SAS4602-2020

Enabling Real-Time Stability Monitoring Models Using SAS® Viya® and SAS® Event Stream Processing

Sanjeev Heda, Sathish Gangichetty, and Steve Sparano, SAS Institute Inc.

ABSTRACT

Today with the Internet of Things (IoT), devices generate vast amounts of data that can be used to derive insights into the health and operation of an asset. To realize value from this data, analytics are needed in order to provide context for the state of the asset. In industrial applications, one such type of analytic is stability monitoring. These models provide an intelligent way to track the ongoing health of an asset where the metric of interest is checked against the bounds generated by the model. Automating such models to function in real time delivers immediate business value beyond detecting abnormal behavior because, in many cases, they provide the first step in identifying an invaluable window of opportunity to course-correct in mission-critical situations.

To that end, these models need to be deployed against streaming data where they can continuously analyze the data. Once abnormalities that merit actions are identified, these alerts need to be raised to a person who can take action. Supporting evidence of the data related to the alarm is also captured to confirm that this is a real issue and what action needs to be taken.

INTRODUCTION

Today, many assets across multiple industries are becoming more instrumented and connected to enterprise platforms to provide additional insight into their health and operation. IDC estimates that IoT investment will reach $1.12 trillion in 2023 (MacGillivray and Torchia, 2019). One key area for many Industrial organizations is using this data to improve predictive maintenance and increased reliability for these assets. Analytics play a critical role, but that is only one piece of the puzzle. An organization needs to accomplish three key steps in order to realize this value: develop, deploy, and operationalize. This paper provides a walkthrough of a relevant example to demonstrate the high-level framework for each step using SAS® Viya® and SAS® Event Stream Processing.

The main purpose of this paper is to show the following:

- How to create a stability monitoring model in SAS Viya using Python SAS Scripting Wrapper for Analytics Transfer (SWAT).
- How to register the stability monitoring model in SAS® Model Manager.
- How to consume models registered in SAS Model Manager in SAS Event Stream Processing.
- How to operationalize alerts and analytic outputs using open-source components.

Figure 1 illustrates the overall process and architecture that will be outlined in this paper.

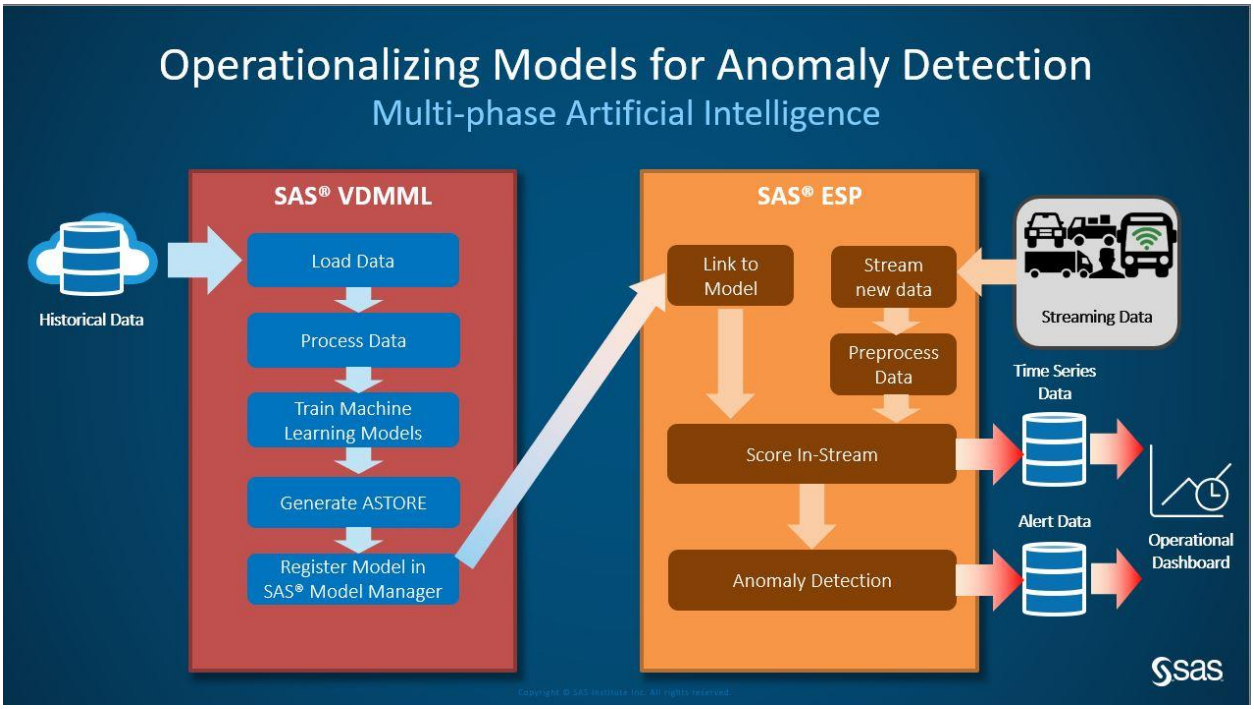

Figure 1: Overall Process and Architecture

BACKGROUND AND FOUNDATION

BACKGROUND OF PROBLEM STATEMENT

The example used in this paper comes from the Prognostics Center of Excellence at NASA's Ames Research Center. The data sets contain simulated results of turbofan engine degradation across multiple engines. There are a total of four data sets with various training and testing trajectories included. A given data set may have one or more failure modes present and one or more operating modes present. Each data set contains a unit number, time in cycles, three operational setting parameters, and 26 sensor measurements.

This example was previously analyzed using SAS and a stability monitoring model in SAS technical paper "Monitoring Turbofan Engine Degradation Using Stability Monitoring Procedures" but this paper did not go further than creating the model. To provide continuity with the earlier paper, this paper will use the same model type and same data set. This continuity includes using the same target variable to determine engine degradation and the same input variables into the model. This paper uses the training data set from FD003 for Engine #24, which contains a total of 494 cycles.

- Cycles 1-90 are used for creating the stability monitor model.
- Cycles 150 through 494 are streamed to SAS Event Stream Processing and scored by the trained stability monitoring model.
- The target variable is Ratio of Fuel Flow to Ps30.
- Input Variables are Pressure at fan inlet, Total pressure in bypass duct, Corrected core speed, Bleed enthalpy, HPT coolant bleed, and LPT coolant bleed.

FOUNDATION OF APPROACH

The foundation for this analytic approach is based the analytic lifecycle, which every analytic development process uses. The analytic lifecycle is comprised of three key phases: data, discovery, and deployment. (See Figure 2.)

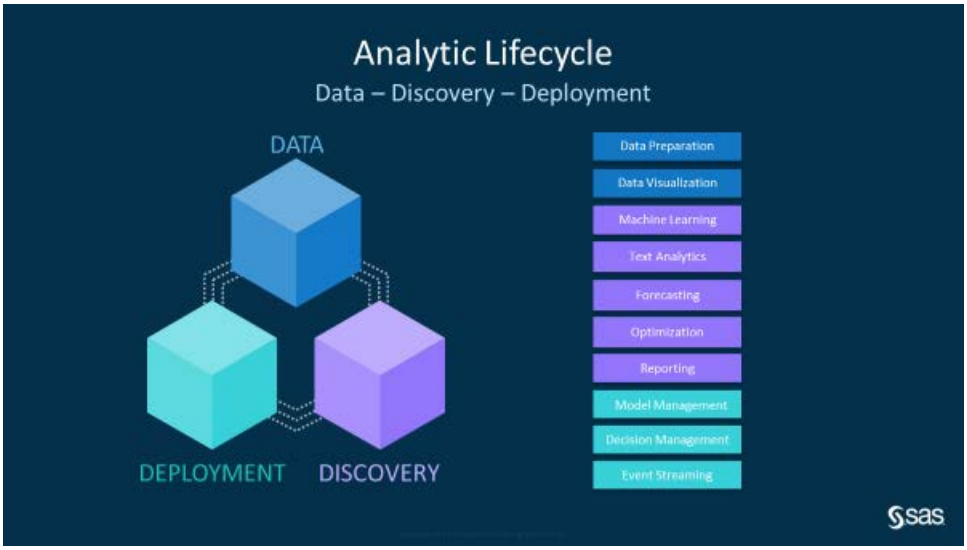

Figure 2: Analytic Lifecycle with Key Components and Examples Highlighted

The phases typically contain these following:

- Data – ingest and prepare various data sets into an analytically ready dataset
- Discovery – explore, visualize, build, and validate models on data to solve the business problem
- Deployment – operationalize these analytic models to realize business value

The analytic lifecycle is meant to be a continuous and iterative process as new learnings and requirements are identified. SAS enables the analytic lifecycle with the SAS Platform. For this predictive maintenance example, SAS® Analytics for IoT is the preferred analytics solution, because it includes artificial intelligence and machine learning analytics for high-frequency time series data, as well as the underlying capabilities to streamline data ingestion and preparation.

DEVELOP

CREATION OF STABILITY MONITORING MODEL

The turbofan engine data for this example has been collected and ready for analysis. If it had not been, additional data preprocessing and transformations would need to be performed on the data. The next step in enabling predictive maintenance is to create and validate analytics that provide context on the state of the asset and determine when an action should take place. You can perform this step using multiple analytic techniques such as threshold-based analytics, pattern matching, supervised/unsupervised machine learning, and so forth. In general, these analytics can be broken down into three categories:

1. Feature calculations (such as threshold base and Western Electric)
2. In-stream training and in-stream scoring (such as clustering methods and regression)
3. Out-of-stream training and in-stream scoring (such as gradient boosting and decision trees)

For this example, this paper uses the stability monitoring model, which falls into the third category. These models provide an intelligent way to track the on-going health of an asset, where the metric of interest is checked against the boundary conditions generated by the model. This requires training the model on historical data across various operating modes where the asset is mostly considered healthy and normal.

These are the steps used to create a stability monitoring model using SAS Viya with Python SWAT in a Jupyter Notebook:

1. Create connection to the CAS server using SWAT.
2. Load the stability monitoring action set from the CAS connection.
3. Load the data as a Pandas DataFrame from the CAS table (if data is not in CAS, load data into memory and upload into CAS).

```
cas_tbl = 'sm_base'
caslib = 'public'
cas_df = conn.upload_frame(df, casout=dict(caslib=caslib,
name=cas_tbl, replace=True))
```

4. Subset the data for the first 90 cycles of the engine data set to use for training.

```
train = cas_df[cas_df['cycle_number'] <= 90]

train['stable'] = 1
```

5. Persist the data on the server side by creating a new CAS table.
6. Set up the stability monitoring models by configuring project specifications using the stability monitoring calibration action. In this case, three stability monitoring models are created: Mean, Regression, and ARIMA. Note that the resultant ASTORE model is saved as a CAS table after training is complete.

```
In [15]: input_vars = ['pressure_at_fan_inlet', 'total_pressure_in_bypassduct',
                      'corrected_core_speed', 'bleed_enthalpy', 'hpt_coolant_bleed', 'lpt_coolant_bleed']
         target = 'ratio_of_fuel_flow_to_ps30'
         datetime_variable = 'cycle_number'
         stability_indicator = 'stable'

         default_model_spec = {'xvarnames': input_vars,
                               'xtsf': ['none', 'none', 'none', 'none', 'none', 'none'],
                               'xreqlist': ['0', '0', '0', '0', '0', '0'],
                               'targettsf': 'none'}

         models = [dict(id=1, type='REG'),
                   dict(id=2, type='ARIMA')]

         results = conn.stabilitymonitoring.smcalib(
             projdefs=[
                 {
                     'id': 1,
                     'modellist': models,
                     'projvars': dict(xvarnames=input_vars,
                                      targetname=target,
                                      dttmvarname=datetime_variable,
                                      stablevarname=stability_indicator),
                     'modelvars': [default_model_spec for i in range(len(models))],
                     'projdata':train_tbl
                 }
             ],
             projouttable=dict(name='outtab', replace=True),
             holdoutfittable=dict(name='holdout', replace=True),
             scoreouttable=dict(name='scoretable', replace=True),
             scoreinfotable=dict(name='scoreinfotab', replace=True)
         )

         NOTE:  Calibrating project ID = 1.
         NOTE:  Calibration successful for project ID = 1, model ID = 0.
         NOTE:  Calibration successful for project ID = 1, model ID = 1.
         NOTE:  Calibration successful for project ID = 1, model ID = 2.
```

Figure 3: Training Stability Monitoring Models in Jupyter Notebook Using Python SWAT

7. Once complete, the various models can be visualized to see how well they predict the actual value for the target variable

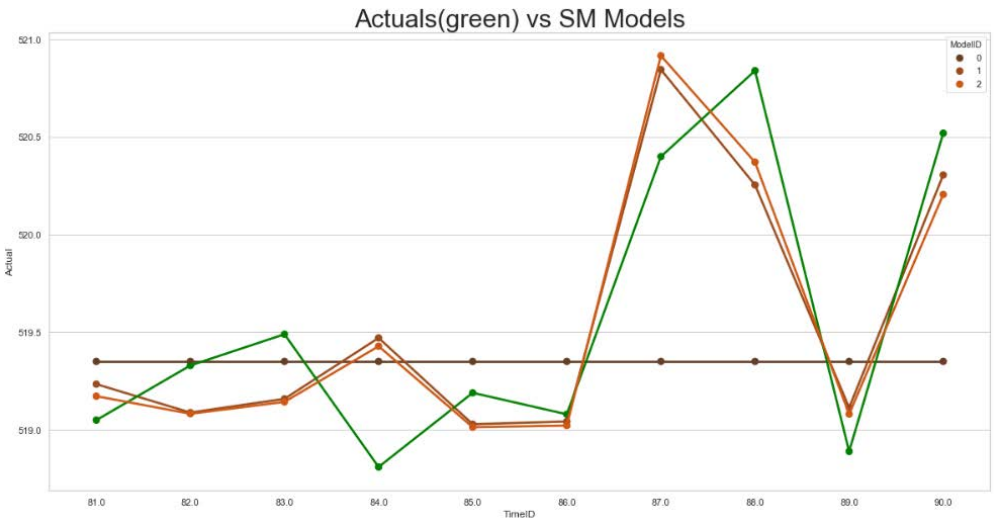

Figure 4: Comparison of Stability Monitoring Models Prediction Versus Actual Values

After completing this process, the mean absolute percentage error (MAPE) can be used to determine which model is the best in predicting this value. In this example, the ARIMA model (identified by ModelID=2) performs the best. This model is the one that we select to be scored in real-time using SAS Event Stream Processing.

REGISTERING MODEL IN SAS MODEL MANAGER

After a model is selected and validated, it can then be registered in SAS Model Manager. Registering a model in a centralized repository is a critical step in the process for a variety of reasons, including governance, security, lineage, monitoring, and model deployment. Registering is necessary for high-valued assets such as analytic models that can be critical to business processes.

The stability monitoring ASTORE model created in the previous section is stored as a CAS table named scoretable. We can use Python SWAT to copy this model from this CAS table to SAS Model Manager, as seen in Figure 5.

```
In [28]: from sasctl import Session
         from sasctl.tasks import register_model

         sm = conn.CASTable('scoretable')

         with Session(hostname=f'http://{host}/cas-shared-default-http',
                      username=user, password=pswd, protocol='http', port=80,
                      verify_ssl=False):
             register_model(sm, 'smodel_swat', 'stability_monitoring',
                            force=True,version='latest') #set force = False  if  writing to existing project

         NOTE: Added action set 'astore'.
         NOTE: Cloud Analytic Services saved the file _88C8E2A612AA4BA1847259902.sashdat in caslib ModelStore.
```

Figure 5: Registering Stability Monitoring Model into SAS Model Manager

Figure 5 shows that the model is named smodel_swat, and that it is contained in the SAS Model Manager project stability_monitoring. Figure 6 shows this model in SAS Model Manager.

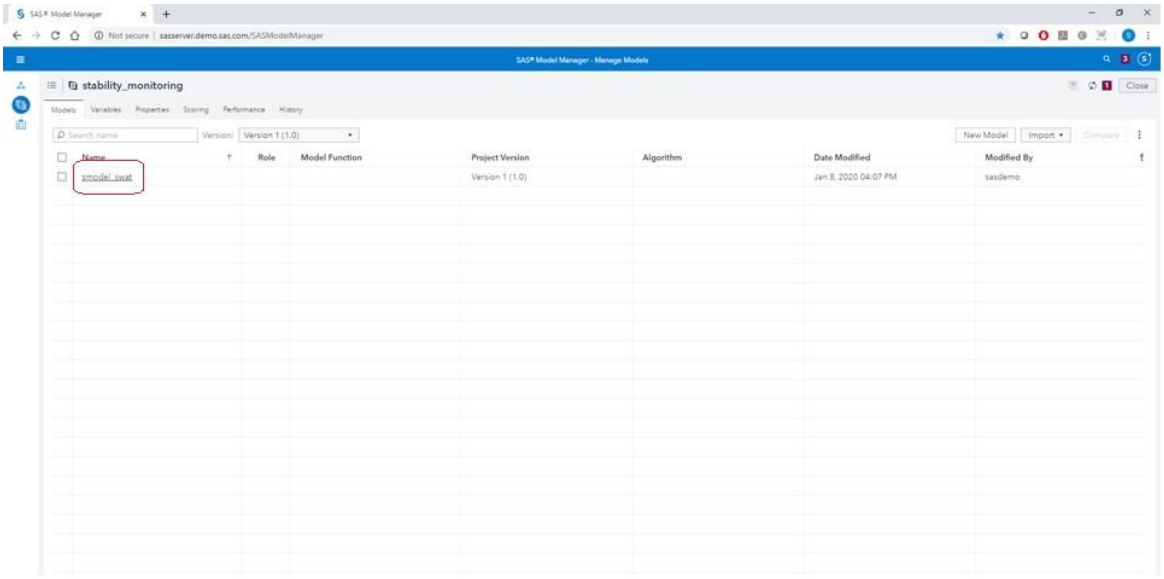

Figure 6: SAS Model Manager with the Registered Stability Monitoring Model

Previously, you could register these types of models in SAS Model Manager, but you could not execute them in SAS® Micro Analytic Service. In SAS Viya 3.5, the process of registering certain types of models in SAS Model Manager was updated to allow for this execution. The process now wraps DS2 code around the model, which enables it to be scored. However, additional parameters are required before stability monitoring models can be scored. For this example, the ASTORE file contains three model types: Mean, Regression, and ARIMA. The stability monitoring model requires that we define values for ProjectId and ModelId. For this reason, we must modify the DS2 wrapper code before we can use this model in SAS Event Stream Processing.

To access the DS2 wrapper code, select the model in SAS Model Manager. This displays the various files and information associated with this model. The dmcas_packagescorecode.sas program contains the DS2 wrapper code to execute this model. hereafter you open this program, you can make the modifications illustrated in Figure 7. For this example, because the ARIMA model was determined to be the best, ProjectId is set to 1 and ModelId is set to 2. Note that the prefix to setting these options must match how the score package is initialized.

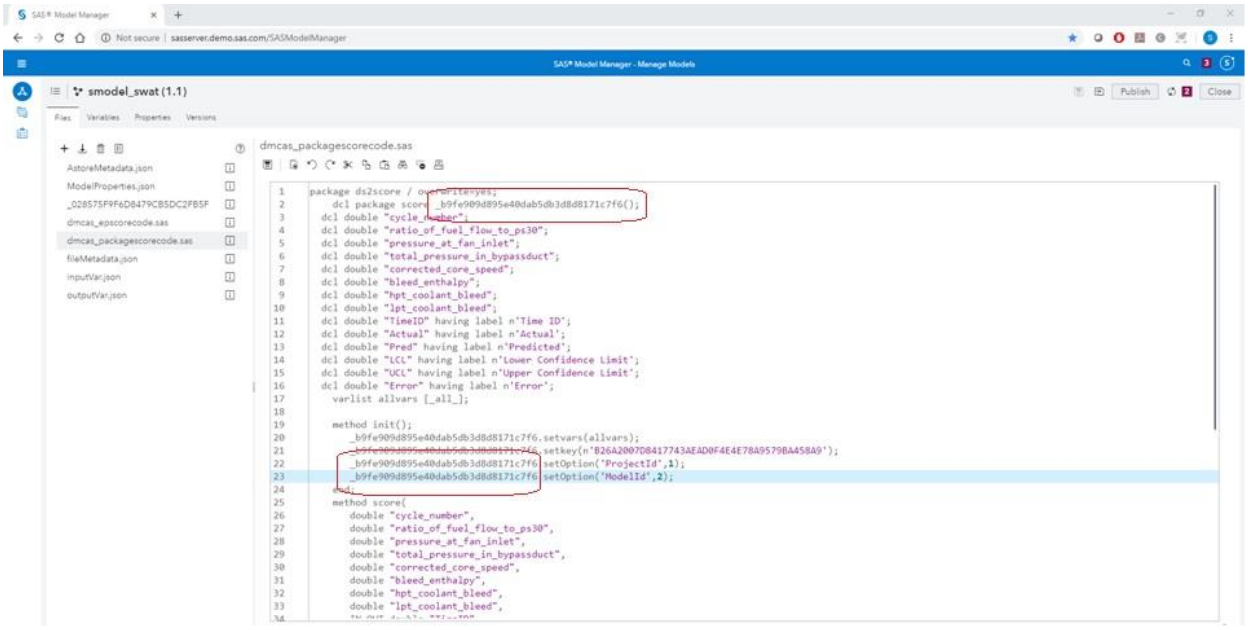

Figure 7: Update to DS2 Wrapper Code for Model

After the modifications are complete, the updated file can be saved. Back in SAS Model Manager, you can now select smodel_swat and mark it as the champion. Only champion models in SAS Model Manager can be consumed in an SAS Event Stream Processing analytic.

DEPLOY

CONSUMING MODEL IN SAS EVENT STREAM PROCESSING

Now that the model has been created and validated to perform its objective, we must now deploy the model against the data that is being generated by these assets. Typically, industrial assets can produce a tremendous amount of data, and they require minimal latency in order to enable a value-added action to be taken. For this example, SAS Event Stream Processing is used to execute analytics against the streaming data to meet these objectives. For other business use cases and requirements, SAS provides a variety of model deployment options including in-database, in-Hadoop, ad hoc, batch, and so on. This flexibility of the various options is useful when handling various business use-cases with their own requirements.

SAS Event Stream Processing provides a flexible and adaptable analytic engine that can be deployed along any point of an enterprise deployment architecture. The analytic engine can publish and subscribe to over 300 endpoints and can execute anywhere from an IoT edge device beside an asset to a distributed cloud environment. In addition, SAS Event Stream Processing provides streamlined integration with models registered in SAS Model Manager. This ability is a differentiator for a couple of reasons. As an SAS Event Stream Processing analytic is deployed, the model selected from SAS Model Manager is retrieved and deployed as part of the analytic package. In addition, any changes made to the model registered in SAS Model Manager provides notifications to the SAS Event Stream Processing analytic that a change has been made. This provides lineage and governance of describing where and how the models are being used.

To get started, a new SAS Event Stream Processing project is created with a blank canvas. To enable data to stream into this analytic, add a Source window to the canvas. For many industrial applications, this Source window can connect to an OPC-UA, Pi Historian, or a message bus technology such as MQTT or Kafka. For this example, the data is connected using a CSV file in order to stream the data. The input schema must be defined the expected

data columns. Note that, for the model to score properly, the input schema must match what the model is expecting or it must be mapped properly in the Calculate window. Figure 8 shows the Source window with the input schema defined added to the SAS Event Stream Processing project and the Calculate selected, which is the next window that is added to the project.

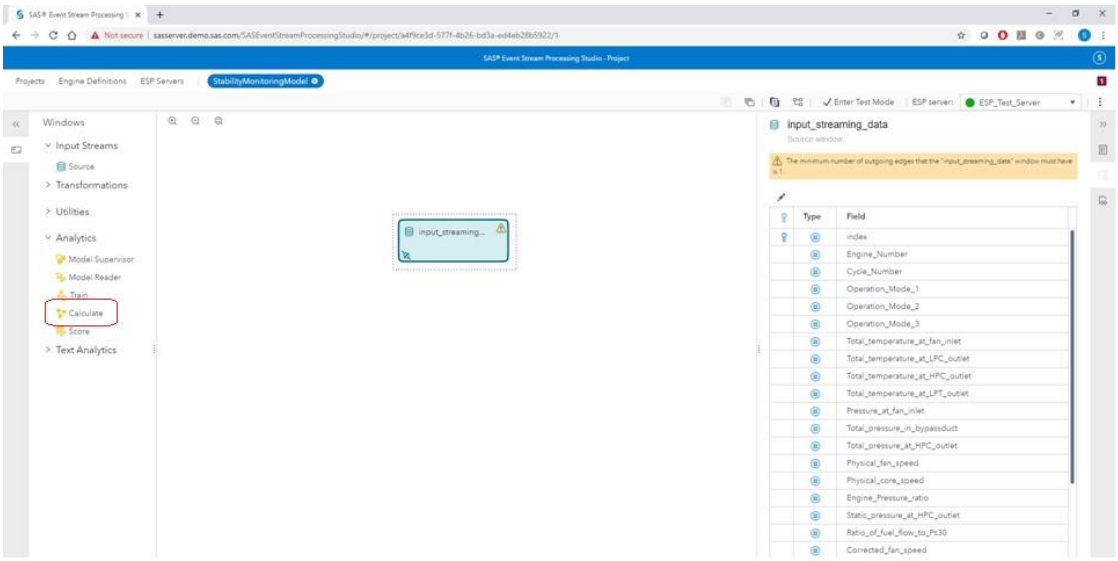

Figure 8: SAS Event Stream Processing Project Canvas with Source Window

The Calculate window contains many algorithms available to use for streaming data. In addition, it also supports the use of SAS Micro Analytic Service modules, which includes importing models from SAS Model Manager. After adding this window to the canvas and connecting it to the Source window, we must change the configuration of the Calculate window to what is seen in Figure 9.

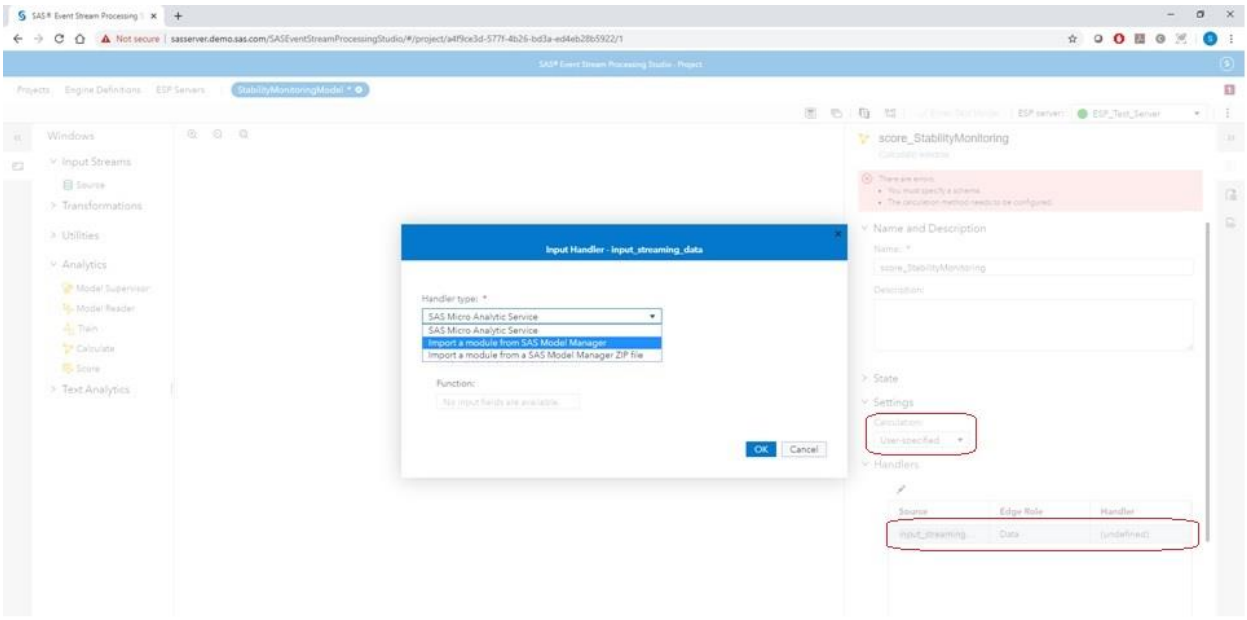

Figure 9: Configuring Calculate Window to Ingest Model

The first step is to select a value of User-Defined for the Calculation. Next, double-click the Handler Source to select what method to used. This creates the pop-up dialog box shown

in Figure 9. From here, we can open SAS Model Manager and select the stability monitoring model that has been registered, as seen in Figure 10.

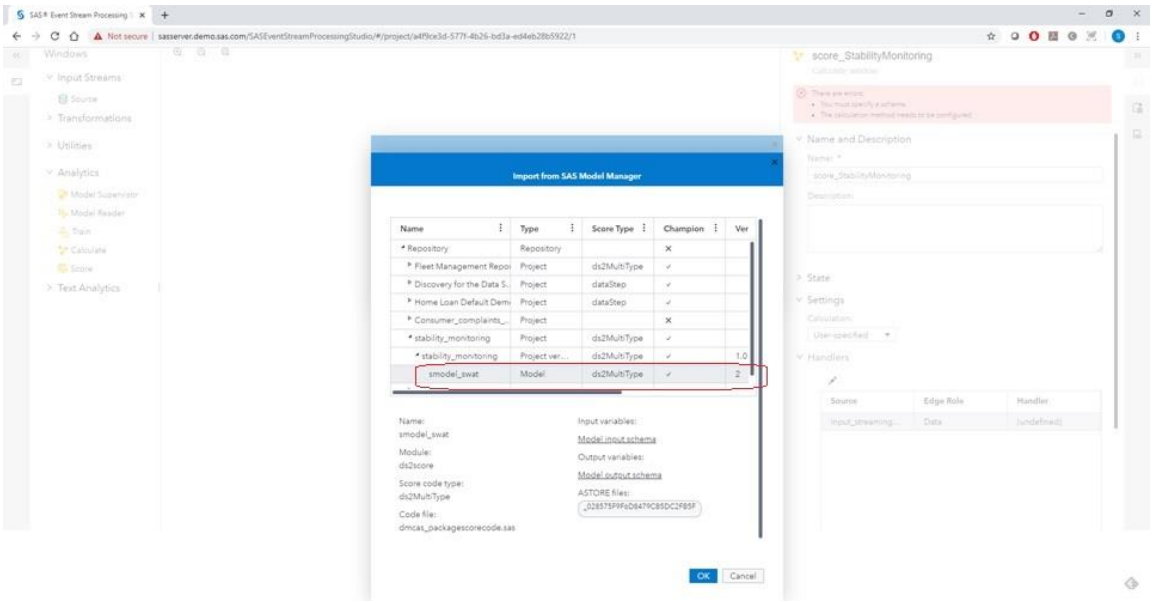

Figure 10: Selecting Model from SAS Model Manager Repository in SAS Event Stream Processing

Here all the models available in the repository are visible along with their associated projects, versions, and whether or not they are champions. After a model is selected, the input and output variables are available to view and confirm. There is also an option to automatically add the output fields to the Calculate window schema.

At this point, the model has been selected and linked to this SAS Event Stream Processing project. It is ready to score incoming data after data is streamed into the project. However, in many industrial applications, additional processing of the data is required.

ADDITIONAL DATA PROCESSING AFTER SCORING DATA WITH MODEL

Typically, in an industrial application, an SAS Event Stream Processing project contains the following components:

- Data ingestion – connect to one or more required data sources.
- Data pre-processing – prepare the streaming data for analysis, including filtering, transforming, aggregating, joining, and so on.
- Model scoring –use the referenced model from SAS Model Manager to score the prepared data.
- Alarm management – determine what alarms should be escalated for further processing.
- Analytic outputs – send the necessary data to downstream systems for consumption.

Alarm Management

So far, we have discussed the first three items in a situation where data pre-processing is not required because of the data set that is being used. Alarm management is a key component. In industrial applications, it is important to determine when an action needs to take place in order to derive value. Confidence is required to know when the action needs to take place. Because the primary data source is time-series data, a single observation is probably not enough to take an action. Typically, the abnormality must persist over some range of time before enough confidence is established. For this example, abnormalities are determined if

the target variable is above the upper control limit (UCL) calculated by the stability monitoring model. However, to establish confidence, the alarm is generated to downstream processes only when at least 10 cycles have shown the target variable above the UCL.

In addition, after the process has determined that an action should be taken, time is required to resolve the issue. Because downstream processes and people are already working to resolve the issue, there is little to no value in repeatedly escalating the same alarm. These subsequent alarms should be suppressed until certain criteria is met to start escalating alarms. For this example, after an alarm is generated, subsequent alarms are suppressed for the next 25 cycles. After this time has cleared, a new alarm can be generated.

Analytic Outputs

As the SAS Event Stream Processing project has completed data ingestion, model scoring, and alarm management, the resultant outputs need to be generated. Outputs of the project can be sent in a variety of ways depending on business requirements, such as e-mail notifications, SMS, and external data systems. Each organization has their own needs and requirements to enable their people to consume these outputs. The delivery of alerts and data for an organization should align to the tools and interfaces used to drive their business decision process. SAS and SAS Event Stream Processing provides the flexibility to integrate with existing systems and platforms with native data integrations, open APIs, and so on. This is a critical piece, because the analytic outputs may need to be delivered to multiple downstream systems, such as maintenance workflows, time-series databases, alarm management, and so on. For this example, the output time series data and alarms are ingested into a Postgres SQL database in their own respective tables.

DEPLOYING USING SAS® EVENT STREAM MANAGER

With the SAS Event Stream Processing project complete, you can version and publish this project to make it ready for deployment. One option to deploy this project is to use SAS Event Stream Manager, which is an application that enables you to manage your SAS Event Stream Processing environment. You can use SAS Event Stream Manager to perform the following tasks:

- Administer and manage SAS Event Stream Processing Servers
- Deploy SAS Event Stream Processing projects in test and production environments
- Monitor the status of projects and deployments

If a new model is registered in SAS Model Manager or if a new version of your SAS Event Stream Processing project is published, SAS Event Stream Manager will notify the user of the changes and request to deploy the updates. This is one option that can be used for deploying analytics. Some organizations have their own methods to manage software deployment where the SAS Event Stream Processing project and associated artifacts can be deployed via these methods.

OPERATIONALIZE

At this point, the analytic is executing and the results are being stored in the Postgres SQL database. However, no value is realized until an action is taken place as a result of this analytic. In an ideal scenario, automated actions can take place if the analytic provided some detectability with 0% false positives and prescriptive instructions of the action that should take place. From a journey of maturity for predictive maintenance, this is the end state where organizations aim to be. However, because most organizations are starting their digital transformations, multiple people are typically involved in the process and need be able to review these alerts to determine what, if any, action needs to be taken.

The decision support process implemented by an organization is critical in ensuring that value is captured. For complex assets and their associated alarms, multiple people may be involved at various steps in the process, each of whom have their own requirements to provide inputs. When an alarm is generated, questions are needed to address what a person needs to know to determine the next steps in the decision support process. Questions may include:

- What data is needed? This can include analytic results, historical time-series data, asset data, and so on.
- What format does the data need to be presented for easy consumption?
- How would the person consume the information? This can include separate visualization, existing systems, and so on.

For this example, the open-source dashboarding tool Grafana is used to read the Postgres SQL data and display the results. An example dashboard created for this example can be seen in Figure 11.

Figure 11: Grafana Dashboard Presenting Alarms for Turbofan Engine

This dashboard can be customized to show what the end user needs to see to decide of what steps to take next. As decision and actions are being made, tangible business value is realized.

CONCLUSION

This paper provides a walkthrough of an example to demonstrate a high-level framework for how an organization can realize value in enabling predictive maintenance with their connected assets. This paper did not provide in depth overview of data pre-processing or exploring various models to determine what performed the best, although these are key processes in the analytics lifecycle. Instead, the focus was more on how a model such as a stability monitoring model can be created, validated, registered, deployed, and operationalized to help realize business value. There are a couple of important takeaways to have with this paper.

The first important takeaway is to have flexibility and adaptability in the framework used to solve a variety of business needs. With connected assets, there can be significant variation present from various equipment designs, connectivity, sensing packages, operational profiles, and so on. It is not one size fits all, where you can universally apply a single analytic technique to solve all the business challenges. Each model has its own strengths and weaknesses, which should be considered when analyzing a business use case. This evaluation is where lineage

and governance are important, so you can know not only what models are available, but how were they tested and validated in case the underlying data or asset behavior changes.

This paper highlights the initial creation and operationalizing of an analytic and model. The second important takeaway is that, over time, monitoring the performance of the analytic model is critical, because asset operations may change, equipment will degrade, and business requirements will change. If no action is taken on a degraded analytic model, the value realized will also diminish. As mentioned before, the analytics lifecycle is meant to be continuous and iterative. This allows introduction of enhanced analytics and algorithms to meet the ever-changing business needs.

REFERENCES

MacGillivray, Carrie and Torchia, Marcus. 2019. "Internet of Things: Spending Trends and Outlook." *Web Conference Proceeding: Tech Supplier*. Framingham, MA. IDC. Available https://www.idc.com/getdoc.jsp?containerId=US45161419

Saxena, Abhinav and Goebel, Kai 2008. "Turbofan Engine Degradation Simulation Data Set." NASA Ames Prognostics Data Repository. NASA Ames Research Center, Moffett Field, CA. Available https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

SAS Institute Inc. 2014. SAS Technical Paper. "Monitoring Turbofan Engine Degradation Using Stability Monitoring Procedures". Available https://support.sas.com/content/dam/SAS/support/en/technical-papers/iot/Monitoring-Turbofan-Engine-Degradation-Using-Stability-Monitoring-Procedures.pdf. Accessed 11/15/2019.

Saxena, Abhinav, Goebel, Kai, Simon, Don, and Eklund, Neil. 2008. "Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation". Available https://ti.arc.nasa.gov/publications/154/download/.

ACKNOWLEDGMENTS

The authors would like to thank Gul Ege, Arin Chaudhuri, Byron Biggs, Sergiy Peredriy, David Duling, Bryan Saunders, Sandy Defelice and Andy Tracy for all their help and insights on this project.

CONTACT INFORMATION

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

Sanjeev Heda
SAS Institute Inc.
Sanjeev.Heda@sas.com

Sathish Gangichetty
SAS Institute Inc.
Sathish.Gangichetty@sas.com

Steve Sparano
SAS Institute Inc.
Steve.Sparano@sas.com