Platform overview: Analytics

Machine learning, governance and deployment with SAS Viya

Antti Heino, Cloud Advisor on ML&AI

3.3.2021

Antti Heino

Cloud Advisor on ML&AI, SAS Finland

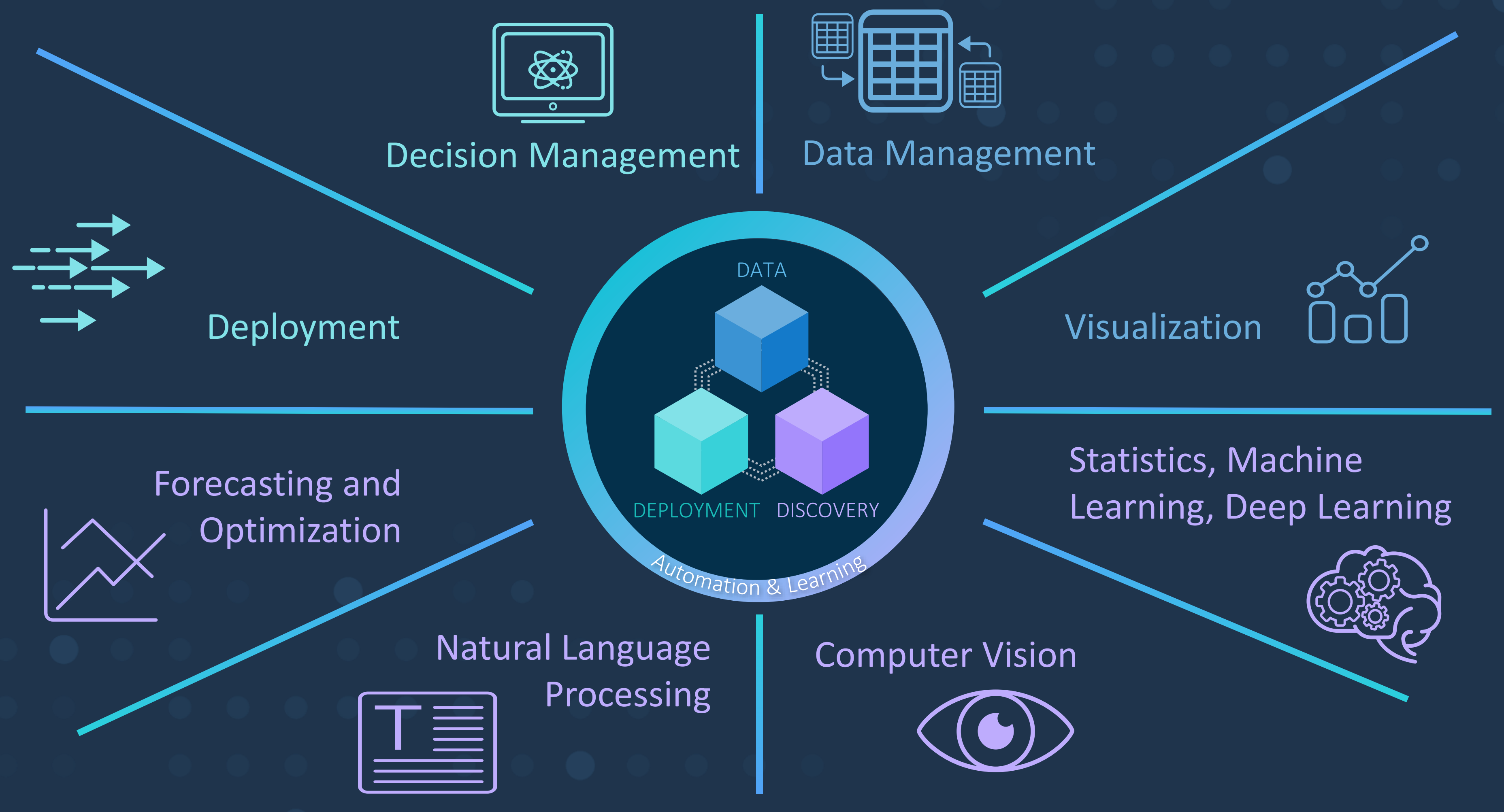
Let's connect

- 3 years of analytics advisory at SAS
  - Predictive analytics
    - Fraud detection
    - Predictive maintenance
    - Industrial process analysis
  - Text Analytics
    - Customer feedback analysis
    - Maintenance log understanding
    - Internal document classification & search
    - Healthcare document classification
  - Computer vision
    - Object detection
    - Video analytics
  - ....

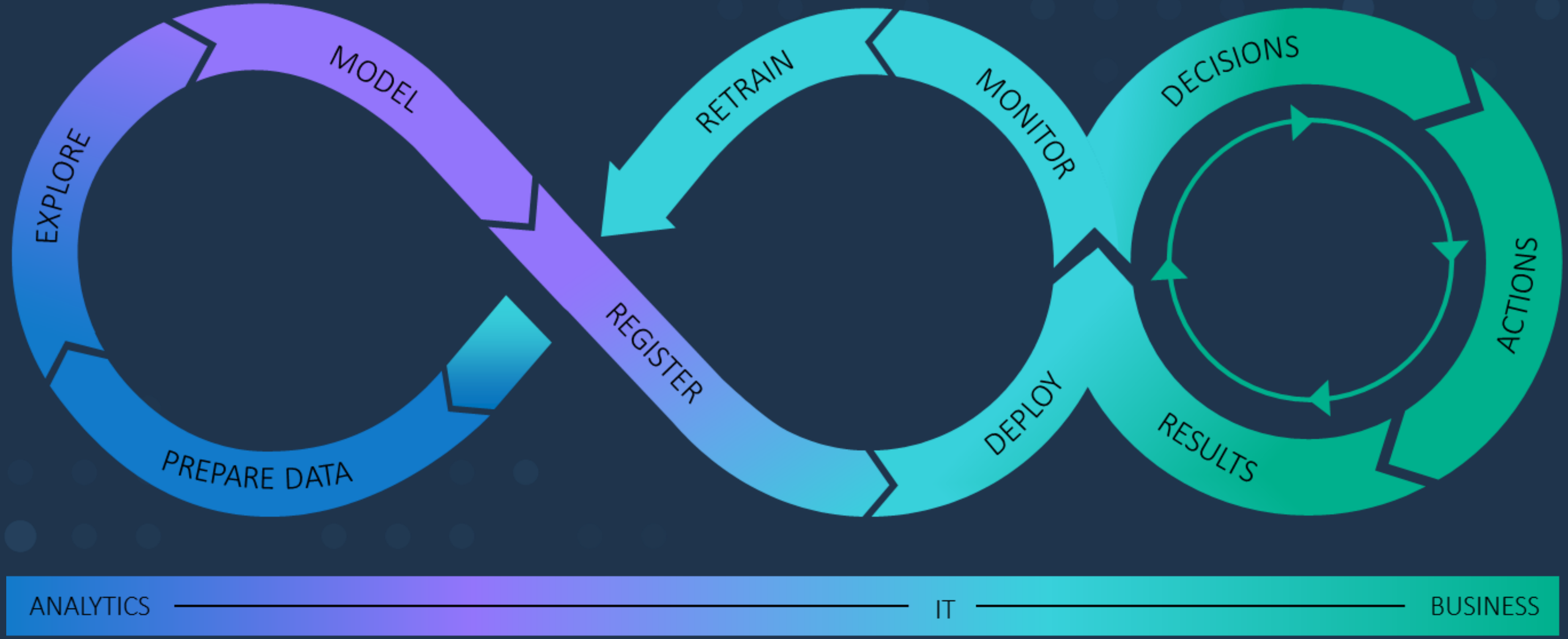
- 3 seasons of AI podcast "Tekoäly Nyt"

- 5 years prior experience on analytics consulting

- Free time
  - Travel
  - Diving
  - Sports


Decision Management
Data Management
Deployment
DATA
DEPLOYMENT
DISCOVERY
Automation & Learning
Visualization
Forecasting and Optimization
Statistics, Machine Learning, Deep Learning
Natural Language Processing
Computer Vision

Operationalizing Analytics



Georgia Pacific

one of the world's leading makers of tissue, pulp, packaging, building products and related chemicals
30k employees, 180+ locations world-wide

*"I think we have about **1,900 models that run multiple times a second. Each one of them is deployed in the SAS platform** to help us in each of those three buckets [process, asset and safety] we talked about. We constantly need to do more with those models or make better models."*

Roshan Shah
VP of Collaboration & Support Center and Advanced Analytics

Article:
https://diginomica.com/georgia-pacific-cuts-complex-data-modelling-times-half-sas

Video:
https://vshow.on24.com/vshow/Global_Forum/exhibits/Industry_Connection
Manufacturing -> Digital Transformation at Georgia-Pacific

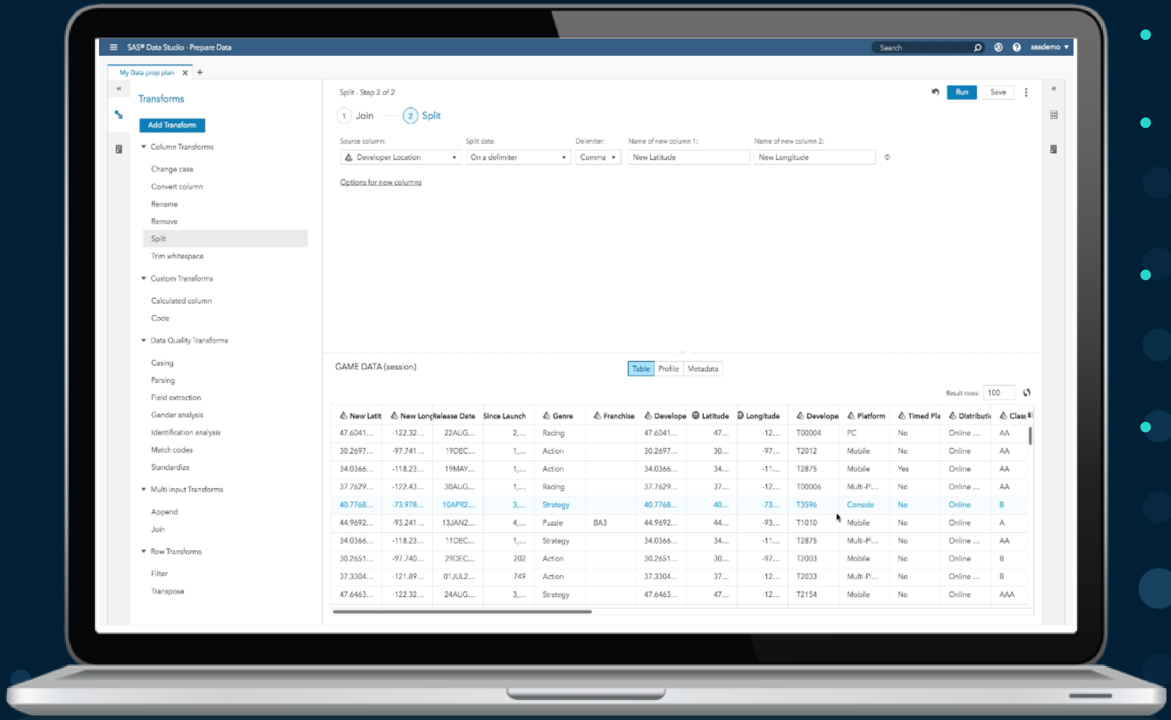
“You can have multiple folks going into the platform and being able to build and deploy those models, really, really quickly. Something that we used to struggle with in the past is we would build a model, but then we would hand that off to IT. Not that there's anything wrong with that, it's just we didn't know how to add all the additional things - such as error handling - that need to go with it.

What we've been able to do with SAS is we can enable a citizen data scientist, or an engineer, to build those models. And what really matters, they can go from taking a complex time series data and building a neural net within minutes and hours. They can go and deploy it.”

"It used to take us on average about twelve weeks to take a complex model, get it built, and deploy that, and then put it in production. I think we could easily say that that's gone, on average, about three to six weeks, sometimes even shorter. That's pretty unprecedented.

We don't care so much about it being perfectly accurate. We look for time to value of money. These folks are able to do that, which in turn, frees up the data scientists to solve the much more complex problems. If you look at it from IT's perspective, it really frees them up to focus more on, how do you make data available."

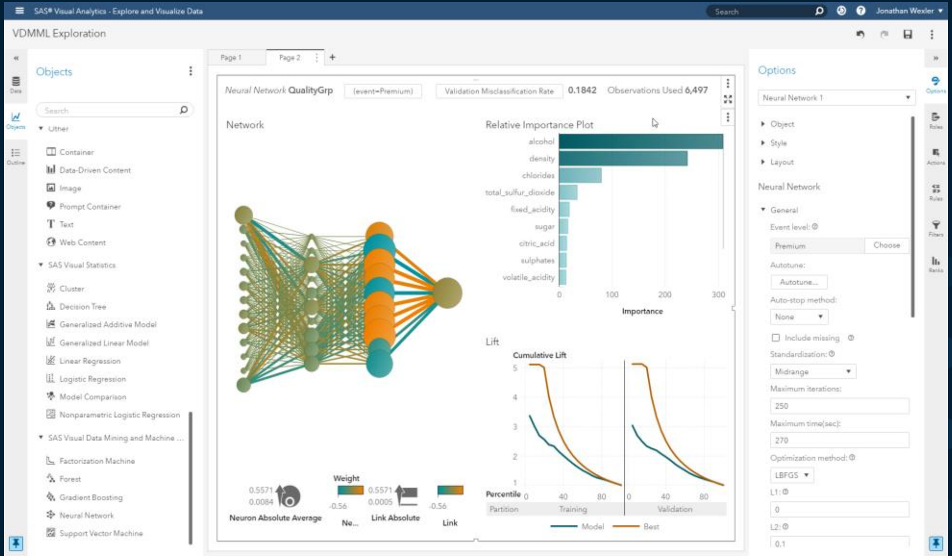
Data Preparation



- Access to different data sources
- Training-Validation Data Partitioning
- Feature Engineering (e.g. parameters, interactions)
- Variable selection and missing values

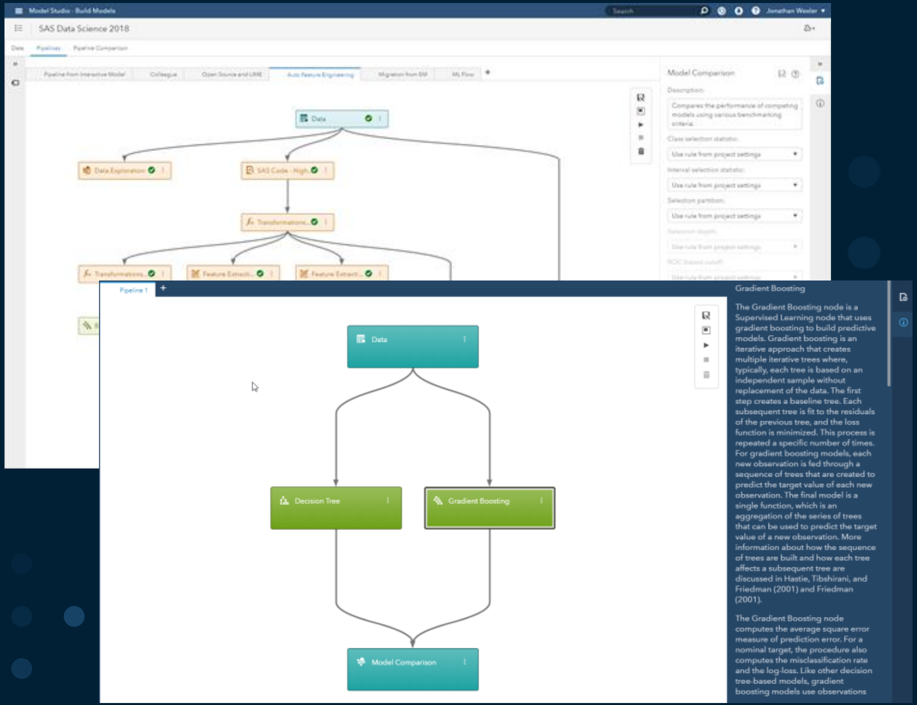
Visual exploration & Machine Learning

- Interactively discover relationships, trends, outliers
- Smart autocharting
- Analytics driven visualizations
- Explore predicted outputs
- Variable Transformation



- Decision Forests
- Neural Networks (Deep Learning and Computer Vision)
- Gradient Boosting
- Support Vector Machines
- Factorization Machines
- Bayesian Networks
- Dirichlet Gaussian Mixture Models
- Semi-supervised learning
- T-SNE

Model Studio



- Pipeline of activities
- Best practices templates
- Automated feature engineering
- Drag and drop and access to code
- Nodes are run asynchronously
- Algorithm annotations
- R and Python support

Coding interfaces

Getting Started with SAS Viya Programming

Access the capabilities of the SAS Viya through Cloud Analytic Services (CAS) actions for data access and analytics, run directly from SAS applications.

REST

REST APIs for any client language to access SAS analytics, data and services. The REST APIs are written to make it easy to integrate the capabilities of SAS Viya to help build applications or create scripts.

Python

SAS integrates with Python through various code libraries and tools that allow open source developers to unite the Python language with the analytic power of SAS.

Getting Started with SAS Viya for R

Combine R language functions with SAS through various code libraries.

Getting Started with SAS Viya for Java™

Java APIs for using SAS Viya CAS actions

Getting Started with SAS Viya for Lua

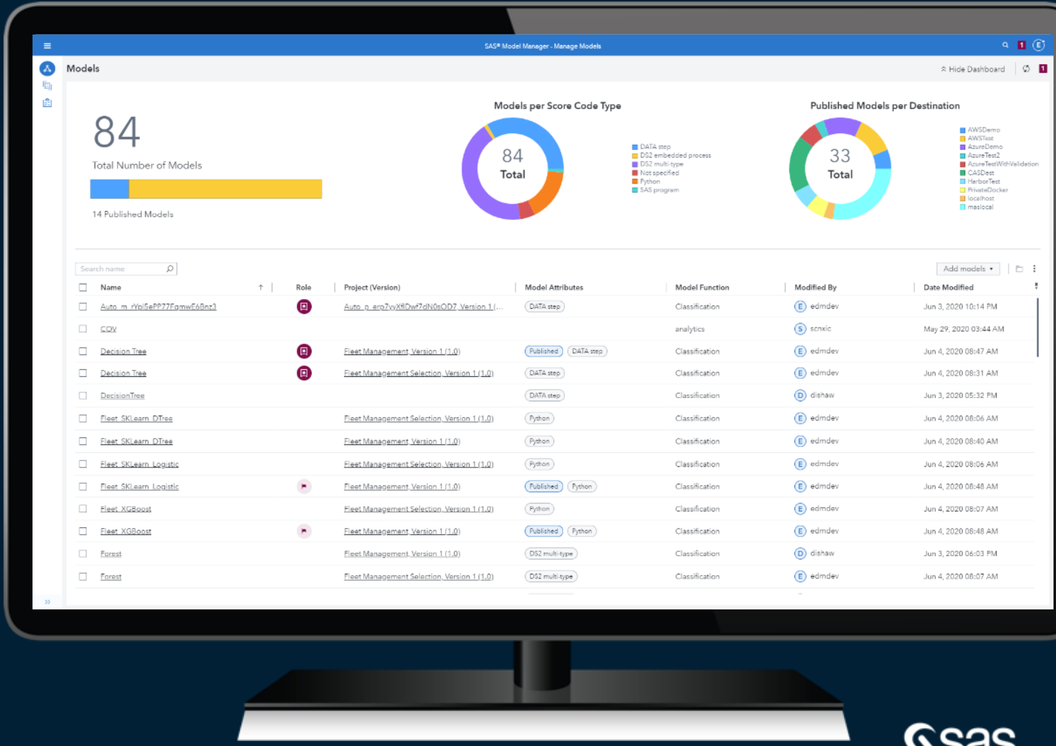
Lua APIs for using SAS Viya CAS actions

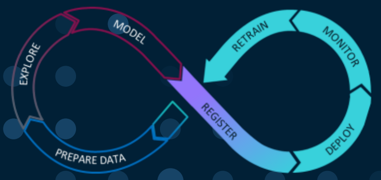
Developer.sas.com

Register

Organize & Manage Analytic Assets

- Central, searchable repository for *all* models/pipelines
- Compare models side-by-side
- Version control and track project history

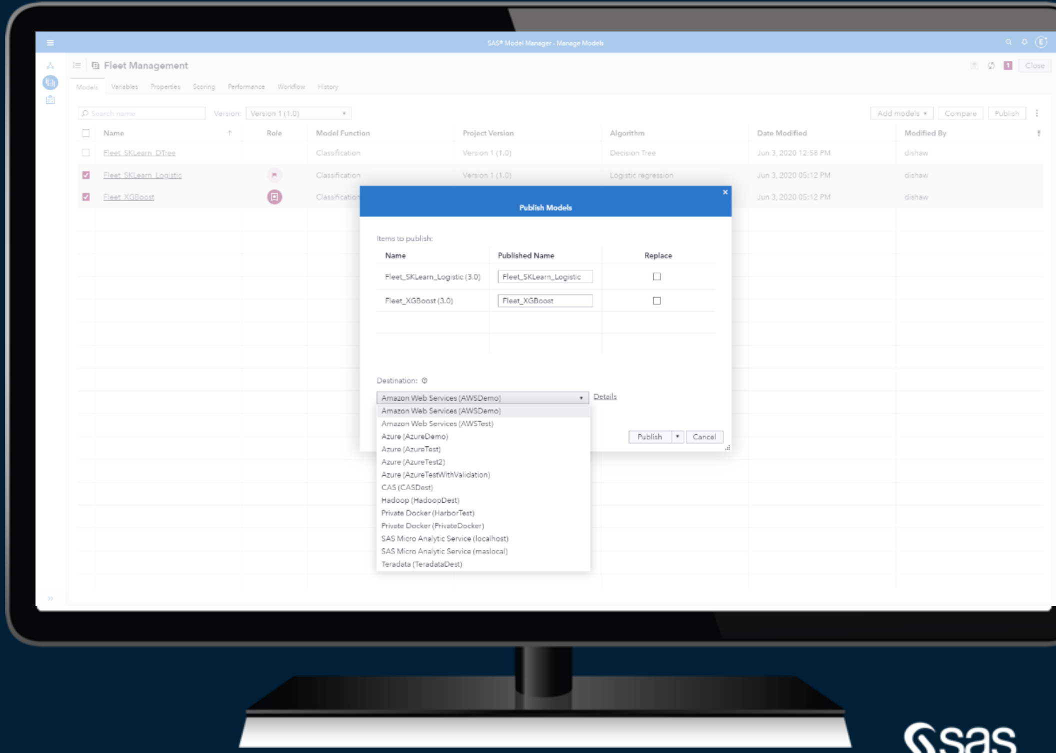


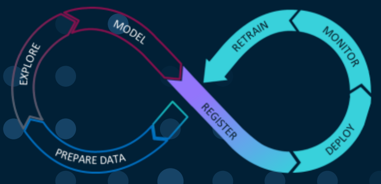


Deploy

Deployment and Scoring

- Quick and easy access to different production environments
- Deploy in-batch, streaming, cloud or edge device
- Azure Container Publishing Destination for Open Source models
- Supported publishing destinations include SAS Cloud Analytic Services (CAS), Apache Hadoop, SAS Micro Analytic Service (MAS), Teradata, as well as container destinations such as Amazon Web Services, Azure, and Private Docker

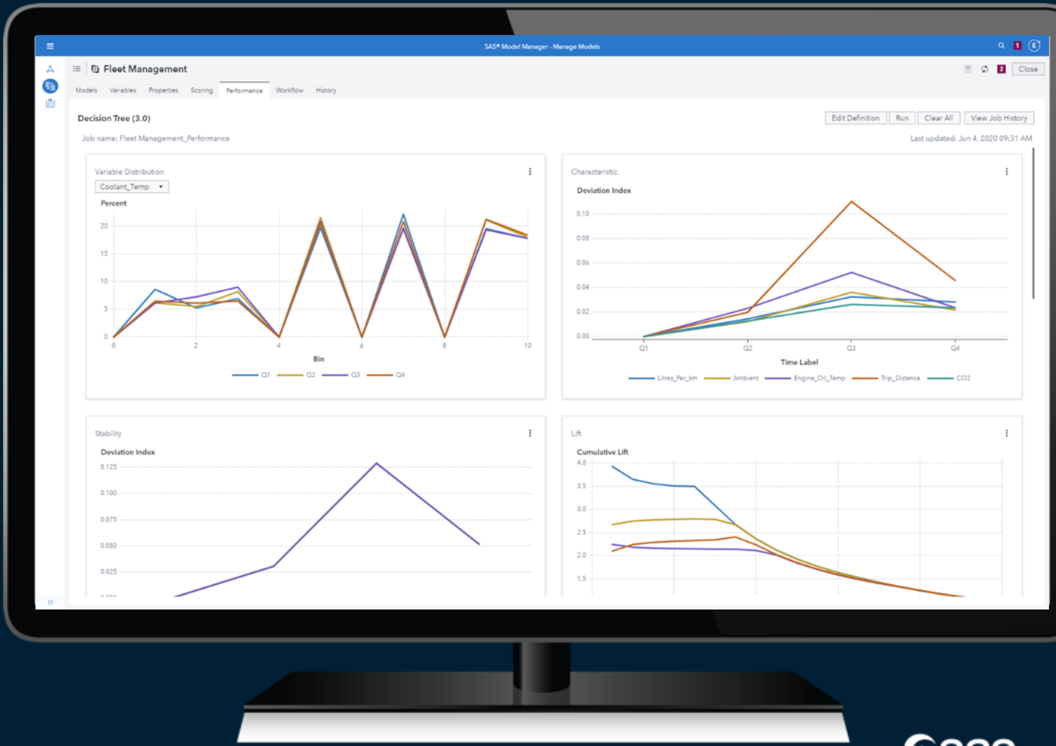


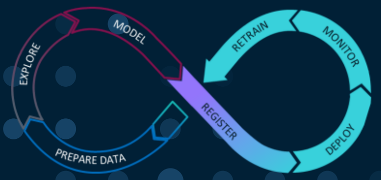


Monitor

Monitor Model Drift and Performance

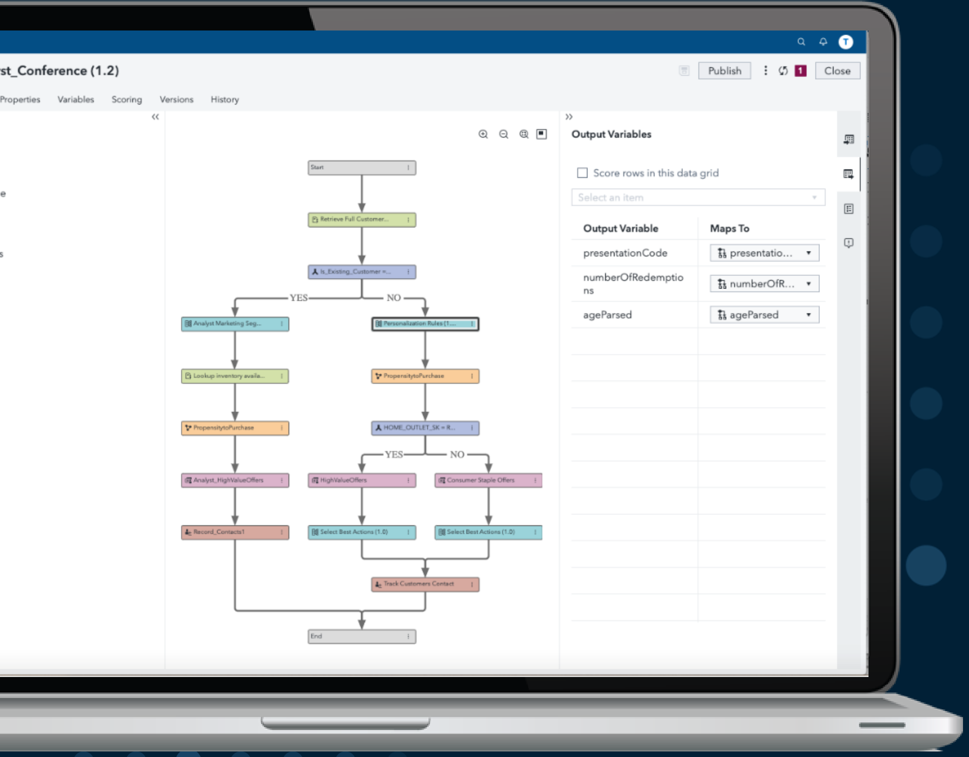
- Wizard to simply generation of out of the box performance reports
- Performance-monitoring tasks including data source changes, score value changes, model accuracy over multiple time periods, and input/output variable feature contribution plot
- Users can access the data to generate their own reports

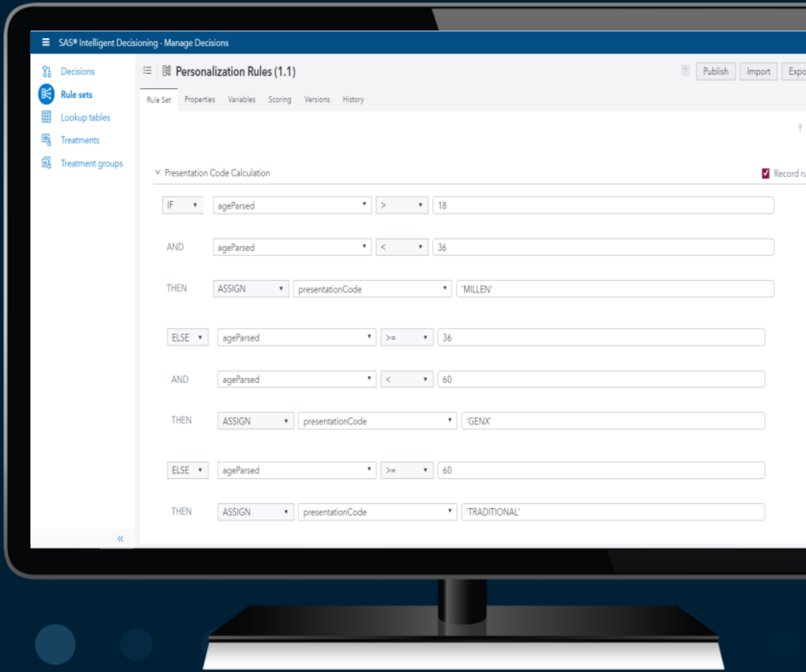




Designing Decisions

Decision Flow



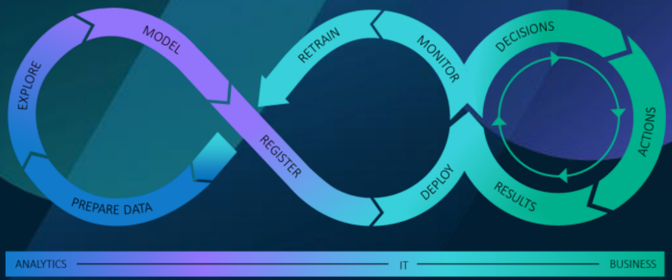


Business Rules Management

Modeling gas turbine measurements to reduce emissions

SAS Viya demo steps

- Exploration
- Modeling
- Model tuning and registration (Model Studio & Open source)
- Model deployment (single OS model & decision flow)



Case description

CASE DESCRIPTION

The dataset contains 11 sensor measures aggregated over one hour (by means of average or sum) from a gas turbine power plant for the purpose of studying flue gas emissions, namely CO and NOx. CO emission is removed from inputs as it may not be available at prediction time.

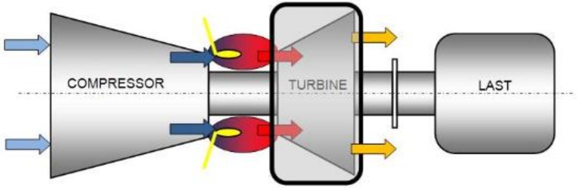
Nitrogen oxides are produced in combustion processes, partly from nitrogen compounds in the fuel, but mostly by direct combination of atmospheric oxygen and nitrogen in flames.

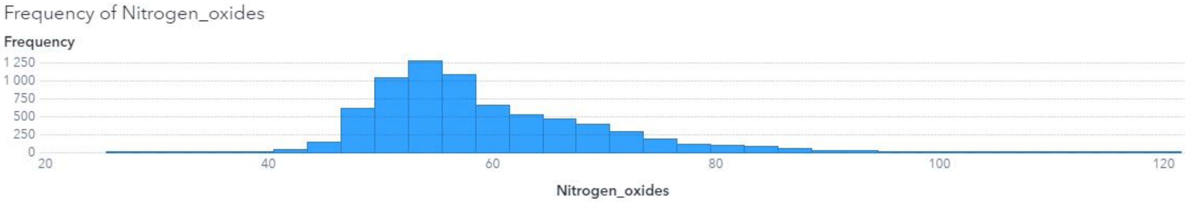
Elevated levels of nitrogen dioxide can cause damage to the human respiratory tract and increase a person's vulnerability to, and the severity of, respiratory infections and asthma. High levels of nitrogen dioxide are also harmful to vegetation—damaging foliage, decreasing growth or reducing crop yields.

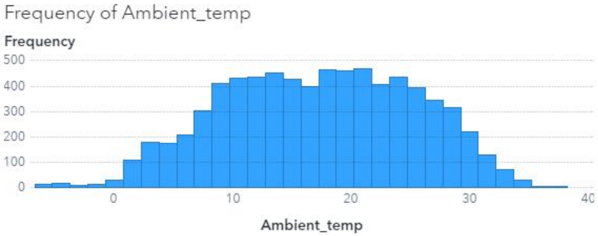
Source: http://archive.ics.uci.edu/ml/datasets/Gas+Turbine+CO+and+NOx+Emission+Data+Set

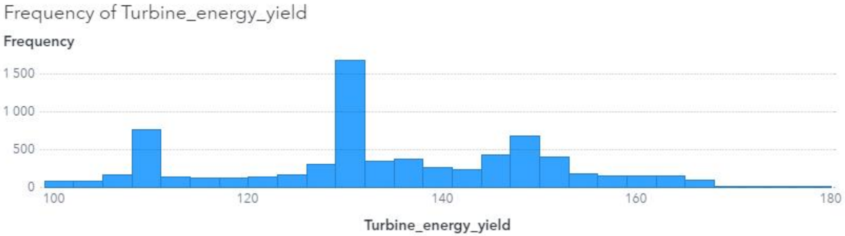
OBJECTIVE

Model Nitrogen oxide (NOx) emissions to understand when they are highest and can we reduce them using that information









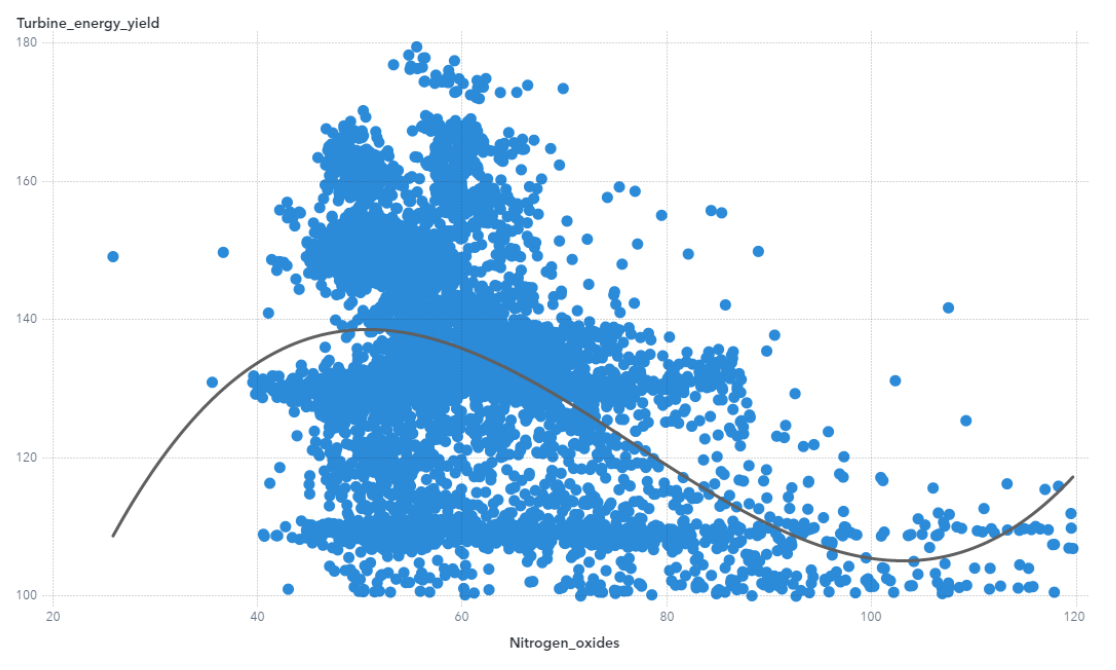
Frequency

7,4 t.

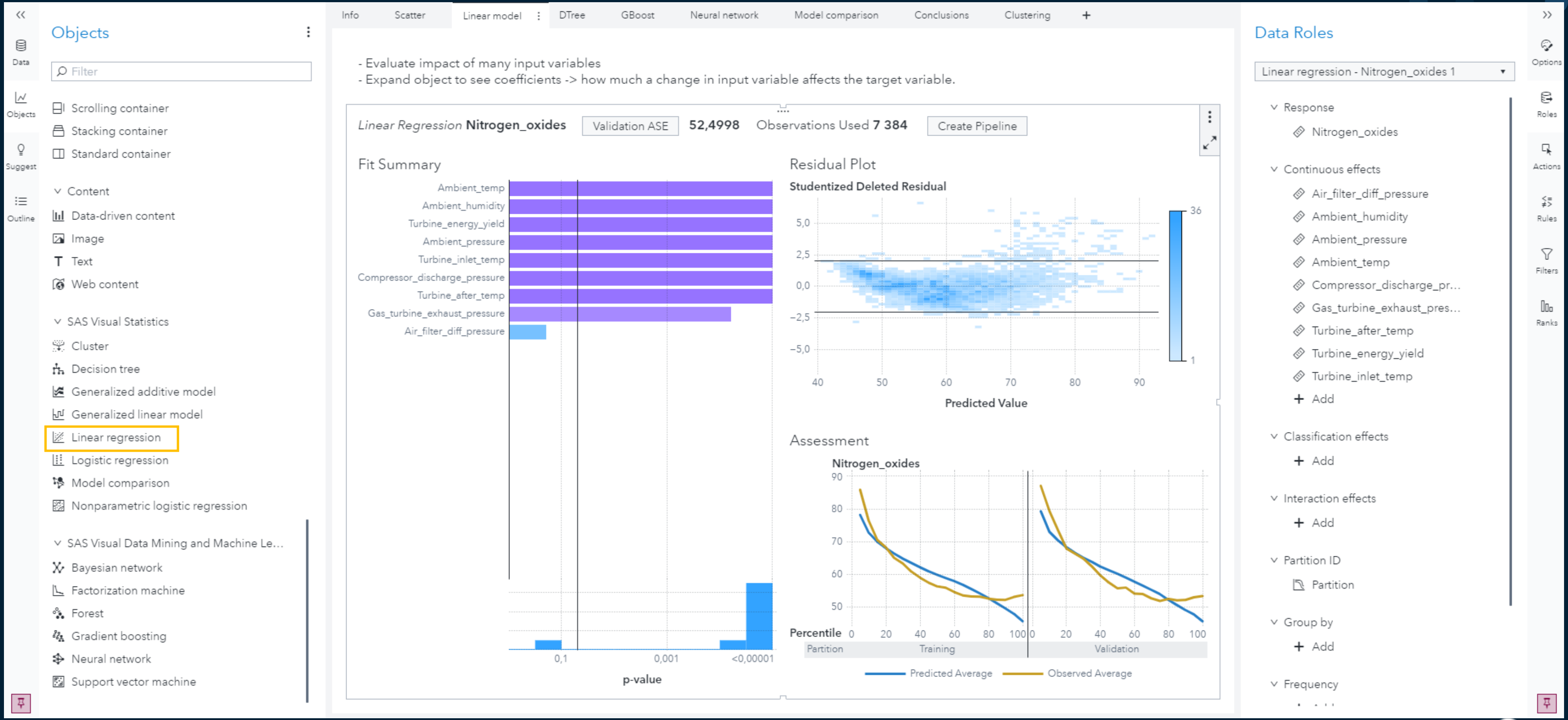
Understanding the data by modeling simple relationships

- Using fit lines on scatter plot we can see that we are able to achieve high energy yields also with lower NOx levels

- With this method we are only able to study relationship between two variables



Linear models with multiple inputs can be built just as easily

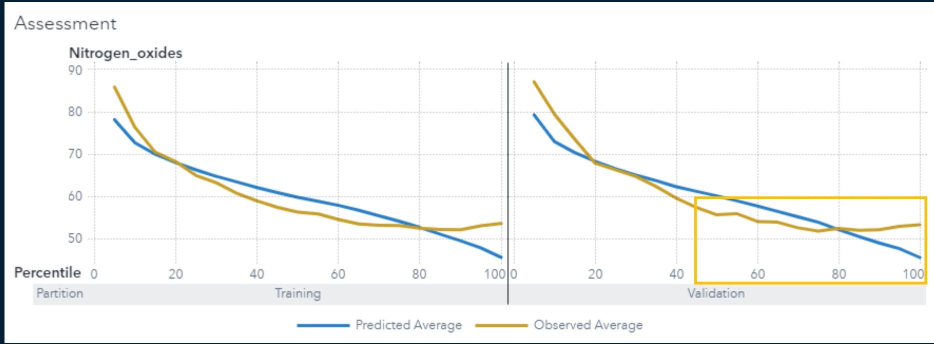


Parameter estimates and assessment statistics tell us that linear model might not be the best choice

- Parameter estimates show how each input affects the target (NOx level)
- The assessment plots indicate that the model is still far from perfect. Probably because of non-linear relationships we are not taking into account
- Explained variance is 58%
- Let’s try other algorithms

Dimensions | Overall ANOVA | Fit Statistics | Parameter Estimates | Type III Test | Assessment | Assessment Statistics

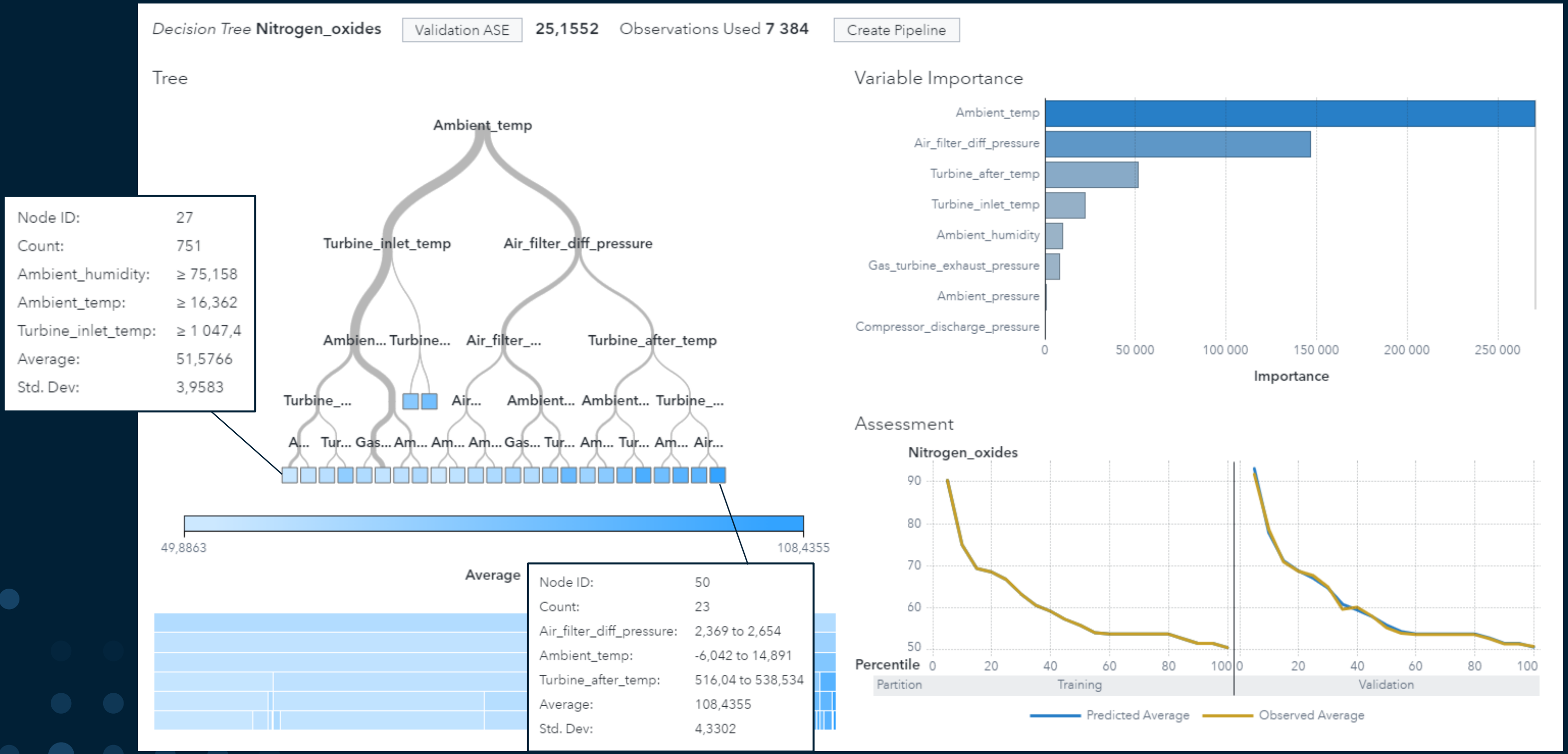
Parameter	Estimate	Standard Error	t Value
Compressor_discharge_pressure	12,28666	2,094132	5,867187
Turbine_after_temp	−0,95727	0,181769	−5,26638
Gas_turbine_exhaust_pressure	0,290256	0,072239	4,018009
Air_filter_diff_pressure	0,951382	0,727478	1,307781



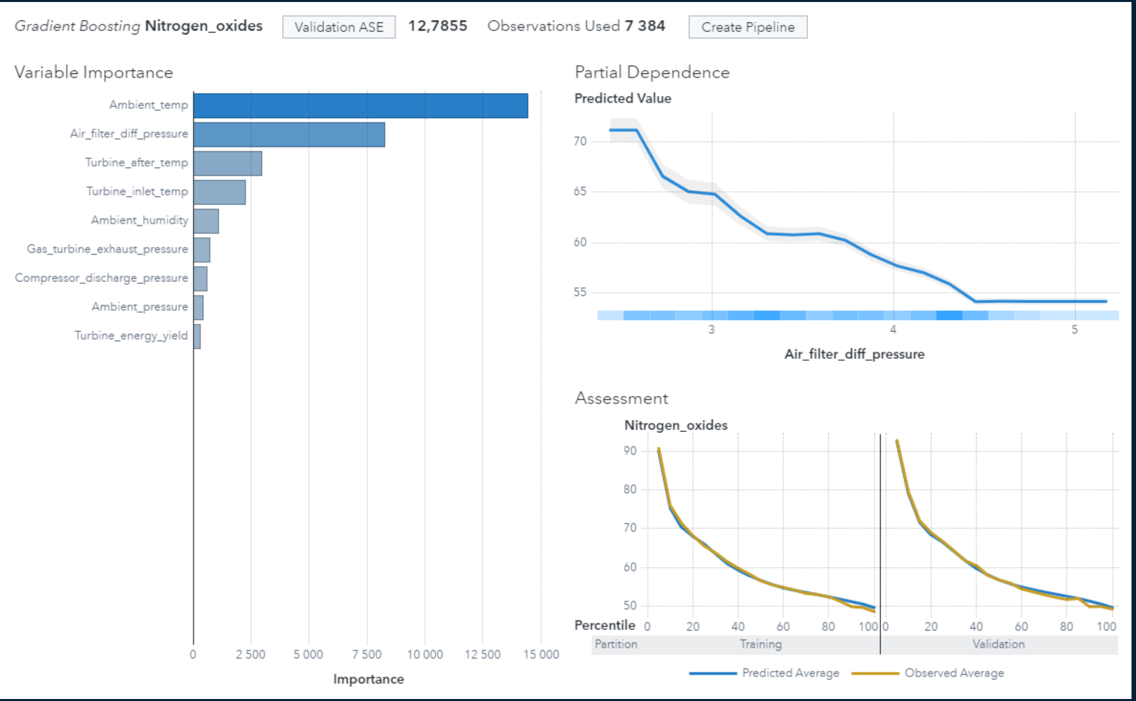
Dimensions | Overall ANOVA | Fit Statistics | Parameter Estimates | Type III Test | Assessment | Assessment Statistics

Source	Deg Freedom	Sum of Squares	Mean Square	F Value	Pr > F	R-Square
Model	9	366875,5	40763,94	811,1802	<0,00001	0,585942
Error	5159	259253,3	50,25263	.	.	.

Decision tree is more accurate & we learn what kind of situations produce high/low NOx levels and should be avoided/favored



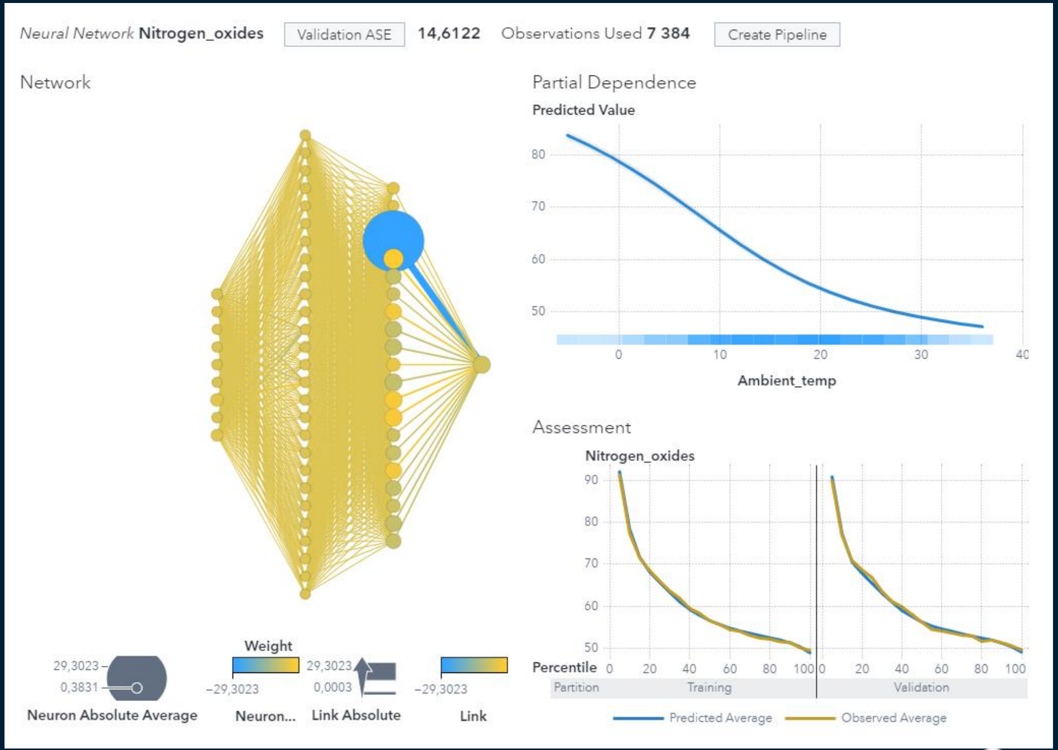
Increasing accuracy at the cost of explainability



- Gradient boosting algorithm (multiple decision trees) cuts the error in half
- Model interpretability methods are needed to understand the predictions
- Partial dependence describes the average effect an input has on the target
- This information can be used to find parameter ranges that result in acceptable NOx levels

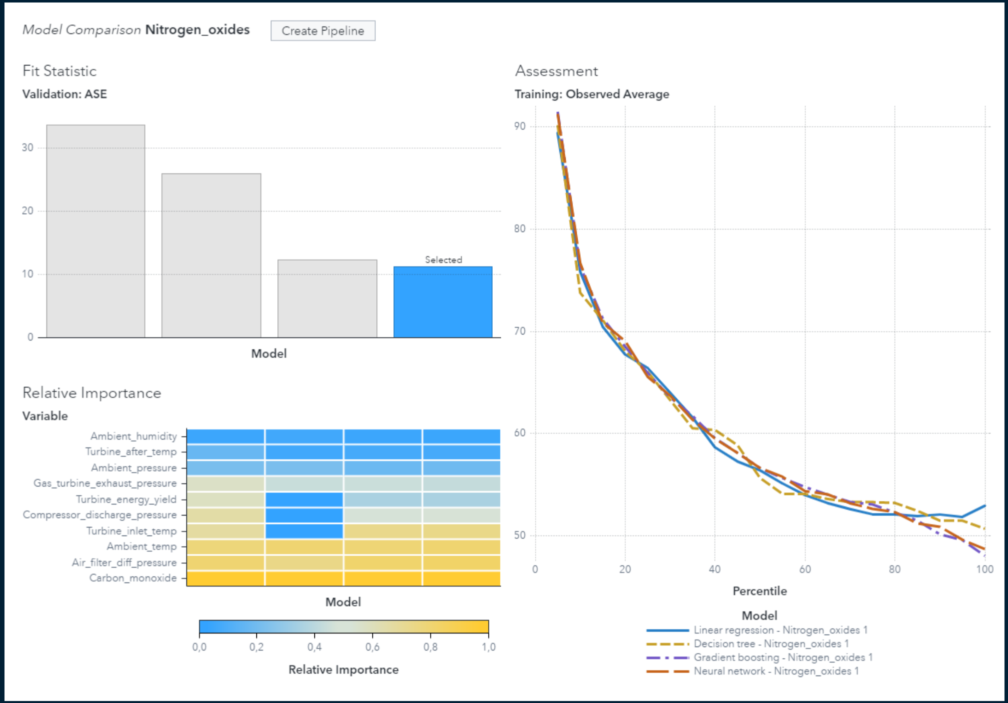
Autotuning can help get the final improvements to accuracy

- Neural networks typically need tuning to get best possible performance
- You can use autotuning to find better hyperparameters for your model
- Neural network slightly outperforms gradient boosting in this case



Model comparison

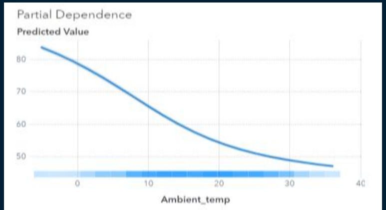
- Model comparison object shows which of the models was best in terms of ASE (averaged squared error)
- It also shows which variables were most important across models



What we have found out so far

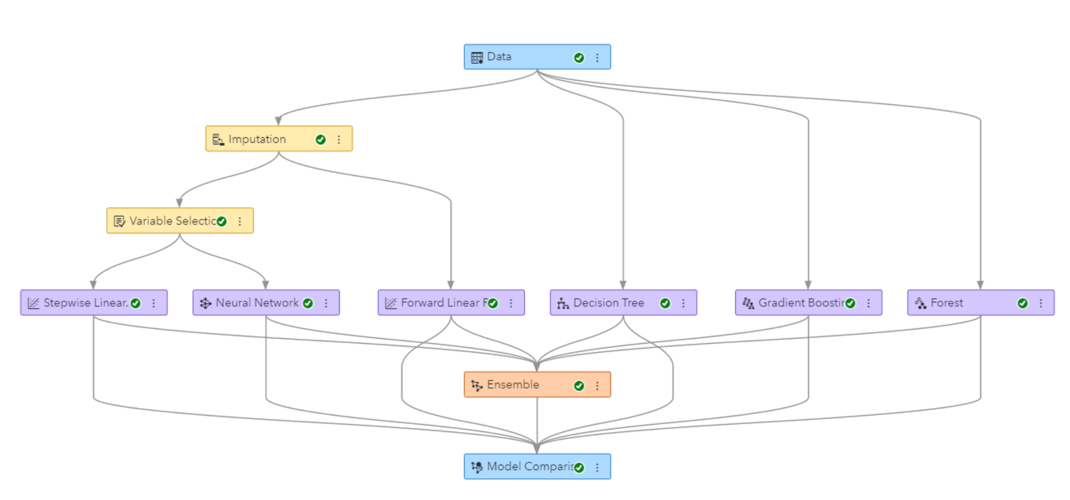
- The relationships in the data may be non-linear
- Decision trees showed how there are certain situations when NOx emissions are low
  - Those settings should be favored when running the turbines
- Partial dependence showed for example how ambient temperature and Air filter difference pressure affect the emissions
  - The information can be used do decide how and when to run the turbines and where they should be located

Node ID:	27
Count:	751
Ambient_humidity:	≥ 75,158
Ambient_temp:	≥ 16,362
Turbine_inlet_temp:	≥ 1 047,4
Average:	51,5766
Std. Dev:	3,9583

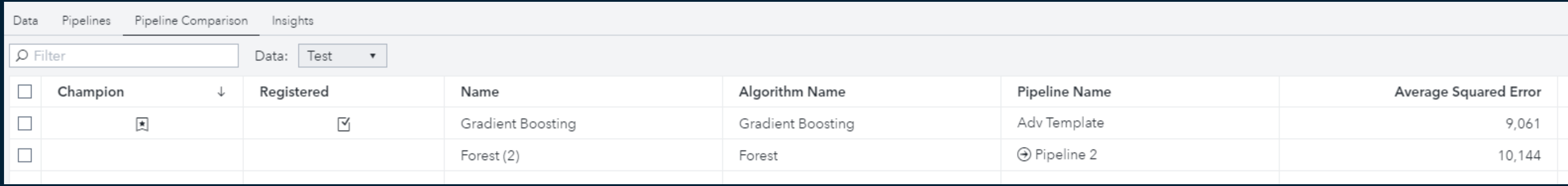


Tuning the accuracy further to get the best possible model

- Pipeline view in Model Studio is aimed for data scientists
- There different modeling strategies can be attempted
  - Imputation
  - Variable selection
  - Transformations
  - PCA etc.
- Different algorithms and modeling strategies can be automatically competed to find the best combination

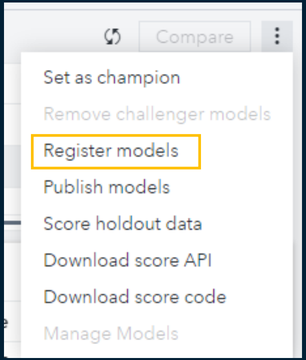


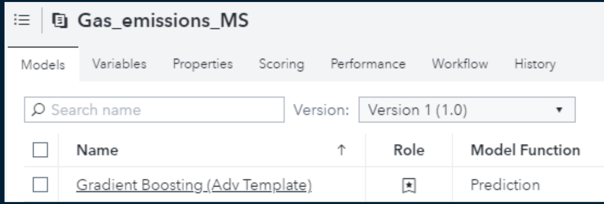
Registering the best possible model to Model Manager



	Champion	Registered	Name	Algorithm Name	Pipeline Name	Average Squared Error
			Gradient Boosting	Gradient Boosting	Adv Template	9,061
			Forest (2)	Forest	Pipeline 2	10,144

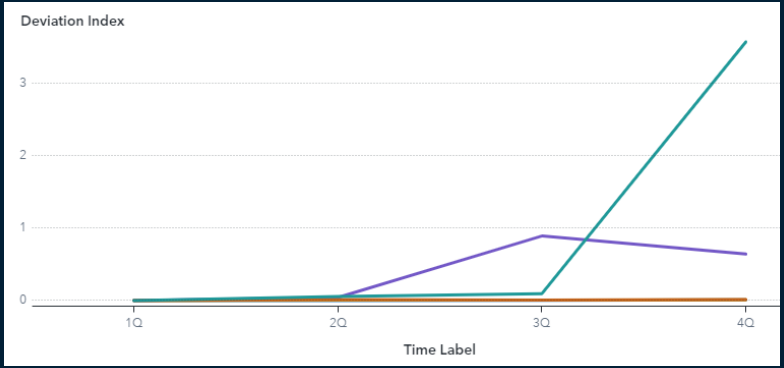
- After running the pipelines with autotuning, the best possible model turned out to be Gradient boosting with ASE of 9,061
- We should now register the model centralized Model Manager
- It was important to get the most accurate model as next we will publish the model to run simulations on situations that produce lowest emissions

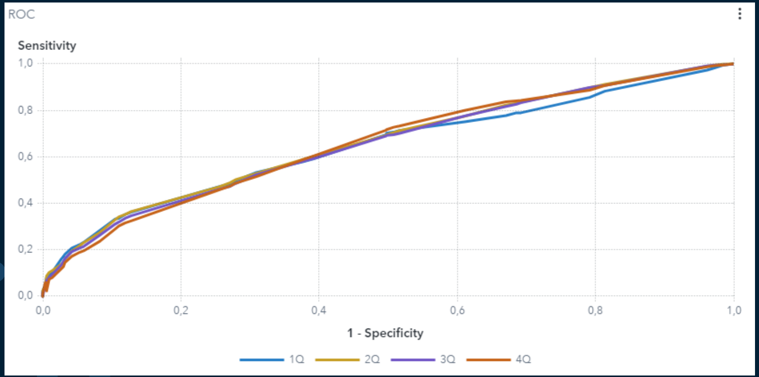




Model Manager helps with monitoring performance of the models that are in production

- Variable distribution shows show there has been a significant change in 4th quarter with one of the inputs





- Different accuracy measurements can be used to track the model performance in production
- When performance deteriorates too low, the model needs to be retrained

Open-source models can be trained with python and registered to the same centralized Model Manager

```
In [17]:  import pandas as pd
          import numpy as np
          import os
          from sasctl import Session, register_model, publish_model
          import xgboost as xg
          from sklearn.model_selection import train_test_split
          os.environ['CAS_CLIENT_SSL_CA_LIST'] = '/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.pem'

In [18]:  df = pd.read_csv('GAS_EM_SMPL.csv')

In [19]:  df2 = df.iloc[:1000,:]
          X = df2.iloc[:,:-1]
          y = df2.iloc[:,-1]

In [20]:  X.head()
Out[20]:
```

	Ambient_temp	Ambient_pressure	Ambient_humidity	Air_filter_diff_pressure	Gas_turbine_exhaust_pressure	Turbine_inlet_temp	Turbine_after_temp	Turl
0	1.95320	1020.1	84.985	2.5304	20.116	1048.7	544.92	
1	1.21910	1020.1	87.523	2.3937	18.584	1045.5	548.50	
2	0.94915	1022.2	78.335	2.7789	22.264	1068.8	549.95	
3	1.00750	1021.7	76.942	2.8170	23.358	1075.2	549.63	
4	1.28580	1021.6	76.732	2.8377	23.483	1076.2	549.68	

```
In [21]:  y.head()
Out[21]:  0    113.250
          1    112.020
          2     88.147
          3     87.078
          4     82.515
          Name: Nitrogen_oxides, dtype: float64

In [22]:  xTrain, xTest, yTrain, yTest = train_test_split(X, y, test_size=0.3, random_state=42)

In [24]:  xgb_r = xg.XGBRegressor(objective ='reg:squarederror',
                           n_estimators = 10, seed = 123)

In [25]:  xgb_r.fit(xTrain, yTrain)
```

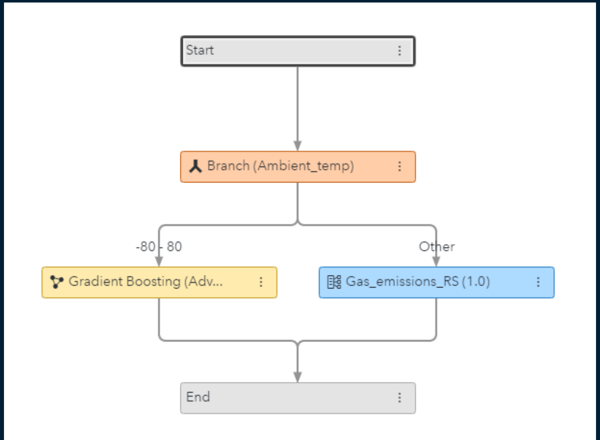
- XGBoost model is trained in jupyter with the same data
- Data can be also pulled for Viya's in-memory tables
- The model can be registered to Model Manager
- The model can be easily deployed

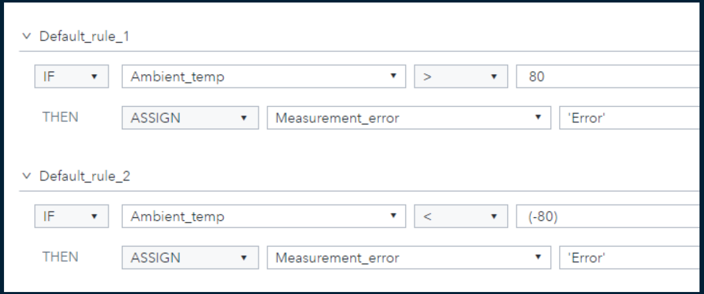
```
In [27]:  with Session('localhost', 'user', 'pw'):
              model_name = 'GE_XGB'
              project_name = 'Gas_emission_XGB'

              # Register the model in SAS Model Manager
              register_model(xgb_r, model_name, project_name, input=xTrain, force=True)
```

Models usually need logic around them before they can be used in production processes

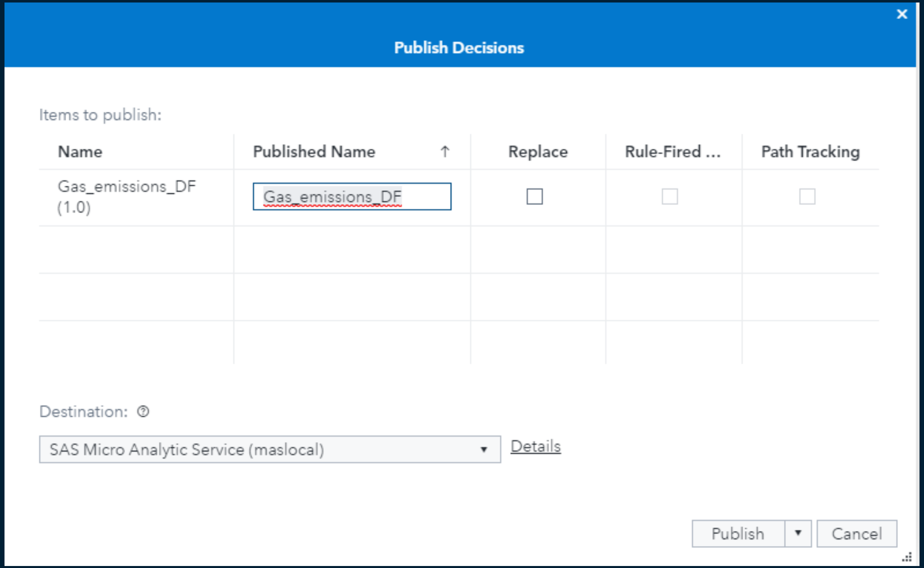
- Decision flows are a great way to add surrounding business logic to a model
- It is much more manageable than hard coding that logic somewhere
- You can add
  - Data base / API calls for more data
  - Business rules / data quality checks
  - Multiple models
  - Decision branches etc.



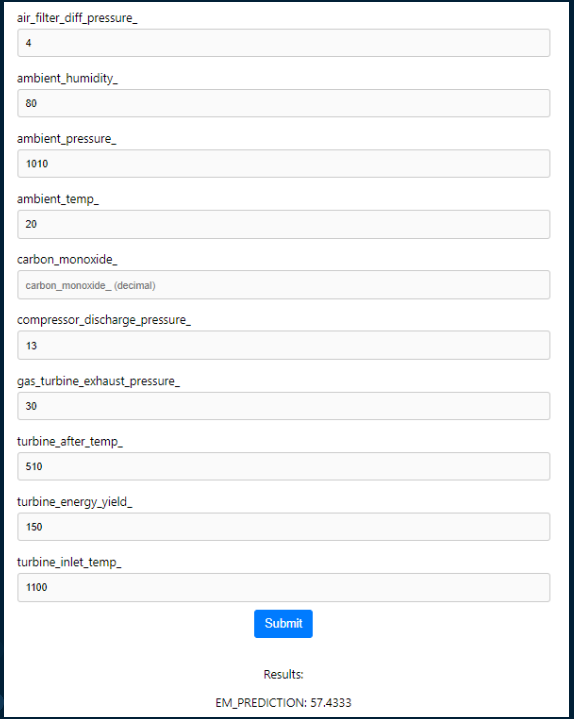


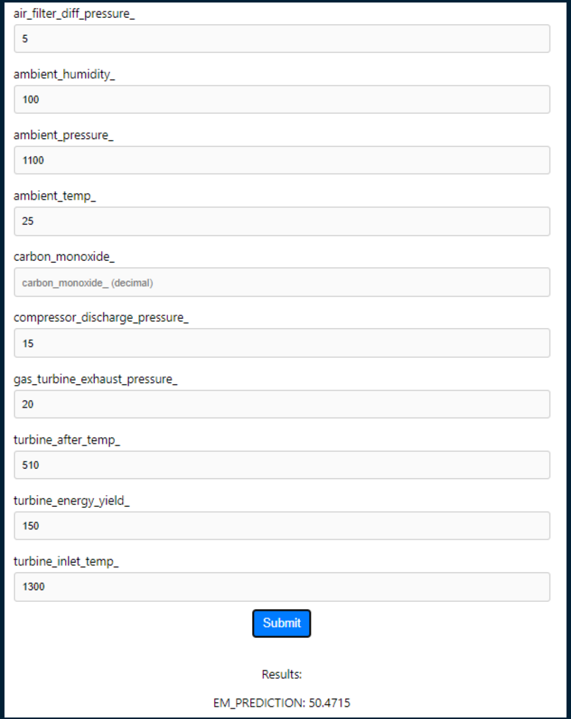
Publishing models or decisions to production

- Models and decision flows can be published on the real-time engine MAS or the batch engine CAS
- After publishing the model is ready to return predictions based on inputs
- Other publishing destinations can be configured, such as containers



Sending scoring requests to the model via REST to simulate different scenarios and the resulting NOx emission


air_filter_diff_pressure_
4
ambient_humidity_
80
ambient_pressure_
1010
ambient_temp_
20
carbon_monoxide_
carbon_monoxide_ (decimal)
compressor_discharge_pressure_
13
gas_turbine_exhaust_pressure_
30
turbine_after_temp_
510
turbine_energy_yield_
150
turbine_inlet_temp_
1100
Submit
Results:
EM_PREDICTION: 57.4333


air_filter_diff_pressure_
5
ambient_humidity_
100
ambient_pressure_
1100
ambient_temp_
25
carbon_monoxide_
carbon_monoxide_ (decimal)
compressor_discharge_pressure_
15
gas_turbine_exhaust_pressure_
20
turbine_after_temp_
510
turbine_energy_yield_
150
turbine_inlet_temp_
1300
Submit
Results:
EM_PREDICTION: 50.4715

Resources

- Data set
  - http://archive.ics.uci.edu/ml/datasets/Gas+Turbine+CO+and+NOx+Emission+Data+Set
- SAS VDMML
  - https://www.sas.com/en_us/software/visual-data-mining-machine-learning.html
- ModelOps
  - https://www.sas.com/en_us/solutions/operationalizing-analytics/modelops-approach.html