Paper SAS3496-2019

Session 3496 - The Aftermath What Happens After You Deploy Your Models and Decisions

David R. Duling, SAS Institute Inc.

ABSTRACT

We're making it easy to deploy your models and decisions to numerous run-time environments. However, the model life cycle doesn't end once the model is created. Rather, it is just the beginning of the important phases of model monitoring and analysis. This need extends to SAS models and open-source and Predictive Model Markup Language (PMML) models. In this demonstration, you learn techniques for analyzing model performance, integration with business metrics, and root cause analysis.

INTRODUCTION

You can think of the lifetime of a model as having three major phases. In data preparation, the business operational data is transformed and loaded for targeted business analytics. In the discovery phase, advanced reporting, statistics, and machine learning models are developed for gaining insight into patterns and trends that influence the business. In the deployment phase, those models are used to make predictions in critical business processes that drive operational decisions. This is where the models provide their greatest benefit to the organization; however, it is also where most businesses encounter challenges in completing the analytics lifecycle.

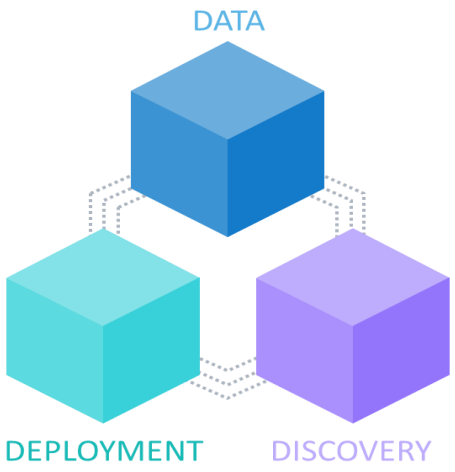


Figure 1. The new analytic life cycle.

A data scientist can spend weeks constructing a good model for prediction or classification using statistical, machine learning, or deep learning techniques. These models can be used to provide insight and inference into existing processes, or to predict outcomes based on new data values. These predictions are used to improve the effectiveness of automated decision-making systems such as the next best offer, credit scoring, loan originations, fraud detection, robotic process automation, and hundreds of other applications. Modern businesses require the use of predictive models to remain competitive.

This paper will focus on model deployment and describe how models are used, how they are monitored, and how they can be evaluated and replaced.

MODEL SCORING

The building of predictive models is often termed *model training* and typically takes place offline in a development environment with saved historical data. The result of training a model is a fixed function that can be used for making predictions with new data values. They key component of most models is *score code* that can be evaluated with to make those predictions. Model scoring is the key function in model deployment. Score code is generally found in the primary language of the system that generated the score code, including the following forms:

SAS Procedures

This form was introduced with SAS® Stat® more than 30 years ago. While this is a very easy form to run in a SAS program, the drawback is that the code cannot easily be executed in non-SAS environments.

The original form of the output statement would replace missing values in the dependent column with new predicted values based on the model.

```
proc reg data= train outest=model;
      model bad= debtinc ninq clage clno;
      output out=scores;
run ; quit ;
```

Later, the Score procedure was introduced to score models that have a generalized linear model form. The model could be saved for use in scoring later without needing to re-create the model.

```
proc score data=train score=model out=scores type=parms;
   var debtinc ninq clage clno;
run;
```

Finally, the Score statement was introduced to some procedures such as proc Logistic to accept separate input and output data sets that were not included in the model training.

```
proc logistic data= train;
      class bad;
      model bad= debtinc ninq clage clno;
      score data= production out=scores;
run;
```

SAS Data Step

SAS® Enterprise Miner® introduced the concept of data step score code. SAS procedures were enhanced to include the code statement to produce score code. The following code shows creation of the SAS data step score code.

```
proc logistic data= train;
      class bad;
      model bad= debtinc ninq clage clno;
      code file= 'c:\temp\scorecode.sas';
```

```
run;
```

This form has some key advantages. The code is transparent and can be audited. Users can easily see the functional form and can reproduce the results in many different tests. The code can be modified and extended. Users can add additional logic for inline preprocessing data preparation such as missing value handling or binning, and for post processing such as model comparison or creation of decision variables. A portion of the data step code is shown below.

```
… snip …
*** Effect: DEBTINC;
_LP0 = _LP0 + (-0.07795619150744) * DEBTINC;
*** Effect: NINQ;
_LP0 = _LP0 + (-0.11881501397556) * NINQ;
*** Effect: CLAGE;
_LP0 = _LP0 + (0.00596147711961) * CLAGE;
*** Effect: CLNO;
_LP0 = _LP0 + (-0.00263720328025) * CLNO;

*** Predicted values;
drop _MAXP _IY _P0 _P1;
_TEMP = 4.34723047453817 + _LP0;
if (_TEMP < 0) then do;
  _TEMP = exp(_TEMP);
  _P0 = _TEMP / (1 + _TEMP);
end;
… snip …
```

SAS DS2

The DS2 language is a more structured form of SAS Data step with definitions for packages and methods that enables building more modular code. The primary scoring advantage is that DS2 language is portable and can be run outside of a SAS server in a variety of scoring environments. Generally, models are converted from Data step to DS2 for scoring outside the SAS server. SAS® Model Manager® will automatically convert Data step to DS2 as needed. SAS® VDMML Model Studio® is an exception that will create DS2 code for models that contain one or more Astore files.

SAS Astore file

Unfortunately, the size of data step files would be come too large with complex models that could produce well over ten thousand lines of data step code. The next solution was to package all the model structure information into a single binary file that could be used to later score new data. These files are named 'Astore' – Analytical Store format. You can execute an Astore model in a SAS program, a DS2 program, or directly in a CAS action. Astore files are portable and can be used in all scoring services produced by SAS. The following code shows both creation of the Astore file and usage of the Astore file.

```
proc gradboost data= casuser.train;
      target bad / level = nominal;
```

```
      input debtinc ninq clage clno / level=interval;
      savestate rstore=casuser.gradboost_model;
run;

proc astore;
    score data=casuser.train
       rstore=casuser.gradboost_model
       out=casuser.scores ;
quit;
```

PMML (Predictive Model Markup Language)

This format was developed by collaborative members of the Data Mining Group standards organization to facilitate the exchange of models between development software and run time scoring systems. A PMML file is not run time code, but rather a description of the model. You still need software to read, interpret, and then execute the model described by the PMML file. SAS® Model Manager® can read PMML files for most models that adhere to the PMML 4.2 standard. The file is converted to SAS Data step code that can be executed as part of a SAS program. In addition, SAS® Enterprise Miner® can create PMML files. The following portion of a PMML file shows lhe Logistic regression model coefficients for this sample. Refer to SAS documentation and DMG information for the range of models that can be expressed as PMML files.

```
  … snip …
  <RegressionTable intercept="4.3472304745" targetCategory="0">
     <NumericPredictor name="DEBTINC" coefficient="-0.077956192"/>
     <NumericPredictor name="NINQ" coefficient="-0.118815014"/>
     <NumericPredictor name="CLAGE" coefficient="0.0059614771"/>
     <NumericPredictor name="CLNO" coefficient="-0.002637203"/>
   </RegressionTable>
  … snip …
```

Python

Python has become the most popular open source language for building machine learning models. SAS enables use of Python in some SAS processes. Most Python models are saved as a Python Pickle file which is a general binary format for saving the state of a Python package. A Python program will read the Pickle file when scoring new data. The process is very similar to using an Astore file in a SAS program. The following program can run in SAS servers that support Python. Python execution is supported by the DS2 PYMAS package that is available with SAS® Model Manager®, SAS® Intelligent Decisioning®, and SAS® Event Stream Processing® (ESP). Python can also be executed directly by ESP. A Python program must be written in the following form with a function definition, the list of input variables, a comment that lists the output variables, and a return statement.

```
import pandas
import pickle
def scoreMNLogitModel (CLAGE, CLNO, DEBTINC, NINQ):
    "Output: I_BAD"
    # Open a read-only binary file for reading the pickle object
    _pFile = open('C:\\MyJob\\SGF\\2019\\MNLogit_Model1.pickle', 'rb')
```

```
    # Unpickle the file to recover the model specification and
estimates
    _thisModelFit = pickle.load(_pFile)
    # Close the binary file
    _pFile.close()
    # Construct the input array for scoring
    # the first term is for the Intercept
    input_array = pandas.DataFrame([[1.0, DEBTINC, NINQ, CLAGE, CLNO]],
         columns = ['const', 'DEBTINC', 'NINQ', 'CLAGE', 'CLNO'])
    # Calculate the predicted probabilities return data frame predProb
    _predProb = _thisModelFit.predict(input_array)
    # Determine the predicted target category
    I_BAD = pandas.to_numeric(_predProb.idxmax(axis = 1))
    return(I_BAD)
```

SCORING SERVICES

SAS provides several scoring services for interactive and operational systems. Table 1 shows a matrix of score code and scoring servers.

SAS

The traditional SAS® server can execute all types of score code. SAS can run interactively through Display Manager or SAS® Studio®, it can run batch jobs processing large tables of data, or it can run as a Stored Process for on-demand applications that don't require sub-second processing.

CAS

The SAS® Viya® CAS server is a big-data, in-memory server for complex data processing and analytical workloads. CAS can run all score code forms except for SAS procedures. CAS provides action sets for data step, DS2, and in-database processing.

INDB

SAS In Database processing is a feature of the SAS Access engines. For SAS® Viya®, In-Database processing is supported for Teradata and Hadoop, both Hive and Spark. This processing is normally used to process large tables of data that are natively created in the database. It is not generally appropriate for scoring transactions being processes by the database.

MAS

The Micro Analytic Score (MAS) service is a standard SAS® Viya® microservice that contains the ability to score models. MAS provides a REST interface that can be used to provide on-demand scoring functions that run in the sub-second time scale. The service is typically used in automation applications such as next best offer or fraud detection. MAS is distributed with the SAS® Model Manager® and SAS® Intelligent Decisioning® packages.

ESP

The SAS® Event Stream Processing® (ESP) server is a dedicated operational real-time system for processing high speed data streams. The server is typically used in IOT

systems that require embedded analytical processing. ESP can be programmed to detect patterns and execute actions, including scoring models.

Code Types	SAS Viya Scoring Services				
	SAS	CAS	INDB	MAS	ESP
Procedure	Y	N	N	N	N
Data Step	Y	Y	N	N	N
DS2	Y	Y	Y	Y	Y
PMML	Y	Y	Y	Y	Y
Astore	Y	Y	Y	Y	Y
Python	Y	Y	N	Y	Y

Table 1. Score code types and SAS® Viya® scoring services.

SENARIO

The remainder of this paper will refer to models created from data than is more complex than the simple examples shown above. The scenario is a fleet of trucks that are outfitted with sensors for collecting real time measurements of various systems on the truck.

Our task is to predict a future need for maintenance for each truck based on history of sensor readings and maintenance events. The target variable is maintenance_flag which has two values: 0 and 1. There are 12 fleets. Each fleet has from 1 to 8 trucks. The entire training data sample is 8307 rows of data. Each row of data contains a set of truck sensor measurements and a target variable that indicates if that truck later needed unscheduled maintenance. The data in Table 1 shows the distribution of the target variable for each fleet.

We can see that one fleet, 01013F1, did not have any truck maintenance events. However, we can build a model for the pool of all truck fleets and apply it to that fleet. This pattern demonstrates the power of using pooled data to improve models for all customers.

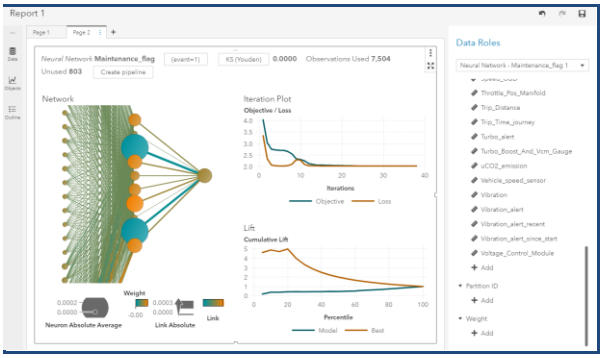
Table of Maintenance_flag by fleetid

	fleetid												
Maintenance_flag	Fleet_00113F1	Fleet_00213F1	Fleet_00313F1	Fleet_00413F1	Fleet_00513F1	Fleet_00613F1	Fleet_00713F1	Fleet_00813F1	Fleet_00913F1	Fleet_01013F1	Fleet_01113F1	Fleet_01213F1	Total
0	479	305	300	1028	470	655	1044	378	674	23	442	624	6422
1	79	102	208	206	149	17	232	94	278	0	315	205	1885
Total	558	407	508	1234	619	672	1276	472	952	23	757	829	8307

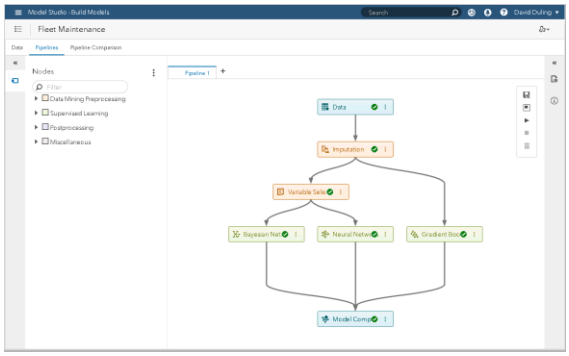
Table 2. Distribution of target values for each truck fleet in the data sample.

MODEL CREATION

While this paper does not primarily focus on model development, we do need a set of models to complete the scenario. We can start by exploring the data and building an initial model in SAS® Visual Statistics®. In this case we created a Neural Network model due to the flexible form. We then transferred the model to Model Studio where we have a more robust set of functions for building more models. We created a model pipeline to train and compare multiple candidate models. In addition to the Neural Network, we also created a Bayesian Network model and a Gradient Boosting Model. These models can be seen in displays 1 and 2. In addition, one of our developers is a Python enthusiast and he created a multinomial logistic model.

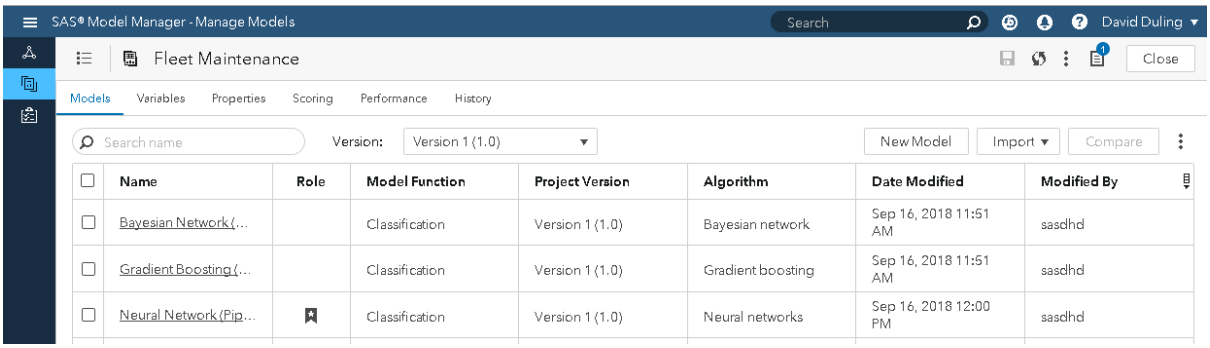


Display 1. Neural Network Model



Display 2. Model Studio pipeline

As the candidate models are trained, we can compare them in SAS® Model Manager®. We create one project for deploying this model. In that project, we can see all the candidate models including their attributes. We now have our set of models and can begin the process of model deployment.



Display 3. Model project showing multiple candidate models.

MODEL SELECTION

The first step in model deployment is to select and validate which model will be used in production. When the business process is being planned, there may be several candidate models that can be used. You will want the model the produces the best results, but you need to balance accuracy with other factors. Once you know your criteria, you can begin the process of model selection.

- Does the model match the business needs? If the model predicts individual sensor readings but you need the prediction of unscheduled maintenance, then you need to reconsider the model. It might a good enough proxy for the model you need, or maybe not.

- Does the model represent the correct time-period? If the model was trained on data from July but will be used in December, when the weather is very different, then you need to reconsider the model.

- How robust is the model? The model should be tested on data that represents the run time systems and model performance compared to both other models and the

original model training statistics. If the model performance significantly deviates, then you should reconsider the model.

- Can the model be explained? In many cases, the business requires the model to be explainable to external regulators, internal auditors, or customers. A great complex machine learning model might be useful for back-testing and measuring performance but might not be appropriate for production decision making.

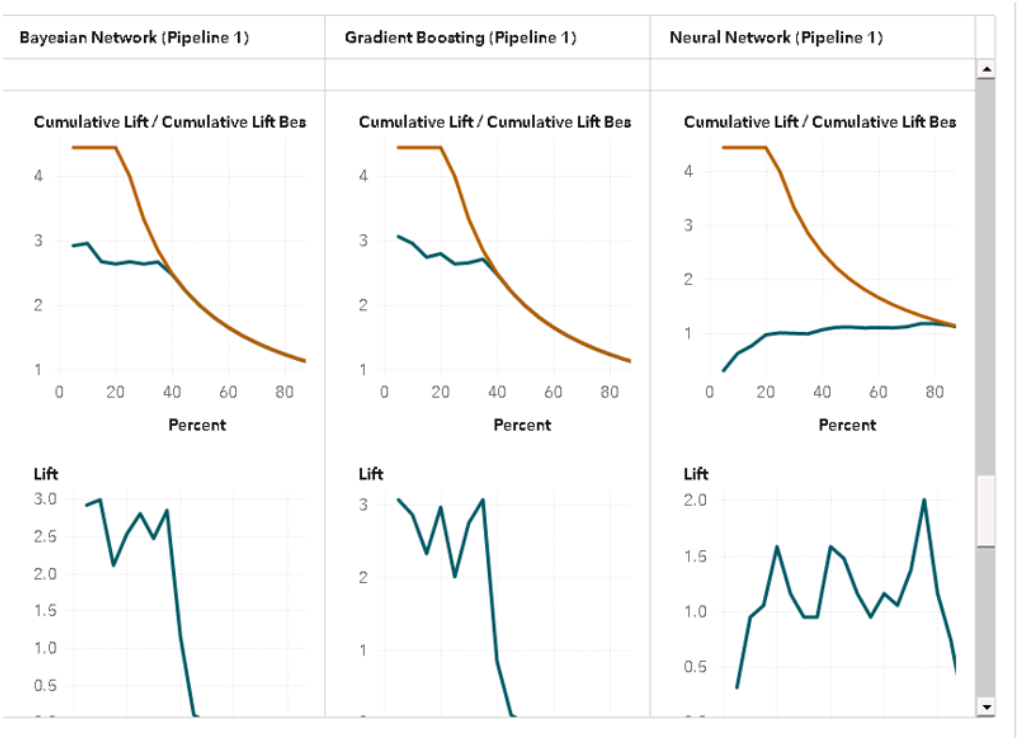
- Does the model score code execute in the needed run time environments? Not all environments can run all score code types. The cost of recoding a model for a new language is very high. All the test processes must be repeated. Models with binary formats such as Astore files or Pickle files may not be portable to new languages.

- Is a new model a notable improvement over an older model? Each new model is a candidate model. It must be compared to the champion model on newer test data. If there is no improvement, then you need to reconsider the new model.

- Does the model require data that is available in the run time system? Some model features might not be available in all run time environments. In that case you need to know the cost to place that data in the run time system.

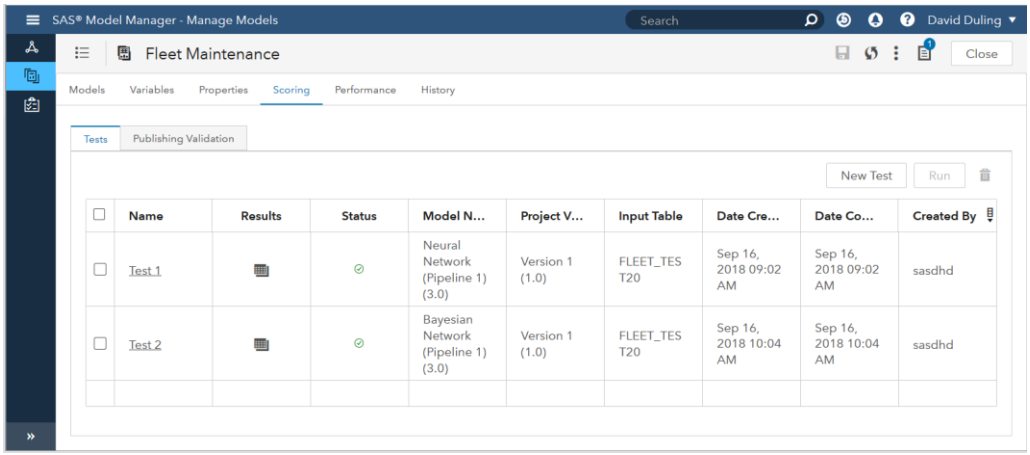
- How fast does the model execute? Some models might require more time to execute than others. This could be due to the functional form, or the quantity or availability of the needed data. If some of the data requires a fetch from an external database, that will cause a significant reduction in run time speed.

We can compare the attributes, variables, and statistics associated with each model. In this case, there is a not a significant difference in the data variables needed for each model; however, there is a difference in the observed expected performance of the models. Display 4 shows the results. The neural network model shows signs of overfitting. The training data Cumulative Lift, shown in the brown plot line, is very good. The validation data plot, shown in the blue line, is much worse. It has an unusual convex shape indicating a potential missing data problem. This model is not robust and should not be used in production. The remaining models, Bayesian Network and Gradient Boosting, show similar performance in their training and validation statistics.



Display 4. Comparison of model statistics inside SAS® Model Manager®.

Another tool is model testing. We can score both models on a common test set and look at the results is detail for differences. To create the test set, we selected the first ten rows of data with target value 1 and next ten rows of data with target value 0. We need to make sure the models can predict these values correctly. In display 5, we have configured and executed the test for each model.



Display 5. Model Testing.

The results are clear as shown in Display 6. The Gradient Boosting model does not correctly predict the target value 1 in this sample. The Bayesian Network correctly predicts both target values and is the better choice for our deployment.

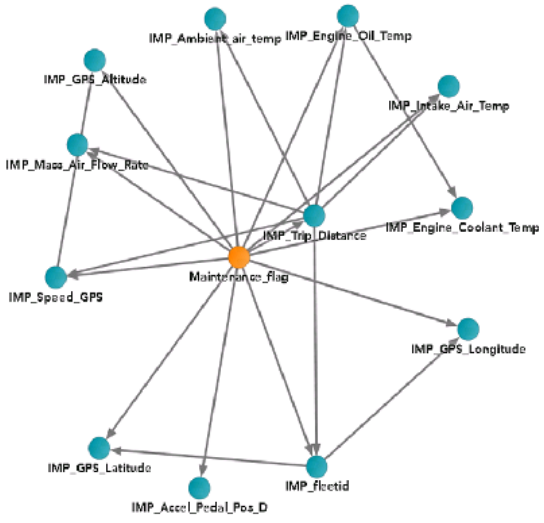
grad boost	bayes net
EM_CLASSIFICATION	EMI_CLASSIFICATION
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

Display 6. The results of model testing showing the better Bayesian Network.

Another benefit of the Bayesian Network in this case is that it is more explainable. This could be important in determining why the truck was referred for maintenance or looking for trends across the fleet. The information might be used in warranty claims. Display 7 shows the list of important variables for this model.

Order of Input Variables

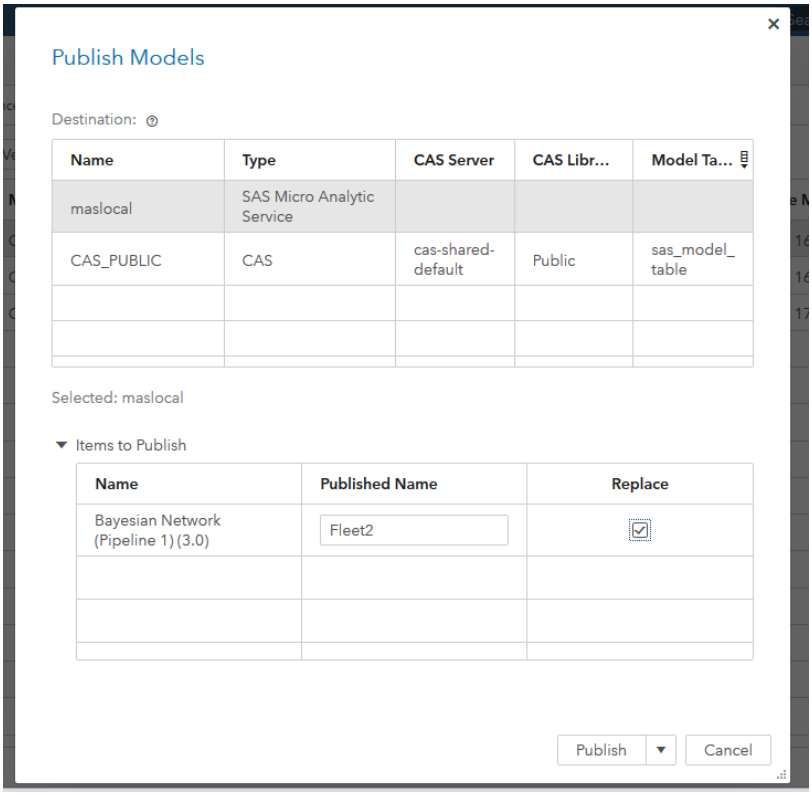
Variable Name	Order	Score
IMP_Voltage_Control_Module	1	-573.9700
IMP_Trip_Distance	2	-1,729.9008
IMP_Speed_GPS	3	-2,407.6189
IMP_GPS_Altitude	4	-2,480.7904
IMP_Ambient_air_temp	5	-2,492.9494
IMP_Intake_Air_Temp	6	-2,521.0292
IMP_Mass_Air_Flow_Rate	7	-2,530.6988
IMP_fleetid	8	-2,546.6659
IMP_Engine_Oil_Temp	9	-2,546.9421
IMP_GPS_Longitude	10	-2,548.6041
IMP_GPS_Latitude	11	-2,552.3625
IMP_Engine_Coolant_Temp	12	-2,590.1810
IMP_Accel_Pedal_Pos_D	13	-2,696.9639



Display 7. The important variables list and diagram for the Bayesian Network model.

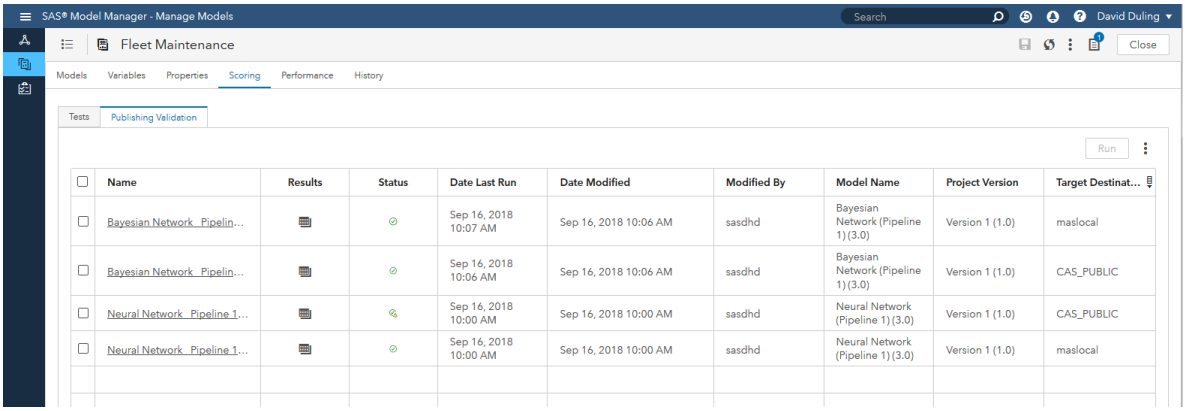
MODEL DEPLOYMENT

Now that we have selected our model, we can deploy the score code it to our production server. In our configuration we have two choices. We can deploy the model to the CAS server for BI applications or batch processing. We can deploy the model the MAS service for online web service processing. In this scenario, we select the MAS service. As the trucks produce new batches of sensor data, new scores will be generated on demand and transmitted to the maintenance facility. When a high score is detected, the facility can schedule trucks for service visits.



Display 8. Model publishing to the MAS service.

The publish function will format the score code and transmit it to the MAS service where it will be compiled into memory and exposed in new web service REST endpoint. There is one more test we should run. We need to test the web service to make sure it produces the correct result. We can use the Publish Validation function in Model Manager to validate the endpoint. In this case, we are using the CAS table to supply the data, but the scoring function calls the MAS web service with the exact same interface that will be used by the business application. Use the same data used for model selection to ensure consistency. This test insures that the model is ready for use. Display 8 shows a set of tests that have already been run in both CAS and MAS to validate the production model. The results of the test will show that the correct scores have been returned by the service.



Display 8. Testing the models before enabling the production process.

MODEL DEPLOYMENT

The main task now is to embed the model in the business application. The first thing we need to do is promote the model to the production fleet management server. SAS provides the Transfer Service to save and move content between servers. You may move all development artifacts including models and rule sets if you expect to make last minute changes in the production environment. If you expect to make no changes, then you can move only the MAS modules to the production server. This approach is less flexible but is more efficient and eliminates some chances of errors. Figure 2 illustrates the promotion of content between environments.

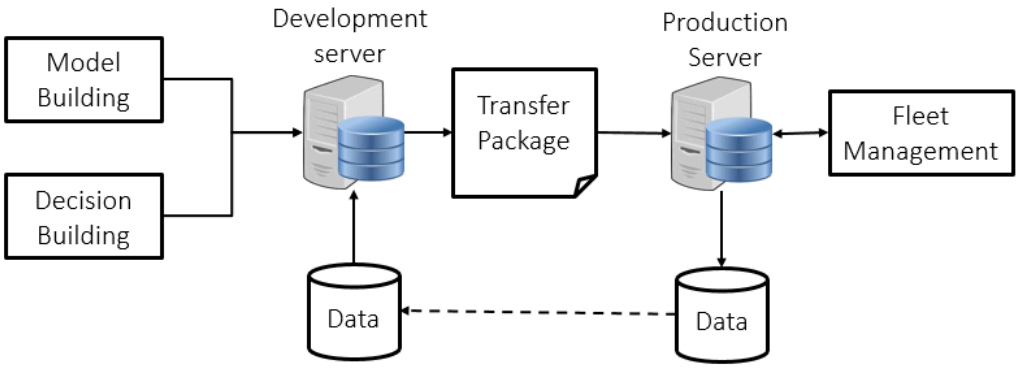


Figure 2. Promotion of models between environments.

In our scenario, we only need to move the MAS modules to the production server. The truck service management application will be programmed to call the MAS service to score new data observations and make optimal decisions. This could happen when the truck is stored overnight as daily data is downloaded and run through the scoring service. It could happen as the truck is moving and sensor measurement data is transmitted by LTE connection to the fleet data center. The fleet management application will receive the new,

run the model scoring service, and run sets of business rules to determine if action is needed and begin a workflow to manage the action. As a result, the truck might be scheduled for maintenance. This scenario is illustrated in Figure 3.

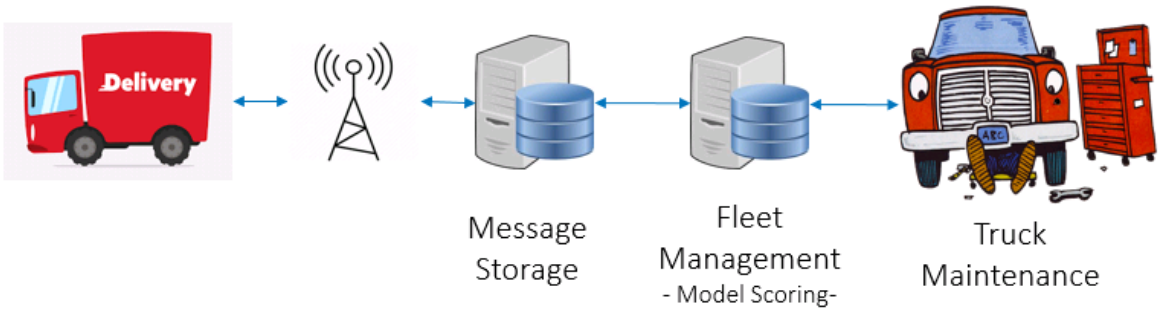


Figure 3. Fleet management system including model scoring.

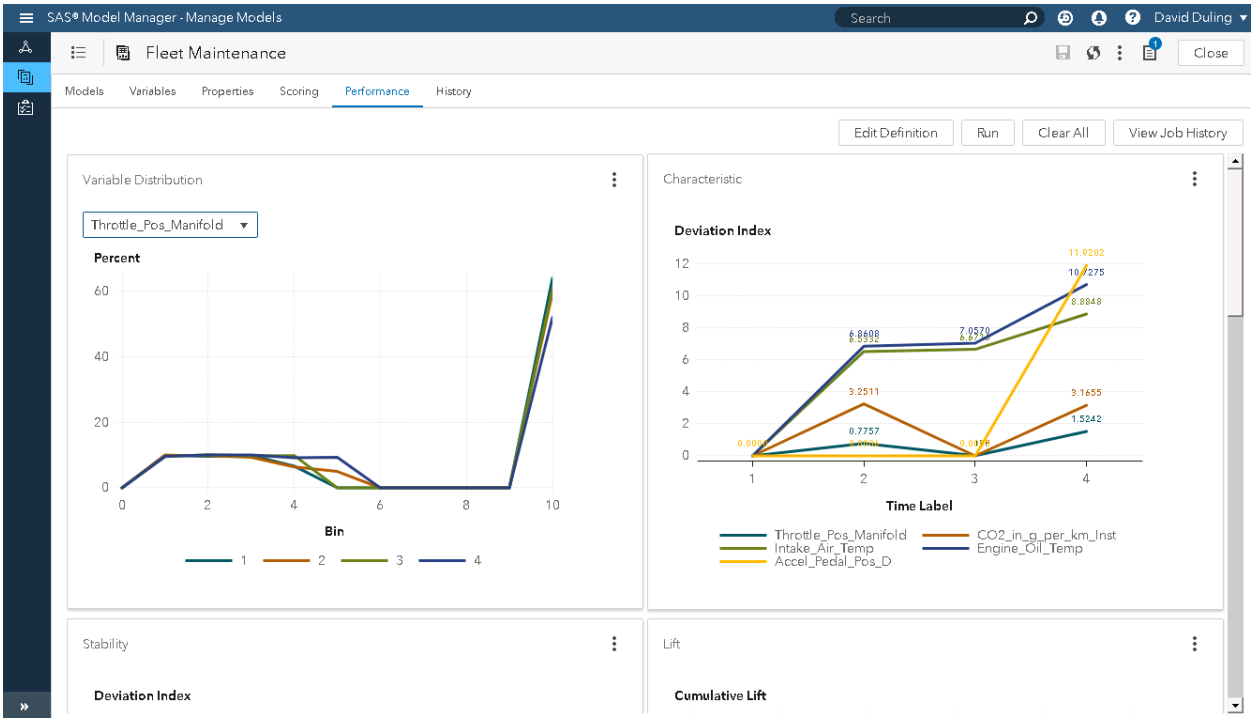
MODEL MONITORING

Once the model has been deployed in the production system, new data will be generated based on the model scores. This data should be stored in data sets for later analysis. This analysis is known as model monitoring. SAS® Model Manager® includes built in reports that compute the necessary measures for input and output data and the core fit statistics for classification and regression models. The first report we need to see is the variable distribution report which shows how the model input data is changing from the original model training data at subsequent each time-period. Variable change is important because small data value changes can have big effects on model accuracy. Each variable is divided into ten bins. The PSI is computed based on the deviation of proportions of data in each bin. We can name the original training sample **A** and the current scoring sample **B**.

Deviation: $\Delta_i = 100*(A_i/\mathbf{A} - B_i/\mathbf{B})$

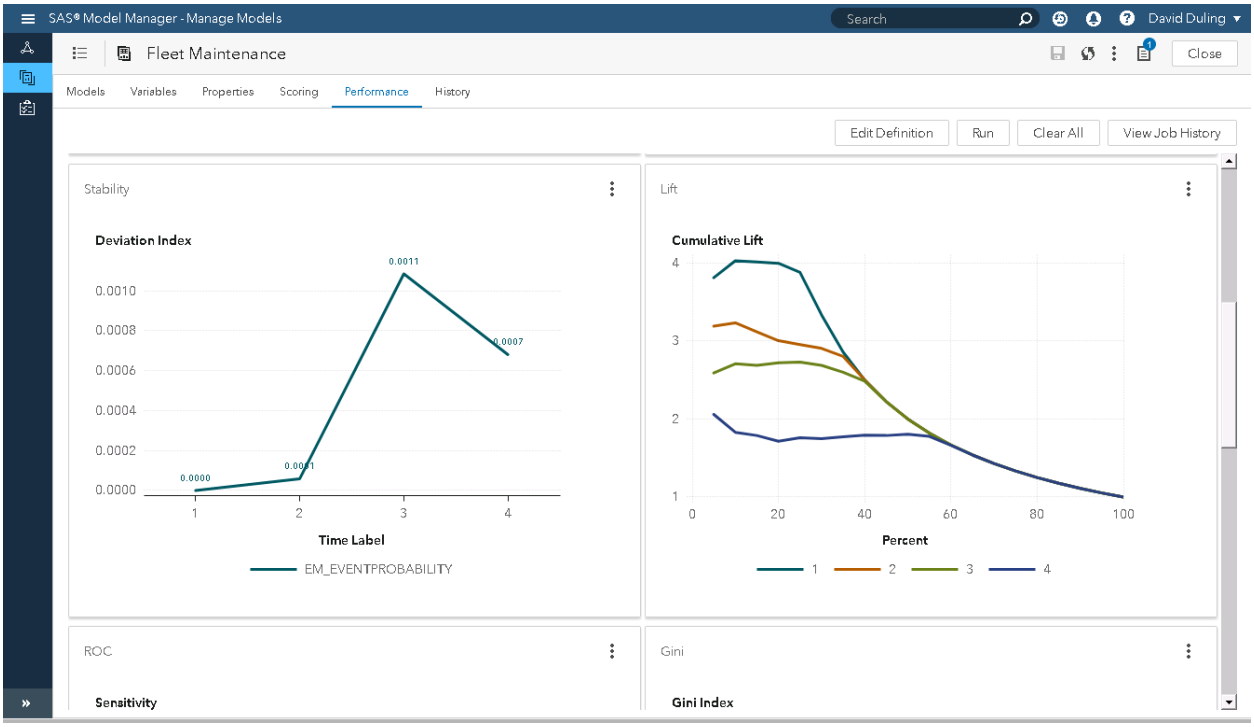
Index: $PSI = \sum_i(\Delta_i * \ln(A_i/B_i))$

In our truck fleet scenario, we are computing the model monitoring at four monthly time points. Each variable is divided into ten bins. In Display 9, the Variable Distribution plot shows the deviation in each bin for the Throttle_Pos_Manifold variable. In the Characteristic plot, the top 5 variables ranked by total PSI are shown at each time point (1,2,3,4). These variables are the ones most likely driving degradation in the model performance. The PSI is an absolute measure of deviation: larger values indicate greater amounts of deviation. The PSI does not show the increasing or decreasing direction of the deviation. One nice property is that PSI can be computed equally for continuous and categorical variables. You can use PSI to determine when the model is no longer reliable and needs to be replaced.



Display 9. Monitoring change in the distributions of predictor variables over time.

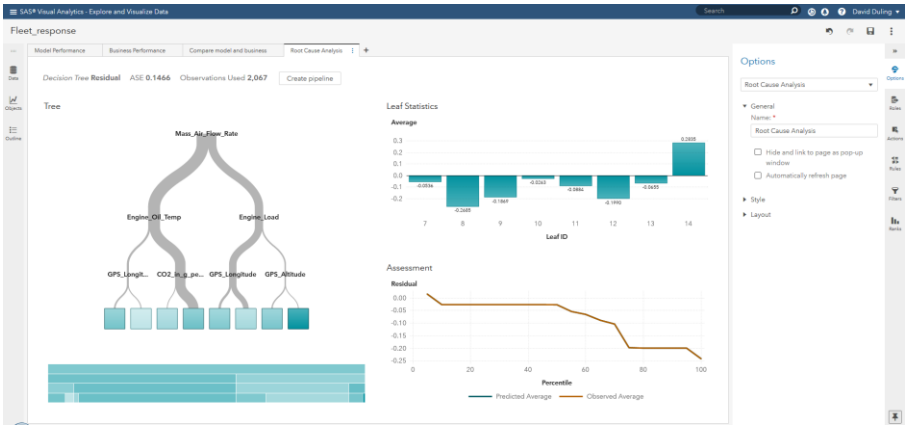
We can apply the same measures to the output of the models. Changes in the distributions of prediction or probability values also indicate that the data has changed indicating that the model might need to be replaced. We use the term Stability for deviation measures applied to model output. However, we can also measure model accuracy directly. After one month of time, the fleet management system will store records on actual truck maintenance events. At each event, we can determine if the truck needed maintenance for comparison to the most recent predictions by the models scores. We can measure model accuracy with model agnostic measures such as Lift, ROC, and Gini index. This calculation is the model accuracy portion of the model monitoring job. Display 10 shows both the stability measure of the predicted probability, and the Lift chart of model accuracy. Lift is a measure of relative model accuracy. The percentiles are created by ranking the model probabilities from high to low. In each percentile, the model accuracy is compared to the overall model accuracy. Higher values of lift indicate better predictive performance. In Display 10, we see both the model stability chart over the four time periods, and the model lift charts at each period. We can see there is a big change in stability in period 3 that is partially corrected in period 4. We can also see that model lift decreases in each period. These are clear indicators that the model performance is degrading.



Display 10. Model output value changes and accuracy changes over time.

MODEL ANALYSIS

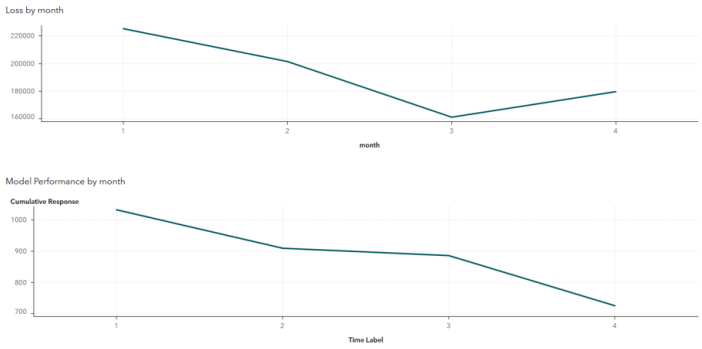
We can analyze change in model performance using standard statistical modeling tools that rank variable importance. This root cause analysis can reveal which factor are truly affecting change in accuracy. In the following analysis, we have created a new variable named Residual as the difference between predicted maintenance (0,1) and actual maintenance (0,1). We then model Residual based on the standard model predictors using a decision tree in SAS® Visual Analytics, as shown in Display 11. This model shows that the variables *Mass_Air_Flow_Rate, Engine_Oil_Temp,* and *Engine_Load* most contribute to model error. The business should ask their engineers to examine the factors influencing those variables to improve truck reliability.



Display 11. Root cause analysis of the model errors.

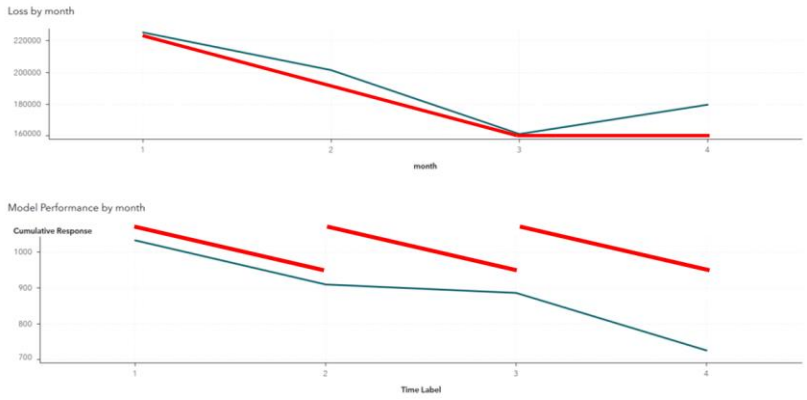
As a result, the business might also require the model be retrained to improve accuracy. The data scientist can trigger the retraining using SAS® Model Manager® for models developed in Model Studio or can directly reanalyze the data and produce new models. In either case, the models can be added to the original model as new versions. The model selection, testing, and deployment processes will then be repeated. The model monitoring jobs can then be executed in subsequent months and the performance of the project's models can be plotted and analyzed over many generations of the deployed model. This view provides a long-term perspective on the impact of predictive models on the business outcome.

We can also directly visualize the impact of model performance on the business objectives. We have data of model performance over time generated by the model monitoring jobs. We can also accumulate data over time on business performance for comparison purposes. In this scenario, we have accolated data on the number of truck maintenance events, maintenance costs, and loss of income due to the loss of truck usage during maintenance events. In Display 10, we have plotted both model performance and business losses by month on the same scale. We can see that in months 1, 2, and 3, that business losses declined, but that model performance was also declining. This shows that we were still deriving benefit from the model despite the degradation. However, in month four, the business losses increased indicating that perhaps model inaccuracy is not contributing to predicted maintenance false positives resulting it excessive loss of truck usage.



Display 11. Comparison of model performance and business loss over time.

We can correct that chart by retraining the models. However, we should ask the question what can do to improve cumulative business performance. Each month of model accuracy data contains the information needed to retrain the candidate models. If we retrained each month, rather than waiting for more significant model error, the monthly degradation in model performance would not change, but the cumulative degradation model performance should be reduced. Figure 12 illustrates this ideal situation. The red line added to the chart shows the expected effect of retraining the model each month. At the start of each month, model accuracy has been restored to the level of model training. If no new data is introduced and no new and improved modeling method is introduced, each model retraining should achieve the same level of accuracy. At the end of the month model accuracy has degraded by the expected amount. In this case, cumulative business losses should decline to their minimal value and remain at that amount. Appropriately frequent retraining of the model should result in minimal business loss. There is a limit to this effect. You should not retrain a model when an insufficient amount of new data has been accumulated, which would result in a less accurate model. You should retrain the model at a rate that does not match the business cycle. For instance, if the trucks were running 24 hours a day in three shifts, then you might not want to apply a model trained on data from the overnight shift on measurements taken in the daytime when ambient temperatures are higher.



Display 12. Ideal model performance degradation and improved business loss.

CONCLUSION

The modern analytic life cycle of model creation, model deployment, and model monitoring provides a strong foundation for using machine learning and artificial intelligence to improve business performance. However, there are many details that need to be addressed to deploy models into automated business processes. Following a diligent process of model selection, testing, deployment, monitoring, and analysis can make the outcomes more reliable and efficient. This paper has demonstrated the SAS approach to controlling the model lifecycle, and choices the business can make to make the outcome more effective.

REFERENCES

- Lam, Ming-Long. 2019. "Monitoring the Relevance of Predictors for a Model Over Time." *Proceedings of the SAS Global Forum 2019*, Dallas, TX. Available at http://support.sas.com/resources/papers/proceedings19/3448-2019.pdf
- Chu, Robert, et al. 2009. "Dashboard Reports for Predictive Model Management." *Proceedings of the SAS Global Forum 2009*. Washington, DC. Available at http://support.sas.com/resources/papers/proceedings09/045-2009.pdf
- SAS Institute Inc. 2018. *SAS® Visual Analytics 8.3: Working with Report Data.* Cary, NC. Available at https://go.documentation.sas.com/?cdcId=vacdc&cdcVersion=8.3&docsetId=vareportdata&docsetTarget=titlepage.htm&locale=en
- SAS Institute Inc. 2018. *SAS® Model Manager 15.3: Performance Monitoring.* Cary, NC. Available at https://go.documentation.sas.com/?cdcId=mdlmgrcdc&cdcVersion=15.2&docsetId=mdlmgrug&docsetTarget=n1t0lz86j0sd4vn11ta4jniiq5tk.htm&locale=en
- SAS Institute Inc. 2018. *SAS® Visual Data Mining and Machine Learning 8.3: User's Guide.* Cary, NC. Available at https://go.documentation.sas.com/?cdcId=vdmmlcdc&cdcVersion=8.3&docsetId=vdmmlug&docsetTarget=titlepage.htm&locale=en
- SAS Institute Inc. 2018. SAS® Micro Analytic Service 5.2: Programming and Administration Guide. Cary, NC. Available at https://go.documentation.sas.com/?docsetId=masag&docsetTarget=titlepage.htm&docsetVersion=5.2&locale=en

ACKNOWLEDGMENTS

The author gratefully acknowledges the help of Ming-Long Lam at SAS Institute.

CONTACT INFORMATION

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

David Duling
SAS Institute Inc
David.Duling@sas.com
www.sas.com