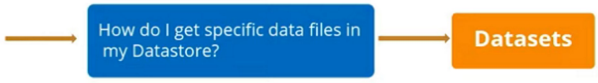
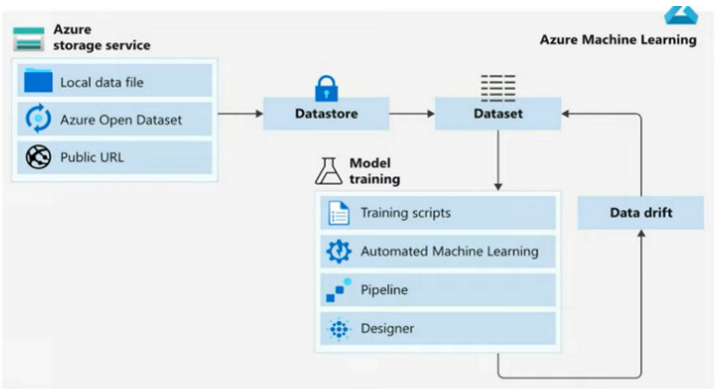
Data Prep / Datasets

Tuesday, July 14, 2020 1:45 PM

- Model training: process to convert data into trained models
- Data preparation needs to happen before a model can be trained, data is key to preventing error, bias, classification etc.

- Data importing and transformation
  - Data import is very important because most ML algorithms depend on good clean data in order to properly and accurately train high quality models
  - Data Wrangling
    - Process of cleaning and transforming data to make it more appropriate for data analysis
      - Explore raw data and check general quality of the dataset
        - Understanding the data beyond data types and simple descriptions
      - Transform the raw data be restructuring, normalizing and cleaning the data
        - Can include handling missing values and detecting errors
      - Validate and publish the data
- The data management process
  - The use of *datastores* and *datasets*
    - Datastores
      - Layer of abstraction that provides isolation from various data storages in Azure
      - Connection information is hidden to provide more security
      - Stores all the information needed to connect to a particular storage service
      - Provides an access mechanism that is independent of the computer resource that is used to drive a machine learning process
      - Compute location independence
        - Provides an access mechanism that is independent of compute resource used to drive a ML process, can be shared and accessed simultaneously by different instances of ML processes





- Datasets
  - Resources for exploring, transforming, and managing data in Azure ML
  - A reference that points to the data in storage, used to get specific data files in the datastores
  - Enters specific data files and specific data into datastores
  - Points to specific training, validation, test data used in ML processes
  - Can accomplish ML tasks
    - Training tasks
    - Consummation of datastores in automated processes
    - Access to datastores through inference processes
    - Data labeling
    - Monitoring datasets for issues
  - Used to interact with your data in the datastore and to package data into consumable objects.
  - Can be created from local files, public URLs, Azure Open Datasets, and files uploaded to the datastores
  - Not copies of the data but references that point to the original data. This means that no extra storage cost is incurred when you create a new dataset
  - Once a dataset is registered in Azure ML workspace, you can share it and reuse it across various other experiments without data ingestion complexities
  - Can have a single copy of some data in your storage, but reference it multiple times—so that you don't need to create multiple copies each time you need that data available
  - Can access data during model training without specifying connection strings or data paths
  - Can easily share data and collaborate with other users
  - Can bookmark the state of your data by using dataset versioning

- Data Access Workflow
  - Create a datastore so you can access data storage services in Azure
  - Create a dataset which you will use for model training in your ML experiment
    - Can mount into experiments compute target
    - Consume directly with automated ML processes or training scripts or designer
  - Create a dataset monitor to detect issues in the data such as *data drift*
    - Data drift: the input data you are feeding into the model is likely to change
      - Creates issues with model accuracy
      - Ex. If you train a model to detect spam emails, can become less accurate over time as new types of spam arise that weren't present in training set
    - Dataset monitors: detects data drift an can have system automatically update the input dataset so you can retrain the model and maintain accuracy

- Versioning: bookmarking the state of your data
  - Used when new data is available for retraining
  - Used when you are applying different approaches to data preparation or feature engineering
  - Version control for data
  - Two datasets types supported in Azure ML
    - Tabular dataset
      - Represents data in a tabular format created by parsing the provided file or list of files
    - Web URL (file dataset)
      - Set of references to single or multiple files in the datastores
  - Key step of end-to-end process that provides support for full traceability
    - Can see full traceback via versioning, provides snapshots at each step
  - **Versions are only references to the original file at a certain point in time, still need to make sure original file is not modified or delete in order to preserve the dataset and dataset versions**

- Feature engineering
- Data drift

Lab: Data Wrangling

Tuesday, July 14, 2020 8:50 PM

Lab Overview

In this lab you learn how to import your own data in the designer to create custom solutions. There are two ways you can import data into the designer in Azure Machine Learning Studio:

- Azure Machine Learning datasets

  Register datasets in Azure Machine Learning to enable advanced features that help you manage your data.

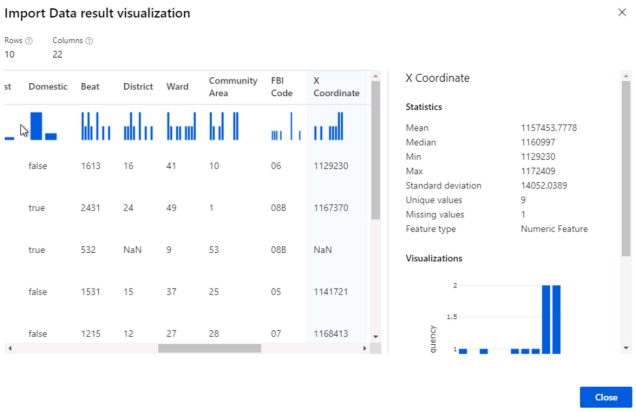
- Import Data module

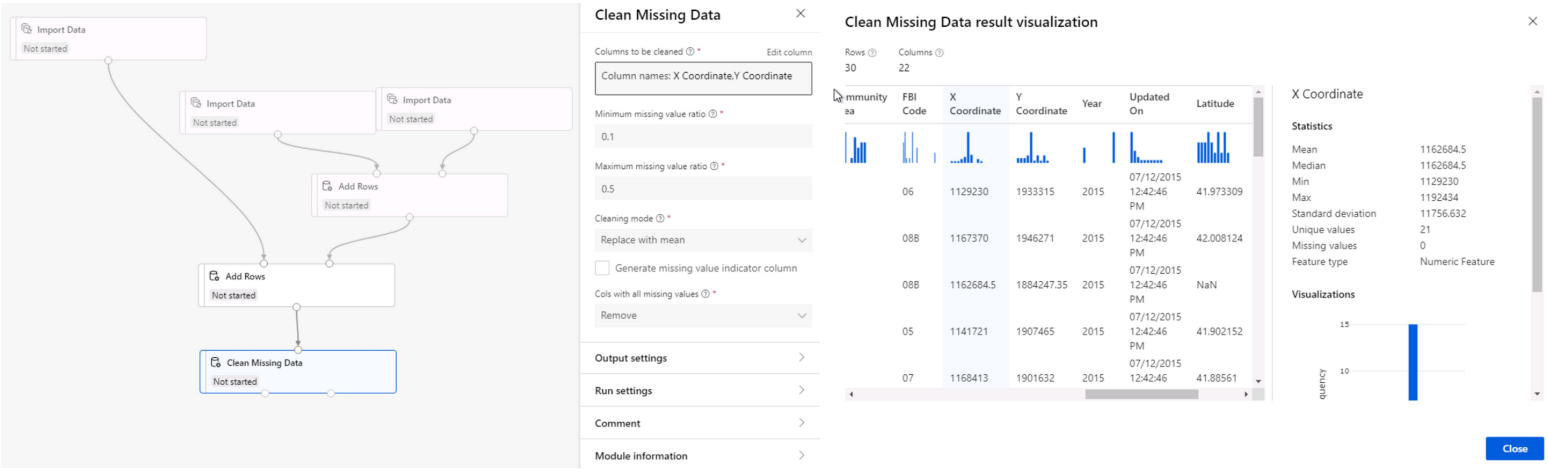
  Use the Import Data module to directly access data from online datasources.

The first approach will be covered later in the next lab, which focuses on registering and versioning a dataset in Azure Machine Learning studio.

While the use of datasets is recommended to import data, you can also use the Import Data module from the designer. Data comes into the designer from either a `Datastore` or from `Tabular Datasets`. Datastores will be covered later in this course, but just for a quick definition, you can use Datastores to access your storage without having to hard code connection information in your scripts. As for the second option, the Tabular datasets, the following datasources are supported in the designer: Delimited files, JSON files, Parquet files or SQL queries.

The following exercise focuses on the Import Data module to load data into a machine learning pipeline from several datasets that will be merged and restructured. We will be using some sample data from the UCI dataset repository to demonstrate how you can perform basic data import transformation steps with the modules available in Azure Machine Learning designer.





Lab: Create/Version Dataset

Tuesday, July 14, 2020 10:11 PM

Name	Version
nyc-taxi-sample-dataset	2

nyc-taxi-sample-dataset Version 2 (latest)

- Version 2 (latest)
- Version 1

Details Consume Explore

Refresh | Generate profile | Unregister | New version

Attributes

Properties
Tabular

Description

Created by
ODL_User 15037

Web Url
https://introtomlsampledata.blob.core.windows.net/data/nyc-taxi/nyc-taxi-sample-data-6months.csv

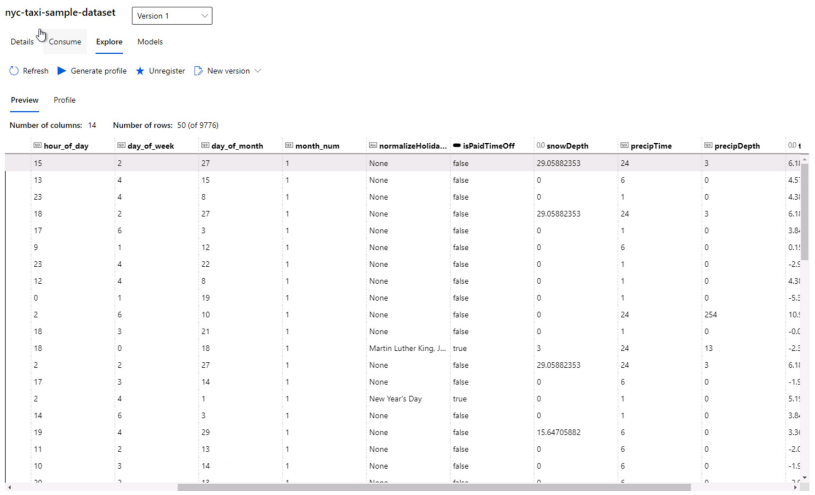
Profile
No profile generated

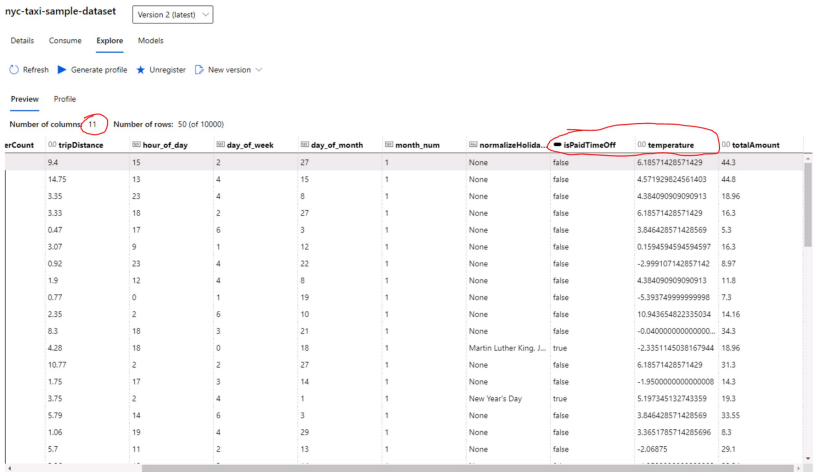
Current version
2

Latest version
2

Created time
Jul 15, 2020 5:28 AM

Modified time
Jul 15, 2020 5:29 AM





Feature Engineering / Selection / Data Drift

Tuesday, July 14, 2020 10:11 PM

- Features
  - Columns in a table
  - Need to pick which features are best appropriate in describing the table and to be used in the algorithm you are trying to train
  - Feature Engineering - given a dataset, create new features
    - Data preparation
    - Can be used to derive new features based on the values of existing features
    - Can apply a mathematical function to a feature or training a separate ML model to create value for new features
    - Select the form you want your new data/ feature in (video, numerical, photo etc)
  - Feature Selection
    - Selecting the features that are the most important or relevant
  - Dimensionality Reduction
    - Many ML algorithms cannot accommodate large numbers of features so # of features needs to be decreased
- Feature Engineering
  - One of core techniques used to increase chances of success in solving ML problems
  - Used to help increase the power of ML algorithms
  - Uses existing features / data to derive new features / values to better train a model
    - Ex. Customer behavior dataset
      - Derive features from "purchases a customer has made"
        - # of purchases in the last month
        - # of purchases in last 3 months
        - Freq of purchases over time
  - Helps derive features that are better suited for types of feature engineering
  - Lays foundation for good models with good capabilities in prediction and stability
  - Not always necessary
  - Various places where feature engineering can be implemented
    - Data source
      - Ex. Database -> can calculate and create new columns via database engine
    - Coding Environments
      - Python env
        - Can use libraries and built in functionality
      - Streaming env
        - Spark supports parallel data processing, useful for large datastreams
    - Training Model Process
      - Not uncommon to derive new features as part of training process
  - Classical ML is more dependent on feature engineering than Deep Learning
    - FE is an explicit process in Classical
    - In DL, FE occurs naturally in the neural network so its sometimes deferred in the training process
  - Deep learning can be used for implementation of certain FE processes like embedding

- Feature Engineering Tasks
  - Aggregation
    - Ex. Sum, count, median, mean
  - Part-of
    - Extract part of a certain data structure
    - Ex. Part of a date
    - Ex. Part of a year date
  - Binning
    - Group entities into bins then apply aggregation techniques over those bins
    - Ex. bin based on category then calculate avg purchases by age
  - Flagging
    - Deriving boolean conditions expressed through boolean values
    - Ex. does customer have purchase in last 6 months
  - Frequency based
    - Calculate various freq of occurrence for certain amts of data
  - Embedding (Feature Learning)
  - Deriving by Example
    - Aim to learn values of new features using examples of existing features

- Data types
  - Numbers, text, image
  - FE process applies differently based on type of data
  - Numerical data is usually tabular
    - Subject to classical FE tasks
  - Text not suitable for numerical processing
    - Needs to translate text into numerical format
    - Text embedding is process in which sequences of words / natural language is transformed into numerical format (usually numerical vectors)
      - Text frequency, inverse frequency, word embedding
  - Image has same problem as text, RBG format is not suited for ML algorithms, needs translation
    - Image is a matrix of tuples of three values
    - Ex. 500x400 px image, each pixel contains the three RGB values
    - Numerical representation is the 60,000 values (500x400x3) in a multidimensional matrix
    - FE not required before training process for images when they are fed into neural networks because the process happens naturally in hidden layers of the neural networks
      - some more complex neural networks have layers dedicated to deciphering data and discovering patterns

- Feature Selection - figures out which features are most useful for a given model
  - Number of features resulting from FE can be huge, which are the features most useful for a given model?
  - Features can be numerous even without FE
  - Reasons to use FS
    - Elimination of irrelevant, redundant, or highly correlated features
      - Highly correlated features provide no addition value to algorithm, fine to pick one and discard the other
    - Reduce dimensionality to improve performance of model training process
      - Large number of ML algorithms that do not cope well with large number of dimensions / features
        - Curse of dimensionality
      - Dimensionality Reduction
        - Referred to being part of the larger family of feature engineering techniques AND also feature learning techniques
        - Algorithms
          - PCA (principle component analysis)
            - Linear technique based on exact mathematical calculation
            - Statistics approach
          - t-SNE (T-distributed stochastic neighboring entities)
            - Probability approach to reduce dimensionality
            - Used a lot for visualizing your data, reduces down to 2-3 dimensions usually
            - Keeps close points from multidimensional points close in the 2-D space as well
          - Feature Embedding
            - Based on training a separate ML model to encode a large number of features into smaller numbers of features
        - Azure Pre-built Modules
          - Filter based feature selection
            - Identify columns in input dataset that have the most predictive power
          - Permutation Feature Importance
            - Determines best features to use by computing a set of feature importance scores

- Data Drift
  - Change in the model input data that leads to degradation of model performance
    - Input data changes and becomes more and more different and the model is essentially trained on outdated and inaccurate data
    - Monitoring helps detect these issues and trigger retraining processes to prevent it
  - Causes
    - Changes in upstream process
      - Ex. Sensor supplying data is replaced or updated so data that is being sent is different
    - Data Quality
      - Ex. Broken sensor gives bad data
    - Natural Data Drift
      - Ex. Customer behavior changing because their interests / habits change
    - Changes in relationships between features / Covariate shift
      - Ex. Previously correlated features can have more or less correlation as time passes
  - Dataset monitors can be set up in Azure to provide alerts in detecting data drift in datasets over time
    - **Detection monitoring and detecting data drift is a key component to creating models that keep performance and accuracy over time**
    - Early detection of data drift indicators is critical in maintaining models over time
    - Model training process does not end once you train your first successful model
      - Need to constantly monitor the performance of this model
    - Works by analyzing historical versions of data and enables profiling of new data over time
    - Data drift algorithm provides overall measure of change that occurs in your data
    - Monitored via datasets in Azure
      - Baseline dataset (dataset to train model originally) is specified
      - Target datasets (input data for the model) are compared to baseline datasets to detect data drift
    - Scenarios to set up dataset monitoring in Azure ML
      - Monitoring to detect data drift between input dat and the previous training data
        - Proxy for model accuracy, increase in difference = decrease in accuracy
      - Monitoring a time series dataset for drift from a previous time period
        - Seasonal drift can be detected and used to alert to more potential issues with model accuracy
      - Perform analysis on past data
        - Better understand dynamics of data to improve model
  - Data Drift results produced in charts and found in Drift Overview results
    - 0% magnitude indicates 0 drift, 100% means you basically have new data

Lab: Engineer and Select Features

Thursday, July 16, 2020 1:42 PM

Engineer and select features

Lab Overview

This lab demonstrates the feature engineering process for building a regression model using bike rental demand prediction as an example. In machine learning predictions, effective feature engineering will lead to a more accurate model. We will use the Bike Rental UCI dataset as the inpu raw data for this experiment. This dataset is based on real data from the Capital Bikeshare compan which operates a bike rental network in Washington DC in the United States. The dataset contains 17,379 rows and 17 columns, each row representing the number of bike rentals within a specific ho of a day in the years 2011 or 2012. Weather conditions (such as temperature, humidity, and wind speed) were included in this raw feature set, and the dates were categorized as holiday vs. weekday etc.

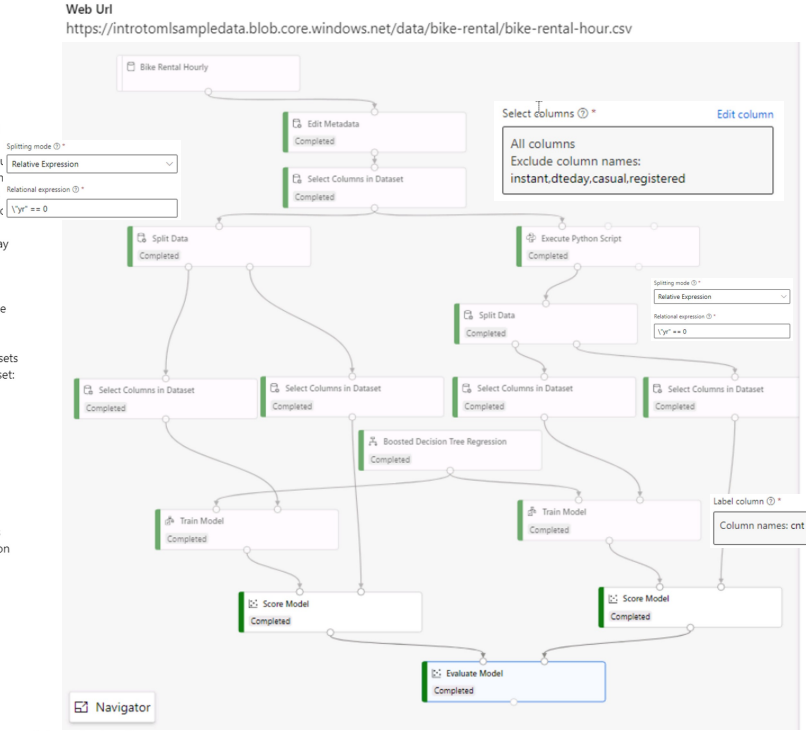
The field to predict is `cnt` which contains a count value ranging from 1 to 977, representing the number of bike rentals within a specific hour. Our main goal is to construct effective features in the training data, so we build two models using the same algorithm, but with two different datasets. Using the Split Data module in the visual designer, we split the input data in such a way that the training data contains records for the year 2011, and the testing data, records for 2012. Both datasets have the same raw data at the origin, but we added different additional features to each training set:

- Set A = weather + holiday + weekday + weekend features for the predicted day
- Set B = number of bikes that were rented in each of the previous 12 hours

We are building two training datasets by combining the feature set as follows:

- Training set 1: feature set A only
- Training set 2: feature sets A+B

For the model, we are using regression because the number of rentals (the label column) contains continuos real numbers. As the algorithm for the experiment, we will be using the Boosted Decision Tree Regression.




```
# The script MUST contain a function named azureml_main
# which is the entry point for this module.

# imports up here can be used to
import pandas as pd
import numpy as np

# The entry point function can contain up to two input arguments:
#   Param<dataframe1>: a pandas.DataFrame
#   Param<dataframe2>: a pandas.DataFrame
def azureml_main(dataframe1 = None, dataframe2 = None):

    # Execution logic goes here
    print(f'Input pandas.DataFrame #1: {dataframe1}')

    # If a zip file is connected to the third input port,
    # it is unzipped under "./Script Bundle". This directory is added
    # to sys.path. Therefore, if your zip file contains a Python file
    # mymodule.py you can import it using:
    # import mymodule

    for i in np.arange(1, 13):
        prev_col_name = 'cnt' if i == 1 else 'Rentals in hour -{}'.format(i-1)
        new_col_name = 'Rentals in hour -{}'.format(i)

        dataframe1[new_col_name] = dataframe1[prev_col_name].shift(1).fillna(0)

    # Return value must be of a sequence of pandas.DataFrame
    # E.g.
    #   - Single return value: return dataframe1,
    #   - Two return values: return dataframe1, dataframe2
    return dataframe1,
```


```
Notice the values for the **Mean_Absolute_Error** metric. The first value (the bigger one) c
orresponds to the model trained on feature set A. The second value (the smaller one) correspo
nds to the model trained on feature sets A + B.

It is remarkable how, using simple feature engineering to derive new features from the existi
ng data set, a new context was created that allowed the model to better understand the dynami
cs of the data and hence, produce a better prediction.
```

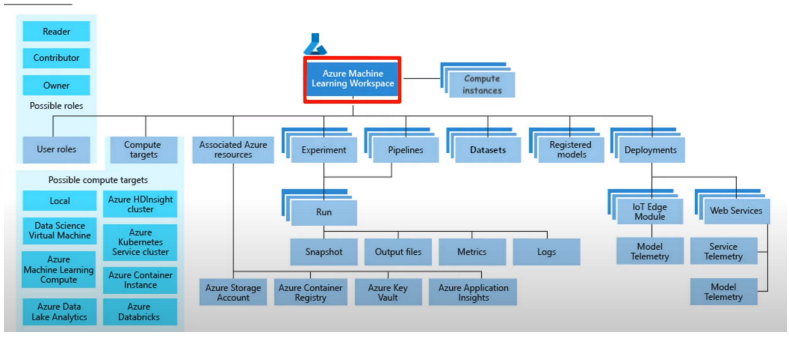
Model Training

Tuesday, July 14, 2020 10:11 PM

- The core model training process
  - Goal is to produce a trained model to use for predictions
  - Mathematically: predict feature Y based on input features X
  - Establish type of problem to solve first
    - Classification, regression etc.
    - Framing of problem influences choice of algorithm and various approaches to get to desired result
  - Prerequisite is understanding, transforming data and engineering / selecting features most relevant to training process
  - Next is to decide if scaling or encoding is needed
  - Model training is iterative process of selecting hyperparameters, training model and evaluating model performance
  - Once model is trained, run it on test data set to see performance of model on 'new' data
- Parameters and Hyperparameters
  - Part of training the model is learning the values of the parameters of the model usually from the data
    - Ex. Finding the missing coefficients in

      $$y = B_0 + B_1 * x_1 + B_2 * x_2 + B_3 * x_3 ... + B_n * x_n$$
  - Hyperparameters: model parameters that are **not** learned from data
    - Ex. # of layers in a deep neural network, # of clusters in a k-means algorithm, learning rate of a model
    - Hyperparameters are chosen and then tested / tuned based on model performance as the model is trained
- Splitting the Data
  - Splitting data into three major sets
    - Training set
      - Used to train the model
      - Used during training process
      - Used to determine the *parameters*
    - Validation set
      - Used to check the performance of the model
      - Used to determine the *hyperparameters*
      - Hyperparameters are tuned until the model performs well with validation data
        - Once hyperparameters are changed, they are retested on validation data
      - Used during training process
    - Test Set
      - Used to validate the model after training
- Model Training in Azure ML
  - Azure ML is a service that provide a comprehensive environment to implement model training processes
  - Centralized workspace with all artifacts utilized in the process
  - Taxonomy
    - Artifacts and Concepts
      - Workplace is the most vital, it encompasses the rest
      - Compute instances allow access to environments where code can be written and executed
        - Jupyter notebooks
        - Code not required, designer also works
      - Datasets are a key component that makes data available the ML training processes
      - Experiments are ML model tasks that are grouped together
        - Container that helps group various artifacts related to your ML processes
        - Run artifact is a process that is delivered and executed in one of the compute resources
          - Training model
          - Validation of model
          - Feature engineering that uses python code to create features
          - Outputs artifacts like output files, logs, metrics, snapshots
      - Possible compute targets are the environments on which your code can run
        - Supports remote environments
      - Model Registry
        - All models are registered here
        - Keeps track of snapshots and versioning so end->end traceability is possible
      - Deployments
        - Deploys registered models to various endpoints
      - Telemetry
        - Data is collected about model once model is deployed for various purposes

Taxonomy of Azure Machine Learning Workspace



- Two of the fundamental machine learning models: *Classifier* and *regressor* are both types of **supervised learning**
  - Training Classifiers
    - Classification: problem that occurs when expected outputs are discrete / categorical
    - Three main types of classification problems
      - Binary Classification
        - Only two output options, binary value is generated as output
          - Not necessarily easy or simple -> anomaly or fraud detection
      - Multiclass Single Label Classification
        - Output has multiple values, 3+ classes generally
        - Multiple potential outputs but output can only belong to single class
        - Ex. Recognition of written numbers, classify each image as one of the numbers
      - Multiclass Multi Label Classification
        - Multiple potential outputs and output can belong to more than one category
        - Ex. Text tagging where one or several tags can be assigned to a single chunk of text
    - Algorithms used to solve classification problems
      - Logistic Regression
      - Support Vector Machine
  - Training Regressors
    - Regressors: problem that occurs when expected outputs are continuous / numerical
    - Two main types of regression problems
      - Regression to arbitrary values
        - No boundary defined on output values, can be anything
        - Ex. Predicting price of houses based on inputs like location, sq ft, etc.
      - Regression to values between 0 and 1
        - Special case of regression problem where the output has to be a value between 0 and 1
        - Lots of optimizations that can be applied to certain algorithms if output is restricted
        - Ex. In fraud detection, instead of flagging yes or no, assign a probability that this transaction is fraudulent
    - Algorithms used to solve regressor problems
      - Linear Regressor
      - Decision Forest Regressor
- The model evaluation process and relevant metrics
  - Evaluating Model Performance
    - Test dataset is a portion of our labeled data that is split off and reserved for evaluating the model's final performance
      - When splitting data, its important to preserve the statistical properties of the data
        - Training, validation, test datasets all need to have similar statistical properties as the original data to prevent / minimize bias in the trained model
        - Ex. A dataset of customer purchases needs to be split so that a mix of customers from different cities needs to be present in all three sets
    - Metrics used to measure model performance are different based on the problem being solved
      - Ex. Classification algorithms have a different set of performance metrics than regression models
      - Establish criteria for the evaluation
        - What is primary metric for evaluation?
        - What is threshold value set so that model needs to meet or exceed before it's considered a "success"?
        - Goal is to guarantee a certain level of quality in your trained ML models
  - Confusion Matrices
    - Determines the possibilities for a classification problem
      - Ex. Training a simple binary classification model to determine if a picture is a dog or a cat
        - Model performing well:
          - images of a dog returns output that 'image contains dog'
          - Images of a cat returns output that 'image contains cat'
        - Confusion matrix looks like this:
		Actual class	
		Cat	**Dog**
Predicted class	**Cat**	**Correct Cat**	Incorrect Cat
	Dog	Incorrect Dog	**Correct Dog**
        - When predicted class matches actual class, a correct classification has been reached
    - Key is to look at diagonals, the name confusion matrix derives from the fact that it's easy to see if the model is getting confused and misclassifying the data or not
    - More generalized form looks like this:
		Actual class	
		Positive	**Negative**
Predicted class	**Positive**	**True Positives (TP)**	False Positives (FP)
	Negative	False Negatives (FN)	**True Negatives (TN)**
    - True positives: positive cases that are correctly predicted as positive by the model
    - False positives: negative cases that are incorrectly predicted as positive by the model
    - True negatives: negative cases that are correctly predicted as negative by the model
    - False negatives: positive cases that are incorrectly predicted as negative by the model

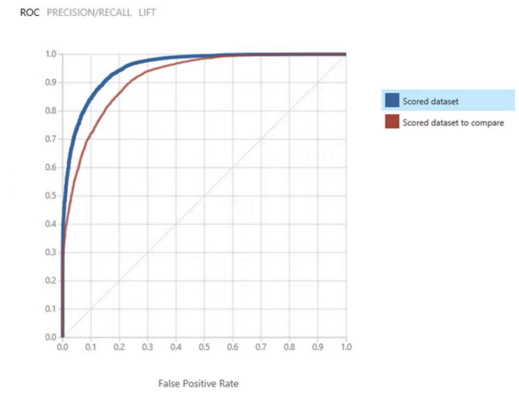
Evaluation Metrics

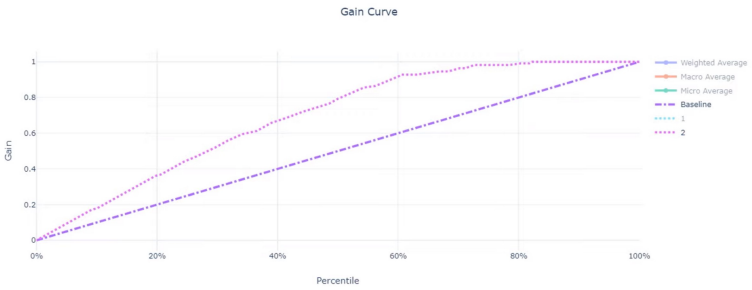
Thursday, July 16, 2020 8:18 PM

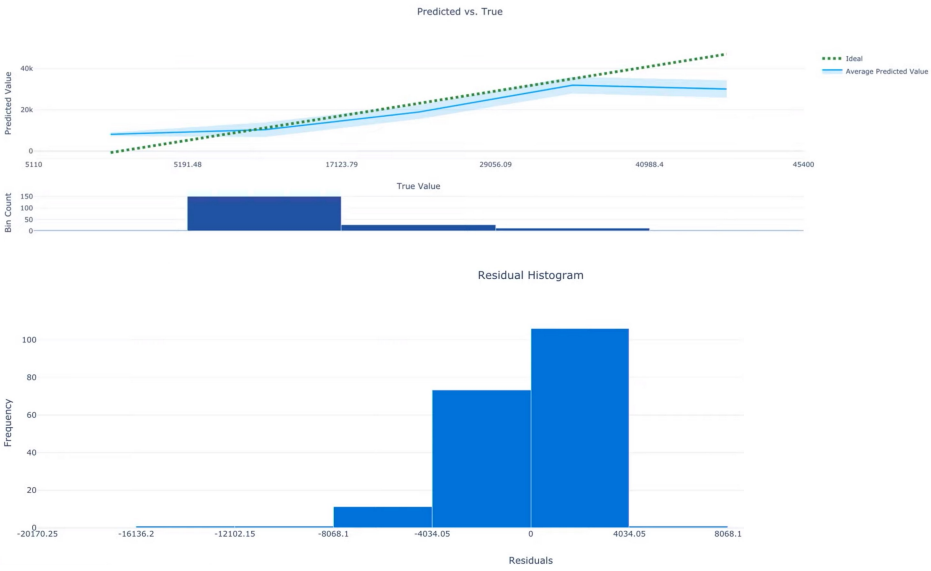
- Classification
  - Confusion matrix is a metric to be used to determine accuracy

Class	Positive	Negative
Positive	TP (True Positives)	FP (False Positives)
Negative	FN (False Negatives)	TN (True Negatives)

    - Evaluation metrics are inferred from this matrix
      - Accuracy: proportion of correct predictions
        - $\frac{TP+TN}{TP+FP+FN+TN}$
      - Precision: proportion of **correct** positive cases over all cases marked positive
        - $\frac{TP}{TP+FP}$
      - Recall: proportion of positives correctly identified
        - $\frac{TP}{TP+FN}$
      - F1 Score: measures balance between precision and recall
        - $2 * \frac{Precision*Recall}{Precision+Recall}$
      - None of these alone are enough, needs to use at least 2 to get a complete image of the classification algorithm
        - Need to use at least two because just knowing if it's marking things right isn't enough, you also need to know what it's doing when it's making mistakes. That way you can figure out the best approach to correct the mistakes. For example, if you only look at accuracy, you know how many correct there are. If you only look at precision, you know that out of the X number your model marked positive, only Y are actually positive. If you only look at Recall, you know that of all the things it should have marked positive, it only marked Z positive. These pieces of information alone aren't super useful in telling you how to correct your algorithm, you need to look at multiple together to get the whole picture.
  - Model Evaluation Charts
    - Receiver Operating Characteristics (ROC) chart
      - Graph of Rate of True Positives vs Rate of False Positives
      - Area Under Curve (AUC) tells you correctness of cases
        - Diagonal line is 0.5 -> random guessing
        - If AUC is 1, then it gets all cases correct
    - Gain and Lift chart
      - Ordering (with rank) the prediction probabilities
      - Measures how much better you are expected to perform by using the classifier over random guessing
        - Baseline is the performance expected when you randomly guess
        - You want the line of expected performance using classifier to be as far from the baseline as possible
- Regression
  - Yields continuous outputs so requires different set of metrics
    - Root Mean Squared Error (RMSE)
      - Metric that measures the square root of the squared differences between predicted and actual values
    - Mean Absolute Error
      - Average of the absolute difference between each prediction and the true value
    - R-Squared
      - Coefficient of determination, how close the predicted values are to the fitted regression line
    - Spearman Correlation
      - Measures the strength and direction of the correlation between predicted values and true values
  - Charts
    - Predicted vs. True chart
      - Displays relationship between predicted value and true value
      - Ideal regressor (diagonal line) is perfectly accurate
      - Histogram of how your true values are distributed in your prediction results
    - Histogram of Residuals
      - Distribution of (true value - predicted value)
      - Ideal distribution is a perfect bell chart indicating low bias







Lab: Train/Evaluate a Model

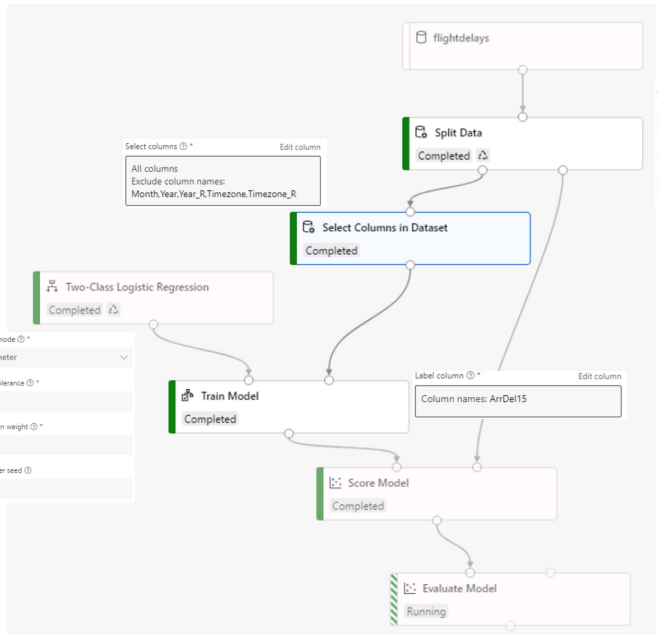
Friday, July 17, 2020 12:05 AM

`https://introtomlsampledata.blob.core.windows.net/data/flightdelays/flightdelays.csv`

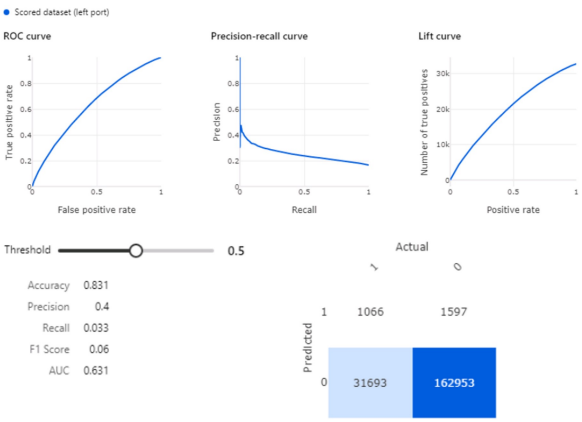
Lab Overview

Azure Machine Learning designer (preview) gives you a cloud-based interactive, visual workspace that you can use to easily and quickly prep data, train and deploy machine learning models. It supports Azure Machine Learning compute, GPU or CPU. Machine Learning designer also supports publishing models as web services on Azure Kubernetes Service that can easily be consumed by other applications.

In this lab, we will be using the `Flight Delays` data set that is enhanced with the weather data. Based on the enriched dataset, we will learn to use the Azure Machine Learning Graphical Interface to process data, build, train, score, and evaluate a classification model to predict if a particular flight will be delayed by 15 minutes or more. To train the model, we will use Azure Machine Learning Compute resource. We will do all of this from the Azure Machine Learning designer without writing a single line of code.



Evaluate Model result visualization



Score bin ↓	Positive exam...	Negative exam...	Fraction above thresh...	Accuracy	F1 Score	Precisi...	Recall	Negative precisi...	Negative recall	Cumulative AUC
(0.900,1.000]	0	0	0.000	0.834	0.000	1.000	0.000	0.834	1.000	0.000
(0.800,0.900]	1	0	0.000	0.834	0.000	1.000	0.000	0.834	1.000	0.000
(0.700,0.800]	28	57	0.000	0.834	0.002	0.337	0.001	0.834	1.000	0.000
(0.600,0.700]	247	267	0.003	0.834	0.017	0.460	0.008	0.835	0.998	0.000
(0.500,0.600]	790	1273	0.013	0.831	0.060	0.400	0.033	0.837	0.990	0.000
(0.400,0.500]	2154	4889	0.049	0.817	0.152	0.332	0.098	0.843	0.961	0.002
(0.300,0.400]	5262	15281	0.153	0.767	0.269	0.280	0.259	0.855	0.868	0.019
(0.200,0.300]	8780	35073	0.376	0.633	0.323	0.233	0.527	0.874	0.655	0.104
(0.100,0.200]	11265	65076	0.762	0.361	0.311	0.190	0.871	0.910	0.259	0.387
(0.000,0.100]	4232	42634	1.000	0.166	0.285	0.166	1.000	1.000	0.000	0.631

Ensemble Learning / Automated ML

Thursday, July 16, 2020 11:48 PM

- Difficult problem in ML -> significant chance that a model can produce wrong results no matter how well trained it is
    - Don't rely on a single model, train multiple instances of ML model and capture collective wisdom to alleviate magnitude of issues produced by individual models
    - Ensemble learning relies on multiple models / multiple algorithms and somehow capturing collective results
    - Automated ML relies on scaling up the process of training models which uses quantity to reduce error

- Ensemble Learning
    - Combing multiple machine learning models to produce one predictive model
    - Take multiple ML algorithms and train on the same data then combine results somehow
    - Three main types
        - Bagging / bootstrap aggregation -> reduce overfitting
            - Reduces variance
            - Homogeneous learners
            - Random sampling with replacement
                - Random subsampling is used to train ML models, process is repeated multiple times
                    - Results in a number of trained ML models
            - Equally weighed average
                - Assign equal weights to predicted outputs of each ML model and is then combined
        - Boosting -> produce a strong learner using weaker learners
            - Use same input data but train multiple ML models using different values in hyperparameters
            - Use same learners but constantly improving performance of resulting model
            - Weak, homogeneous learners
            - Reduces bias
            - Sequential learning
            - Weighted average predictions
        - Stacking
            - Training a large number of completely different models, take output and combine into final output
            - Heterogeneous learners
            - Meta model that learns to combine predictions from base learners
            - Improves prediction accuracy

- Automated Machine Learning -> "Strength in Variety"
    - Automates many of the iterative, time-consuming tasks
        - Feature selection/engineering, scaling features, choosing algorithms, tuning hyperparameters etc.
        - **Trains multiple versions of models with different combinations of hyperparameters / features / algorithms**
    - Allows ML engineers / analysts / data scientists to build models with greater scale, efficiency and productivity while still sustaining model quality
    - Ex. Predicting customer behavior
        - So many algorithms can be used to predict it, how do you pick best one?
        - Fine tuning model hyperparameters is also difficult and time consuming to get correct

        - Different values ('different knob positions') produce different models and results
    - Automated ML explores different combinations to find the best combination for producing the most accurate predictions and results for your problem
    - Azure ML Automation -> 0 code experience
        - Select dataset to be used for ML training procss
        - Allocate set of compute resources to run training jobs
        - Select type of task (classification, regression etc)
        - Select primary evaluation metric and define the "criteria for finishing"
            - Criteria for finishing: level of quality at which the process can stop
        - Automated ML experiment will find best performing model
            - In many cases it ends up being an ensemble model
    - **Not a replacement for data scientists, should be regarded as a way to get a baseline -> a model that has good performance and then you build off the good starting point to improve it further**

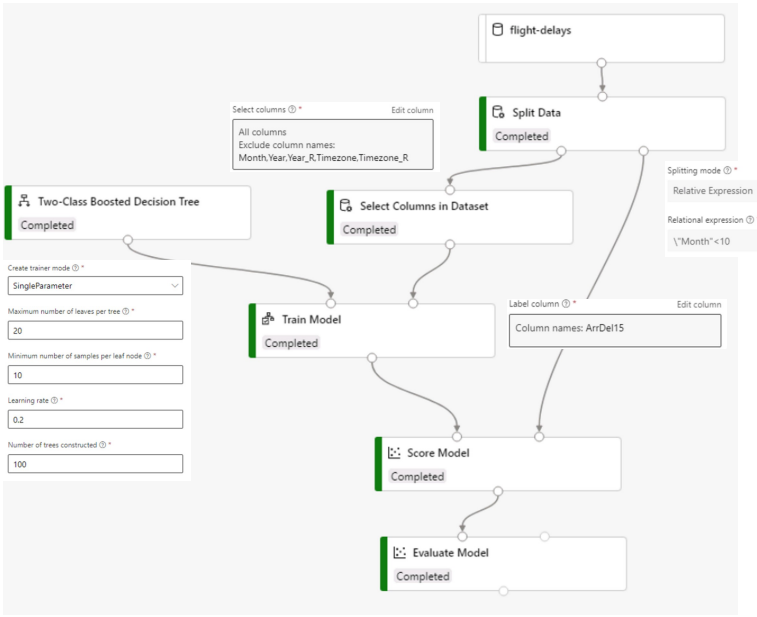
Lab: Train a Two-Class Boosted Decision Tree

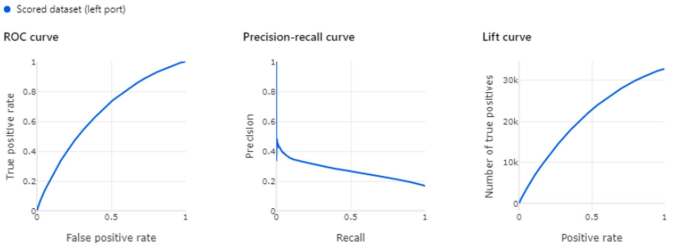
Friday, July 17, 2020 12:05 AM

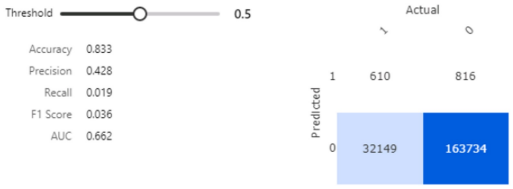
Lab Overview

Azure Machine Learning designer (preview) gives you a cloud-based interactive, visual workspace that you can use to easily and quickly prep data, train and deploy machine learning models. It supports Azure Machine Learning compute, GPU or CPU. Machine Learning designer also supports publishing models as web services on Azure Kubernetes Service that can easily be consumed by other applications.

In this lab, we will be using the `Flight Delays` data set that is enhanced with the weather data. Based on the enriched dataset, we will learn to use the Azure Machine Learning Graphical Interface to process data, build, train, score, and evaluate a classification model to predict if a particular flight will be delayed by 15 minutes or more. The classification algorithm used in this lab will be the ensemble algorithm: **Two-Class Boosted Decision Tree**. To train the model, we will use Azure Machine Learning Compute resource. We will do all of this from the Azure Machine Learning designer without writing a single line of code.







Score bin ↓	Positive exam...	Negative exam...	Fraction above thresh...	Accuracy	F1 Score	Precisi...	Recall	Negative precisi...	Negative recall	Cumulative AUC
(0.900,1.000]	0	0	0.000	0.834	0.000	1.000	0.000	0.834	1.000	0.000
(0.800,0.900]	1	2	0.000	0.834	0.000	0.333	0.000	0.834	1.000	0.000
(0.700,0.800]	32	38	0.000	0.834	0.002	0.452	0.001	0.834	1.000	0.000
(0.600,0.700]	124	132	0.002	0.834	0.009	0.477	0.005	0.834	0.999	0.000
(0.500,0.600]	453	644	0.007	0.833	0.036	0.428	0.019	0.836	0.995	0.000
(0.400,0.500]	1448	2626	0.028	0.827	0.108	0.374	0.063	0.840	0.979	0.001
(0.300,0.400]	4558	10382	0.104	0.797	0.249	0.324	0.202	0.852	0.916	0.009
(0.200,0.300]	9795	31654	0.314	0.687	0.347	0.265	0.501	0.879	0.724	0.078
(0.100,0.200]	12398	71691	0.740	0.386	0.322	0.197	0.879	0.923	0.288	0.389
(0.000,0.100]	3950	47381	1.000	0.166	0.285	0.166	1.000	1.000	0.000	0.662

Lab: Train a Simple Classifier with Automated ML

Friday, July 17, 2020 12:05 AM

Lab Overview

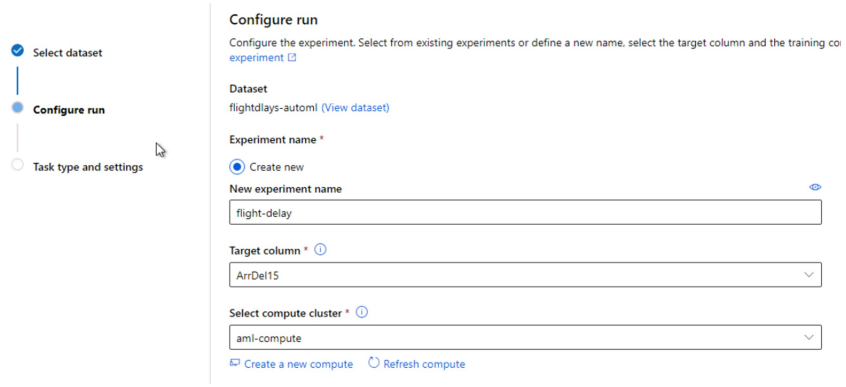
Automated machine learning picks an algorithm and hyperparameters for you and generates a model ready for deployment. There are several options that you can use to configure automated machine learning experiments.

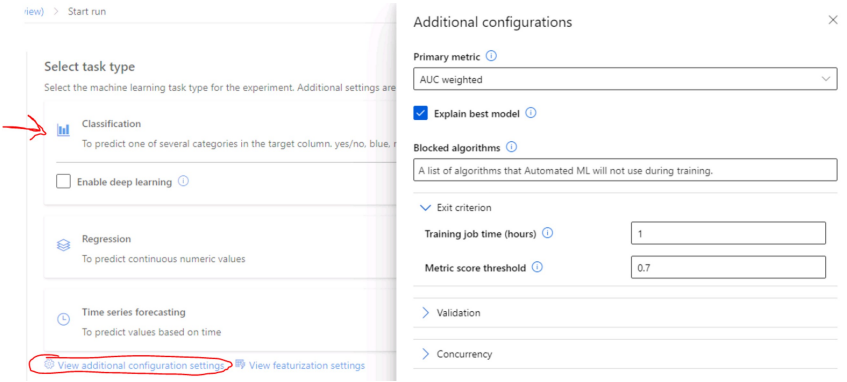
Configuration options available in automated machine learning:

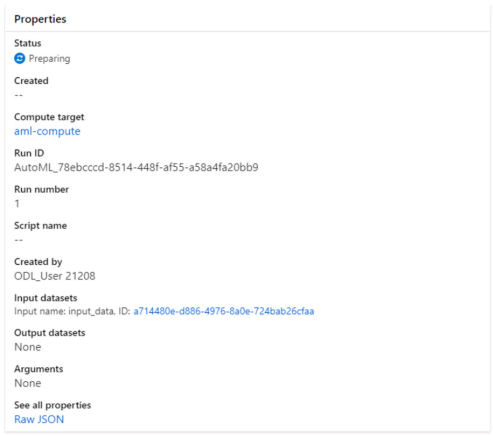
- Select your experiment type: Classification, Regression or Time Series Forecasting
- Data source, formats, and fetch data
- Choose your compute target
- Automated machine learning experiment settings
- Run an automated machine learning experiment
- Explore model metrics
- Register and deploy model

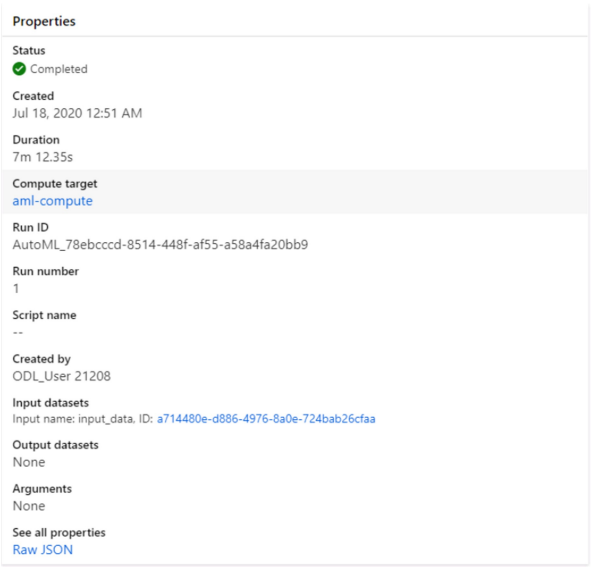
You can create and run automated machine learning experiments in code using the Azure ML Python SDK or if you prefer a no code experience, you can also create your automated machine learning experiments in Azure Machine Learning Studio.

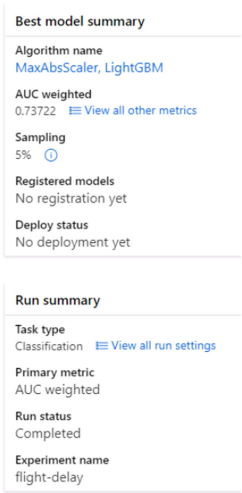
In this lab, you learn how to create, run, and explore automated machine learning experiments in the Azure Machine Learning Studio without a single line of code. As part of this lab, we will be using the `Flight Delays` data set that is enhanced with the weather data. Based on the enriched dataset, we will use automated machine learning to find the best performing classification model to predict if a particular flight will be delayed by 15 minutes or more.

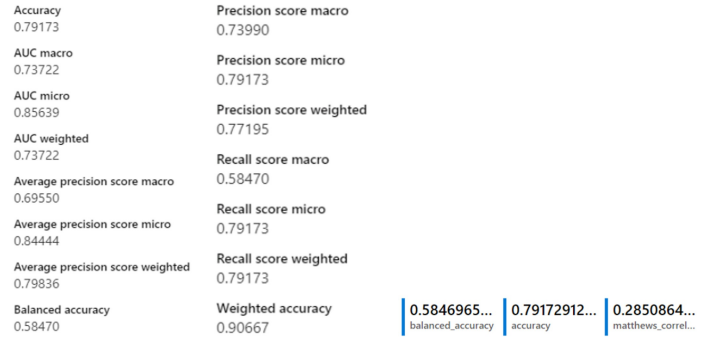


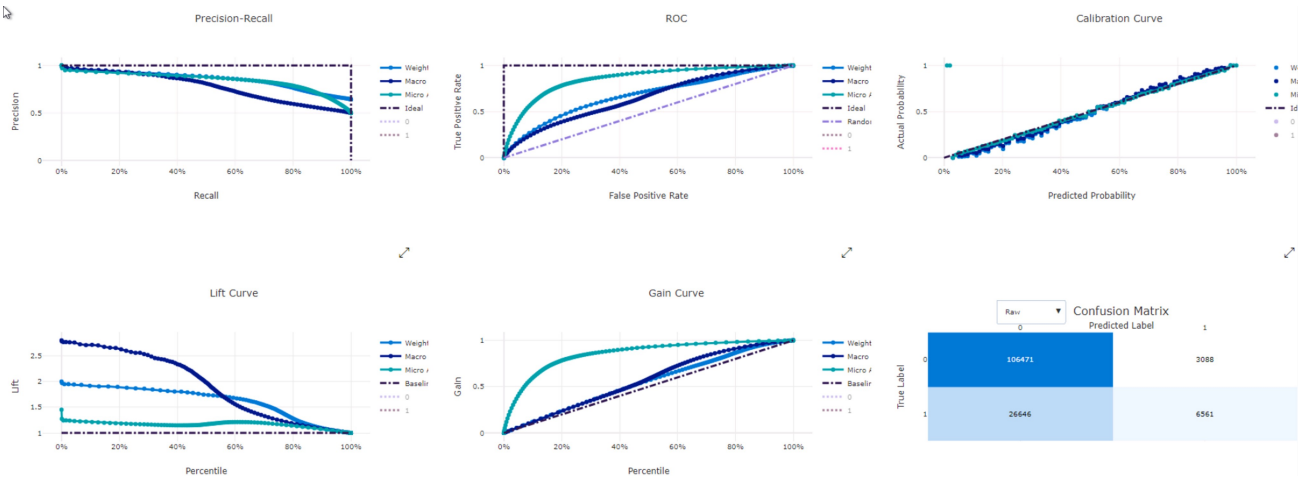












F1 score macro
0.59183

F1 score micro
0.79173

F1 score weighted
0.74459

Log loss
0.47177

Matthews correlation
0.28509

Norm macro recall
0.16939