparseString;
run;
data mailtrain; data test.mailtrain;
    id = _N_; set JunkMail; where Test=0; drop Test;
run;
cas; caslib _all_ assign;
data casuser.mailtrain (replace=yes); set mailtrain; run;
proc sparseml data=casuser.mailtrain C=0.1 MAXITERS=400;
  input sparseString; savestate rstore=casuser.mystate;
run;
data casuser.dmagecrunlabel(drop=good_bad);
   set sampsio.dmagecr(obs=800); id=_N_; run;
data casuser.dmagecrlabel; set sampsio.dmagecr(obs=50); run;
```

Data Information	
Number of Rows	3065
Number of Features	57
Number of Sparse Elements	39801

Fit Statistics	
Statistic	Training
Accuracy	0.8920
Error	0.1080
Sensitivity	0.7709
Specificity	0.9718

Figure 5. (left) Data Information (right) Fit Statistics of the Predictive Model

SOME OTHER SELECTED NEW FEATURES

This section selects and lists some other new features.

THE AUTOTUNE STATEMENT

- Now enables you to create a hyperparameter importance ODS table
- Now supports more options and warm start, saving the configuration history

THE FOREST PROCEDURE

- The NUMBIN= and MINLEAFSIZE= options can now be set by autotuning
- The default value of the NUMBIN= option is now 50 instead of 20
- The default value of the BINMETHOD= option is QUANTILE instead of BUCKET
- For isolation forests, the TARGET statement is no longer required, and fit statistics that include variable importance are no longer computed

THE GRADBOOST PROCEDURE

- The MINLEAFSIZE= option can now be set by autotuning
- The NUMBIN= option default is now 50 instead of 20
- The MAXDEPTH= option default is now 4 instead of 5
- The RIDGE= option default is now 1 instead of 0
- The BINMETHOD= option default is now QUANTILE instead of BUCKET

ANALYTIC STORE SCORING ACTION SET

- Now supports the varying-length binary data type in SAS Embedded Process code and for SAS Event Stream Processing
- Now supports multiple output tables and multiple output rows per input row for SAS Event Stream Processing
- Now supports the hidden Markov model action set, the kernel principal component analysis action set, the mixed modeling action set, the reinforcement learning action set, and the gamselect action in the generalized additive models action set

AUTOTUNE ACTION SET

- The following actions are now supported for autotune: tuneGlm, tuneLabelSpread and tuneLogistic
- The relative importance of each hyperparameter based on all configurations evaluated is now reported in a new results table
- Now outputs the evaluation history table
- The tuneDecisionTree, tuneForest, and tuneGradientBoostTree actions can now optionally tune the leafSize parameter
- The tuneForest action now supports tuning of the nBins parameters

BIOMEDIMAGE ACTION SET

- The processBioMedImages action now:
    - Enables users to filter ROIs by their display color in the DICOMRT_SPECIFIC substep of the ROI2MASK step. Accordingly, if the DICOM-RT data contain contours for multiple organs, each with a different value of the ROI Display Color DICOM attribute, the user can select a single organ for the action to generate segmentation images
    - Incorporates a new step called BINARY_OPERATION, which supports operations on pairs of biomedical images. The step currently includes one substep called MASK_SPECIFIC, which can be used to mask one image with another
    - Includes a new step called HISTOGRAM_EQUALIZATION, which enhances the contrast of biomedical images. This step consists of the BASIC substep, which

considers the histogram of the entire image, and the ADAPTIVE substep, which uses the histogram of a chosen image window

DATA SCIENCE PILOT ACTION SET

- The new detectInteractions action searches for interactions among pairs of predictor variables in a regression or classification data table. The action generates an output CAS table that contains the candidate interactions ranked according to their interaction strength
- The new generateShadowFeatures action generates new random features by a scalable, random permutation of the input features. These randomly permuted features, usually referred to as shadow features, can be used for all relevant feature selection
- The featureMachine action has several new parameters and updates. The rankPolicy parameter specifies the ranking policy to apply to the generated features, the interaction parameter controls the generation of interaction features, and features now come with autogenerated labels that describe the transformations
- The dsAutoMl action has several new parameters. These include an expanded saveState parameter to support the generation of analytic score objects for both the feature generation and the top-k models, generalized linear model (GLM) and logistic model types, and an hpOptimizer parameter to specify the hyperparameter optimizer to use for model hyperparameter tuning

EXPLAIN MODEL ACTION SET

- A new shapleyExplainer action now gives accurate Shapley value estimates by using the HyperSHAP method invented by SAS
- The partialDependence action now supports a second analysis variable and more analysis variable parameters, including using missing values as a valid level and accepting user-defined levels as input
- The linearExplainer action now supports more parameters for training regression models and supports including missing values in the analysis
- All actions in the action set now support the use of DS2 code to accompany an analytic store model, as well as multiple analytic stores as part of a full model

IMAGE ACTION SET

- The condenseImages action generates encoded or decoded image tables, depending on the value of the new decode parameter. Most common photographic image formats, such as JPG and PNG, are supported for the encoding
- The RESIZE step in the processImages action now supports preserving the aspect ratio of images during resizing by using the new type parameter
- The MUTATIONS step in the processImages action has a new parameter called angle. You can use the angle parameter with the ROTATE_LEFT and ROTATE_RIGHT values of the type parameter to rotate images by arbitrary angles
- The MUTATIONS step in the processImages action now supports perspective transformations of images. You can perform a perspective transformation by specifying the WARP_PERSPECTIVE value of the type parameter and the values of the homography matrix in the new homographyValues parameter

CONCLUSION

SAS Visual Data Mining and Machine Learning enables you to better process your data and make better decisions. The new SAS Visual Data Mining and Machine Learning 8.5 offers many great new features and new capabilities. As a SAS programmer, you can have access to those new features by directly calling CAS procedures in the SAS Code node of Model Studio. We believe many of you can find good reasons to upgrade to SAS Visual Data Mining and Machine Learning 8.5 to bring more value to your organization and your customers.

REFERENCES

[1] SAS® Platform. Accessed Feb 8, 2020. https://www.sas.com/en_us/software/platform.html

[2] SAS® Cloud Analytic Services (CAS). Accessed Feb 8, 2020. https://go.documentation.sas.com/?docsetId=whatsdiff&docsetTarget=p1gfaswb875orbn1xn4ao8b8jbfq.htm&docsetVersion=3.2&locale=en

[3] SAS® Viya®. Accessed Feb 8, 2020. https://www.sas.com/en_us/software/viya.html

[4] SAS® Visual Data Mining and Machine Learning. Accessed Feb 8, 2020. https://support.sas.com/en/software/visual-data-mining-and-machine-learning-support.html

[5] CAS actions. Accessed Feb 8, 2020. https://go.documentation.sas.com/?docsetId=pgmdiff&docsetTarget=p06ibhzb2bklaon1a86ili3wpil9.htm&docsetVersion=3.4&locale=en#n038r32gkh6lw9n1nehq3dttefg3

[6] The KPCA Procedure. Accessed Feb 8, 2020. https://go.documentation.sas.com/?docsetId=casml&docsetVersion=8.5&docsetTarget=casml_kpca_overview.htm&locale=en

[7] C. Baker, The Numerical Treatment of Integral Equations, Oxford: Clarendon Press, 1977.

[8] D. Arthur, and S. Vassilvitskii, k-Means++: The Advantages of Careful Seeding, ACM-SIAM Symposium on Discrete Algorithms, 1027−1035, 2007.

[9] The SEMISUPLEARNING Procedure. Accessed Feb 8, 2020. https://go.documentation.sas.com/?docsetId=casml&docsetVersion=8.5&docsetTarget=casml_semisuplearn_overview.htm&locale=en

[10] X. Zhu, Semi-Supervised Learning Literature Survey, Computer Sciences TR 1530, University of Wisconsin Madison, 2008.

[11] X. Chen and T. Wang, Combining Active Learning and Semi-Supervised Learning by Using Selective Label Spreading, IEEE ICDM workshops, 2017.

[12] W. Chiang et al., Parallel Dual Coordinate Descent Method for Large-scale Linear Classification in Multi-core Environments, KDD 2016.

[13] The SPARSEML Procedure. Accessed Feb 8, 2020. https://go.documentation.sas.com/?docsetId=casml&docsetVersion=8.5&docsetTarget=casml_sparseml_overview.htm&locale=en

[14] COO data format. Accessed Feb 8, 2020. https://en.wikipedia.org/wiki/Sparse_matrix#Coordinate_list_(COO)

ACKNOWLEDGMENTS

The author would like to take this opportunity to thank the editor for the editing work and thank the SAS Global Forum 2020 organizers for good organization.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Tao Wang
SAS Institute Inc.
100 SAS Campus Drive
Cary, NC 27513
Phone: 919-531-6576
Email: t.wang@sas.com