Table of Contents

Lesson 4 – Supervised & Unsupervised Learning

Chapter 1: Lesson Overview

This lesson covers two of Machine Learning's fundamental approaches: **supervised** and **unsupervised** learning.

First, we will cover **supervised learning**. Specifically, we will learn:

- More about *classification* and *regression*, two of the most representative supervised learning tasks
- Some of the major *algorithms* involved in supervised learning, as well as how to evaluate and compare their performance
- How to use *automated machine learning* to automate the training and selection of classifiers and regressors, and how to use the Designer in Azure Machine Learning Studio to create automated Machine Learning experiments

Next, the lesson will focus on **unsupervised learning**, including:

- Its most representative learning task, *clustering*
- How unsupervised learning can address challenges like lack of labelled data, the curse of dimensionality, overfitting, feature engineering, and outliers
- An introduction to *representation learning*
- How to train your first clustering model in Azure Machine Learning Studio

Chapter 2: Prelaunch Lab

Prelaunch your lab environment.

NOTE: When you click "Prelaunch Lab", we will begin preparing a lab environment for you. When it is time to access the lab environment, this ensures you will be able to start working. Once your lab is reserved, you can continue on to the next concepts.

Your lab has been, reserved! Please continue to the next concept.

Chapter 3: Supervised Learning: Classification

The first type of *supervised learning* that we will look at is *classification*. Recall that the main distinguishing characteristic of classification is the type of output it produces:

*In a **classification** problem, the outputs are categorical or discrete & predicted output belongs to one or more of those categories.*

Within this broad definition, there are several main approaches, which differ based on how many classes or categories are used, and whether each output can belong to only one class or multiple classes. Let us have a look.

In case of Binary classification, the output has only two categories.

Some of the most common types of classification problems include:

- ***Classification on tabular data:*** The data is available in the form of rows and columns, potentially originating from a wide variety of data sources.
- ***Classification on image or sound data:*** The training data consists of images or sounds whose categories are already, known.
- ***Classification on text data:*** The training data consists of texts whose categories are already, known.

As we discussed in a previous lesson, machine learning requires numerical data. This means that with images, sound, and text, several steps need to be, performed during the preparation phase to transform the data into numerical vectors that can be, accepted by the classification algorithms.

Examples:

- Computer vision
- Speech recognition
- Biometric identification
- Document classification
- Sentiment analysis
- Credit scoring in banking
- Anomaly detection

Categories of Algorithms

There are 3 main categories of Supervised ML algorithms for the Classification problem.

1. **Two Class / Binary Classification** – These are used when we have to predict between 2 Categories. [Yes/No or True/False]
2. **Multi Class Single Label Classification** – These are, used when predicting between several categories however, the output would belong to a single category. [Output Categories would be multiple however Output would fall under only one category]
3. **Multi Class Multi Label Classification** – These are, used when predicting between several categories & the output would belong to one or more categories. [Output Categories would be multiple & Output can fall under one or multiple categories]

Two-Class Classification Algorithms

Predict between **two** categories (**binary** classification)

Algorithm	Characteristics
Two-Class Support Vector Machine	Under 100 features, linear model
Two-Class Averaged Perceptron	Fast training times, linear model
Two-Class Decision Forest	Accurate, fast training times
Two-Class Logistic Regression	Fast training times, linear model
Two-Class Boosted Decision Tree	Accurate, fast training, large memory footprint
Two-Class Neural Network	Accurate, long training times

- SVM is a linear model and works best when used for problems with less than 100 features.
- Average Perceptron Algorithm is also a linear model & it has fast training times.
- Decision Forest is an ensemble algorithm & it is as well accurate with fast training times.
- Logistic regression is also a linear model & it has fast training times.
- Boosted Decision Tree is an ensemble algorithm & it is as well accurate with fast training times, however it has large memory footprint.
- Two-class neural network is accurate but has long training times.

Multi-Class Classification Algorithms

Predict between **several** categories

Algorithm	Characteristics
Multi-Class Logistic Regression	Fast training times, linear model
Multi-Class Neural Network	Accurate, long training times
Multi-Class Decision Forest	Accurate, fast training times
Multi-Class Boosted Decision Tree	Non-parametric, fast training times, and scalable
One-vs-All Multiclass	Depends on the underlying two-class classifier

- One vs. All Multiclass algorithm is, used for multiclass single label problems. It trains a binary classifier, where each classifier is trained to predict one end class. The class with the highest probability or scores across all the binary classifiers will be the predicted class.

QUESTION 1 OF 2

As we just discussed, at a high level there are three main categories of classification algorithms. Can you match each of them with the correct description?

Submit to check your answer choices!

DESCRIPTION	TYPE OF CLASSIFICATION
The classifier chooses from multiple categories; each output can belong to one or more categories.	Multi-class multi-label classification
The classifier chooses from multiple categories; each output belongs to single category only.	Multi-class single-label classification
The classifier choose from only two categories; each output belongs to one or the other.	Binary classification

QUESTION 2 OF 2

Given the following confusion matrix:

Class	Positive	Negative
Positive	50	5
Negative	20	100

What is the value for the Precision metric?

- ○ 0.71
- ○ 0.95
- ✓ 0.91
- ○ 0.86

Chapter 4: Lab - Two Class Classifiers Performance

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.

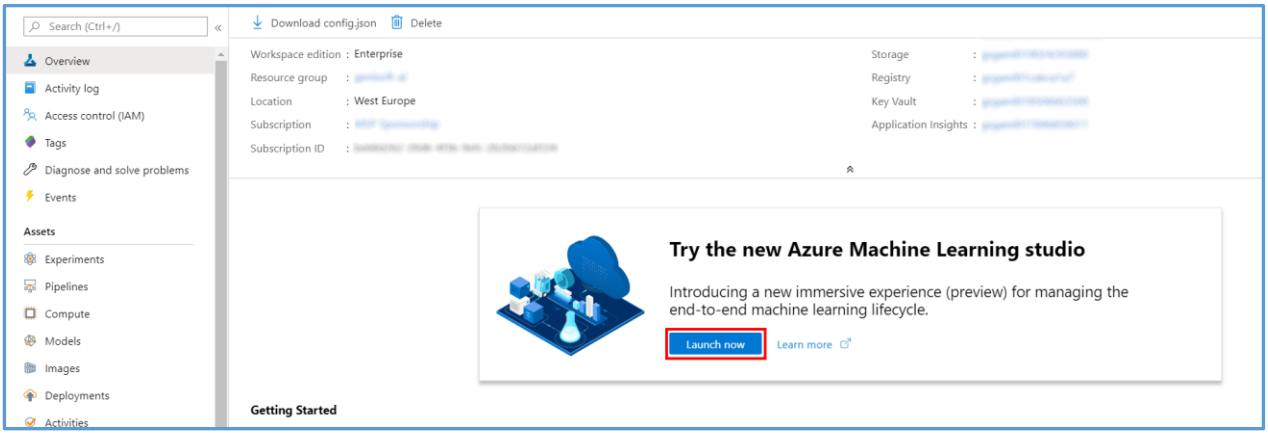
In this lab, we will be compare the performance of two binary classifiers: Two-Class Boosted Decision Tree and Two-Class Logistic Regression for predicting customer churn. The goal is to run an expensive marketing campaign for high, risk customers; thus, the **precision** metric is going to be key in evaluating performance of these two algorithms. We will do all of this from the Azure Machine Learning designer without writing a single line of code.

Exercise 1: Create Training Pipeline

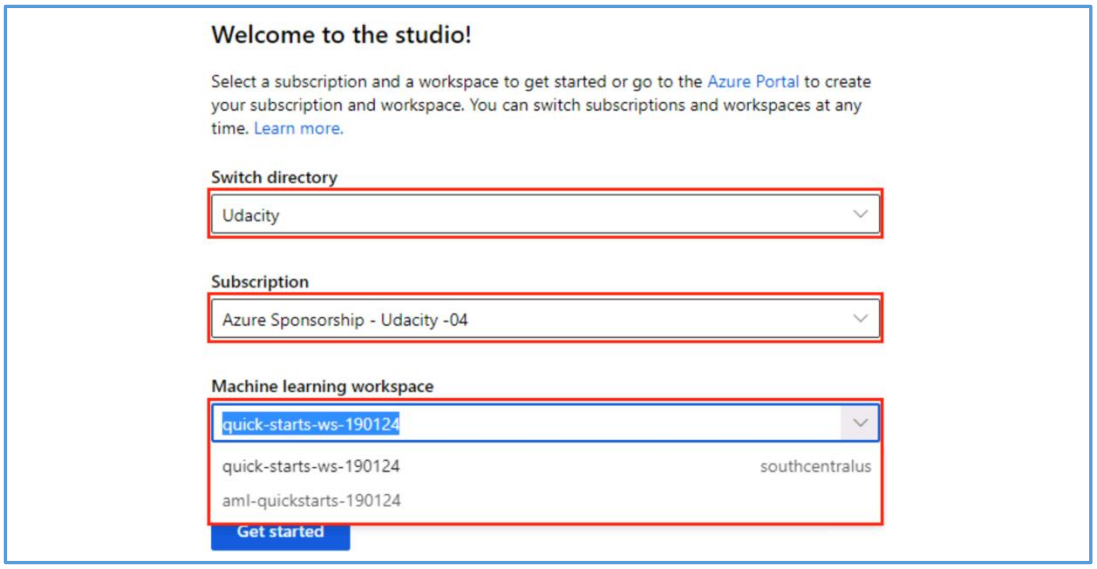
Task 1: Open Sample 5: Binary Classification – Customer Relationship Prediction

1. In Azure portal, open the available machine learning workspace.

2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

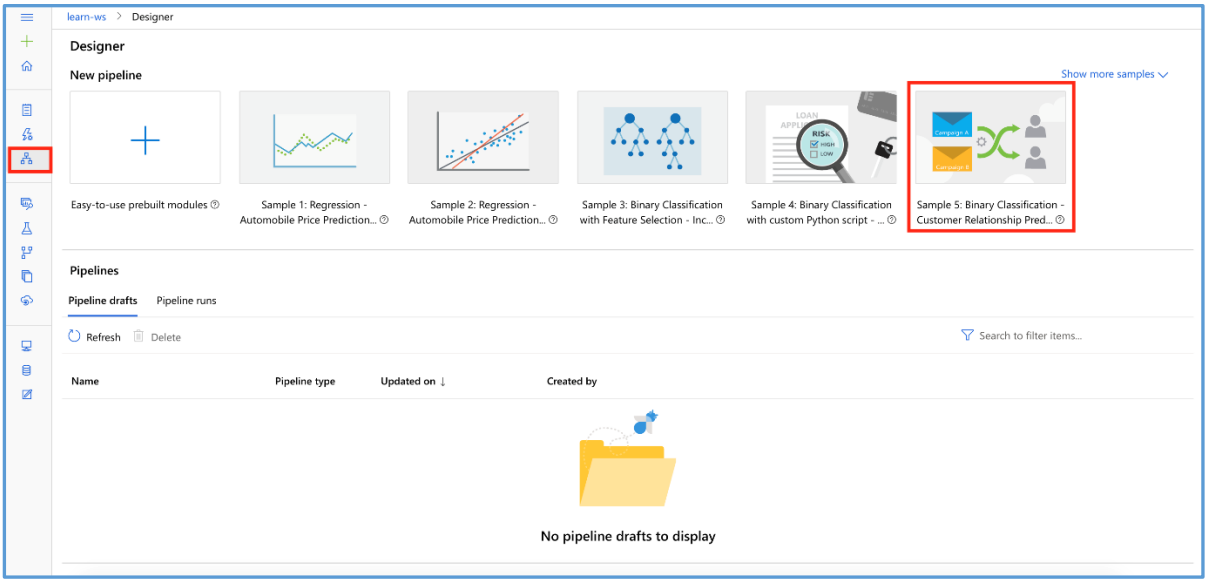


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



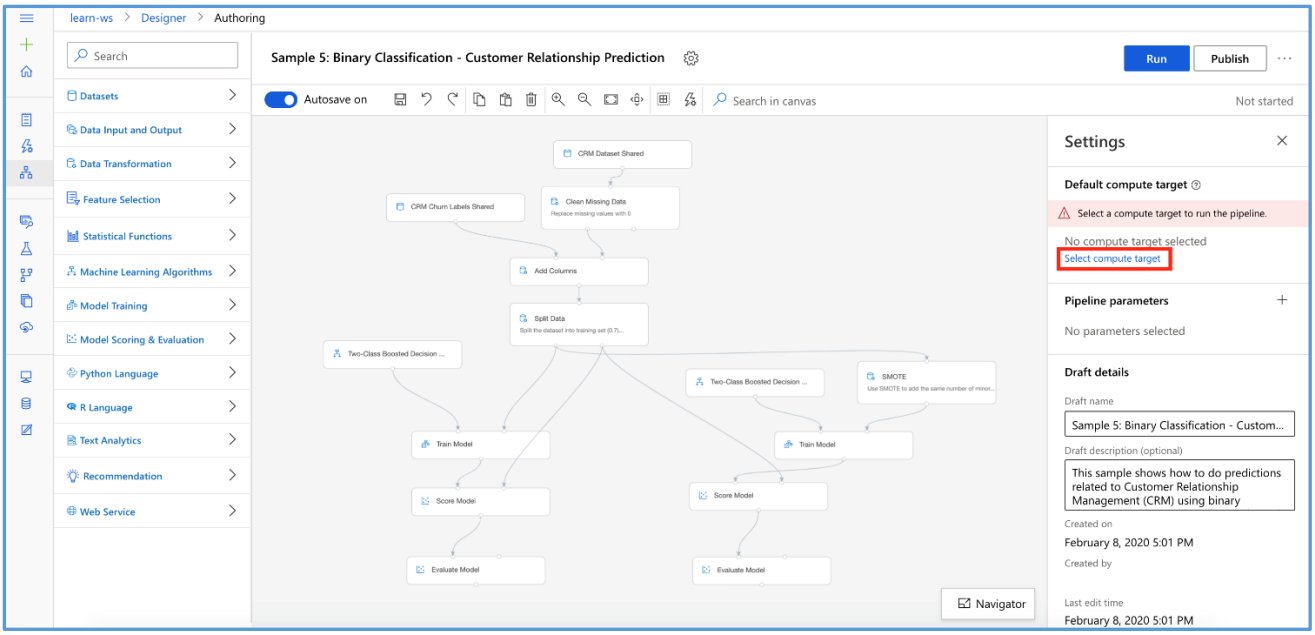
For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it does not matter which) and then click **Get started**.

4. From the studio, select **Designer, Sample 5: Binary Classification – Customer Relationship Prediction**.



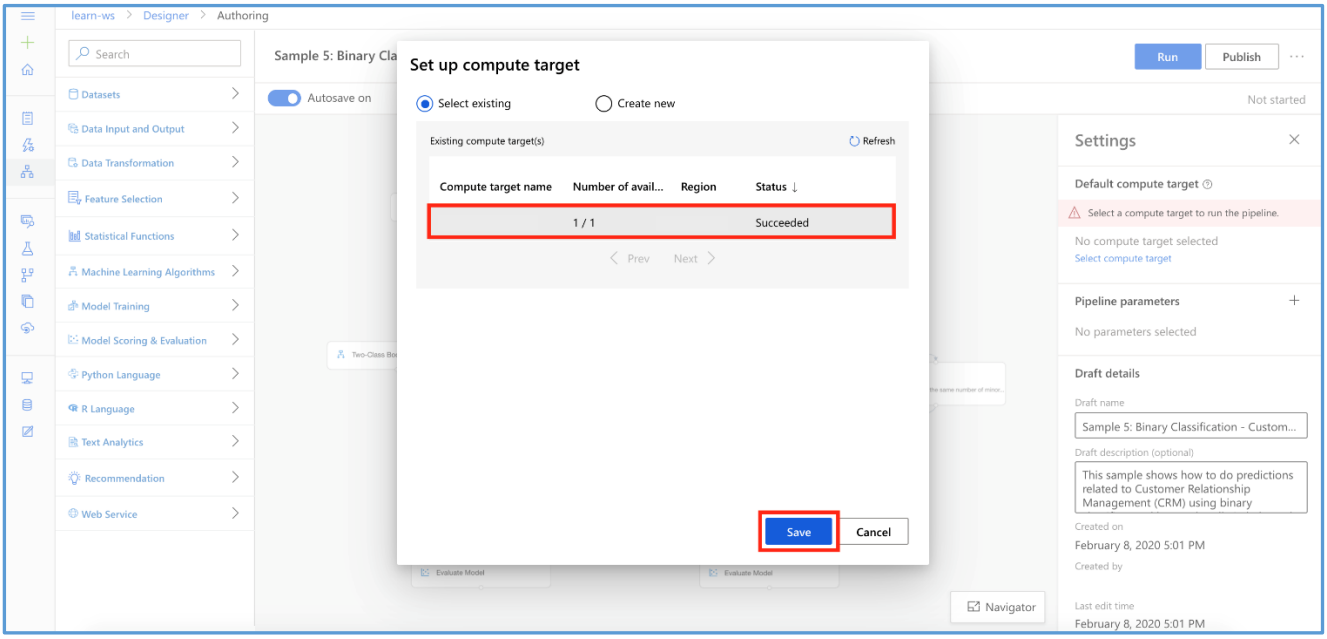
Task 2: Setup Compute Target

1. In the settings panel on the right, select **Select compute target**.



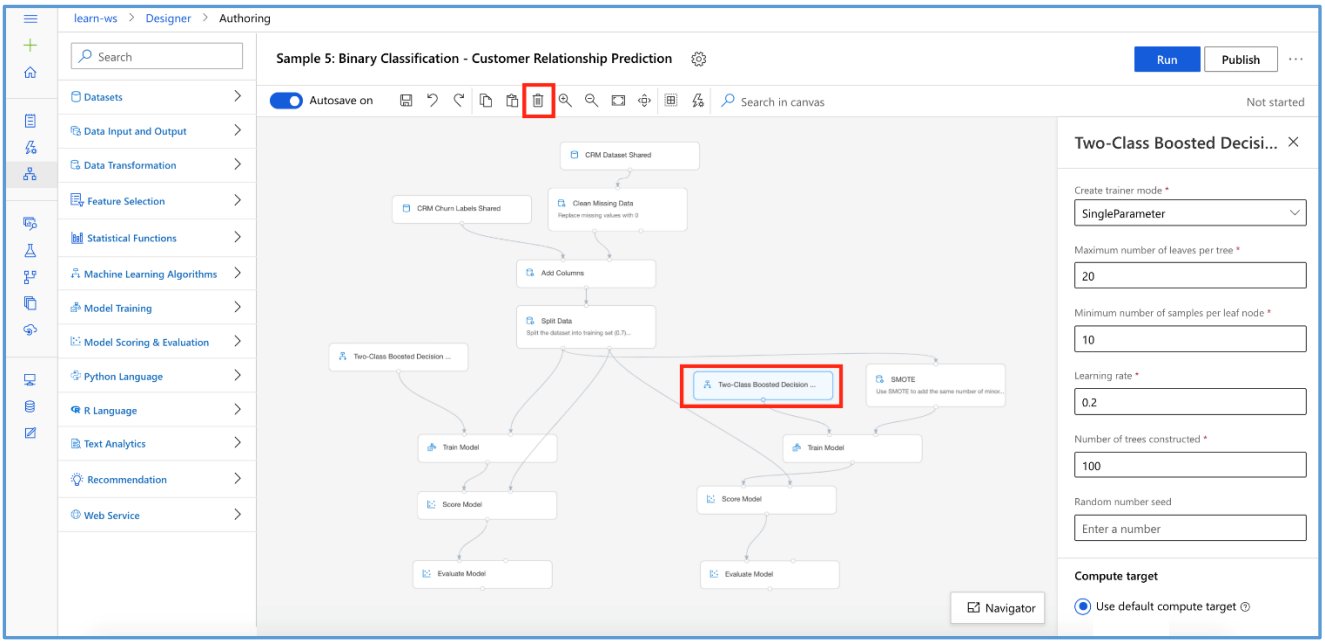
2. In the Set up compute target editor, select the available compute, and then select **Save**.

Note: If you are facing difficulties in accessing pop-up windows or buttons in the user interface, please refer to the Help section in the lab environment.

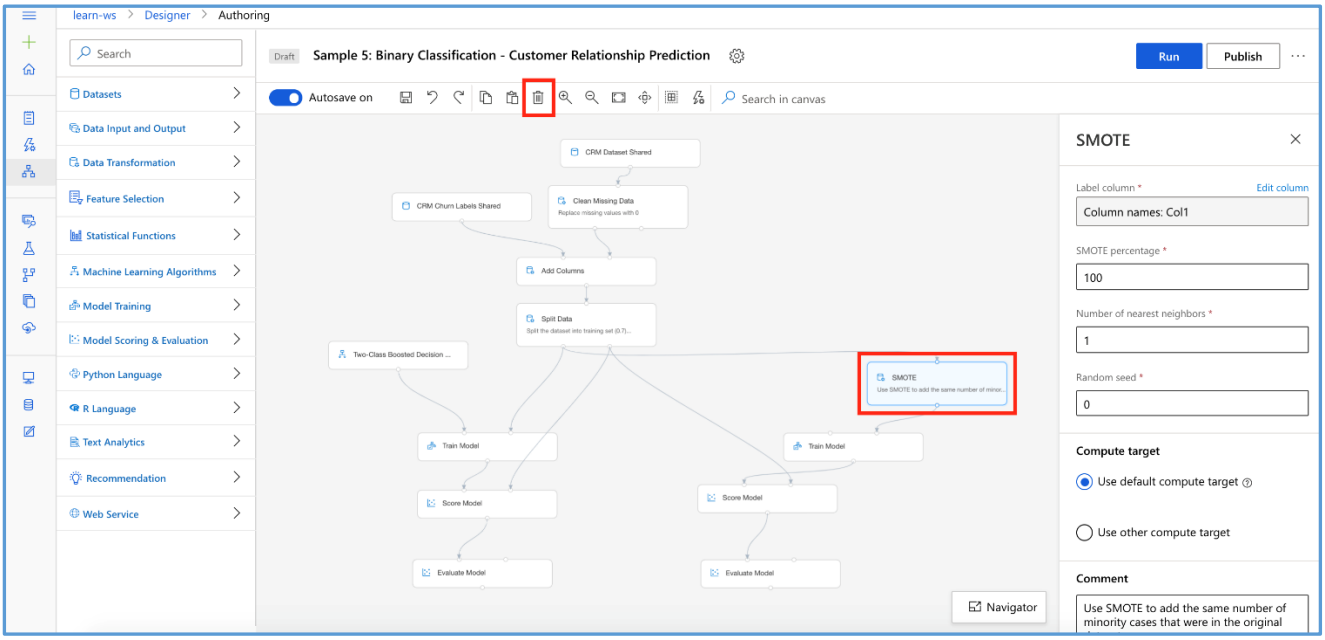


Task 3: Delete Pipeline Modules

1. From the right-hand-side of the pipeline, select the **Two-Class Boosted Decision Tree module** and then select the **Delete Icon**.



2. From the right-hand-side of the pipeline, select the **SMOTE module** and then select the **Delete Icon**.



Task 4: Setup the Two-Class Logistic Regression Module

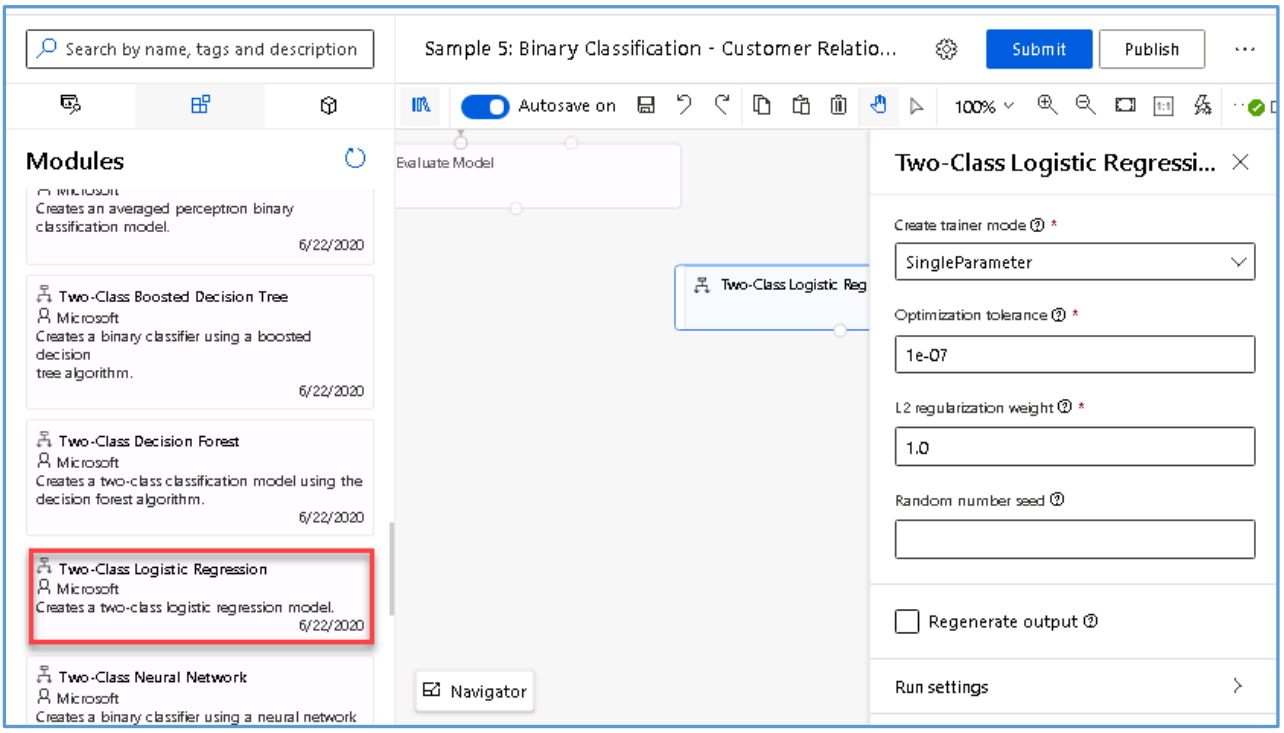
1. Select **Machine Learning Algorithms** section in the left navigation. Follow the steps outlined below:

    1. Select the **Two-Class Logistic Regression** prebuilt module

    2. Drag and drop the selected module on to the canvas

3. Connect the Two-Class Logistic Regression module to the first input of the Train Model module

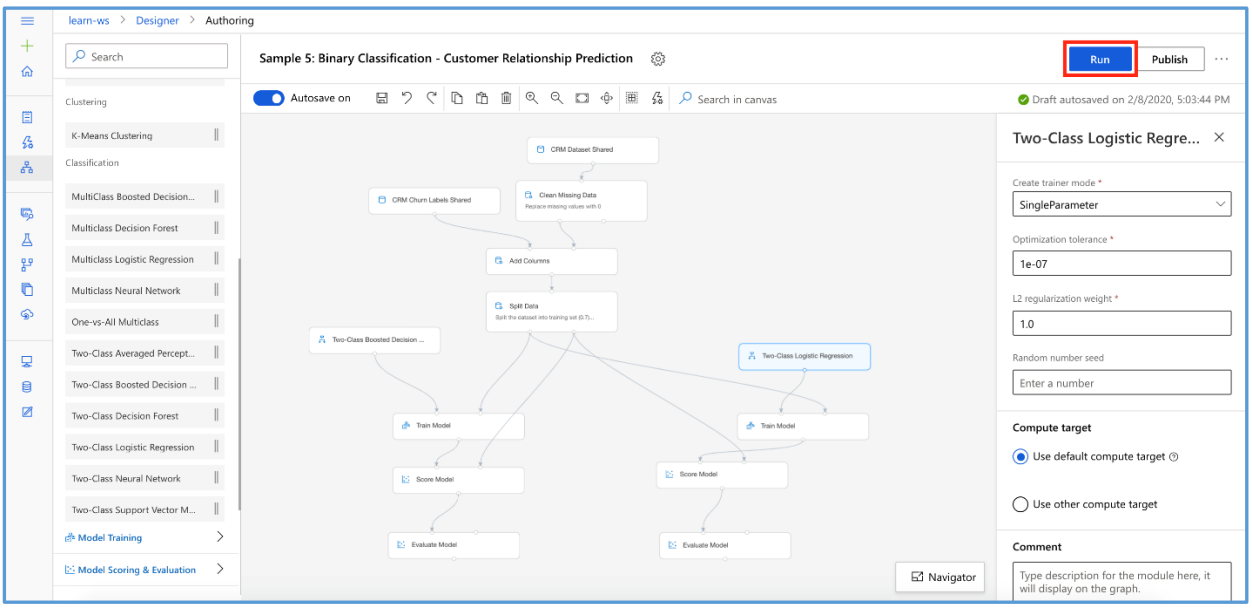
4. Connect the first output of the Split Data module to the second input of the Train Model module



Exercise 2: Submit Training Pipeline

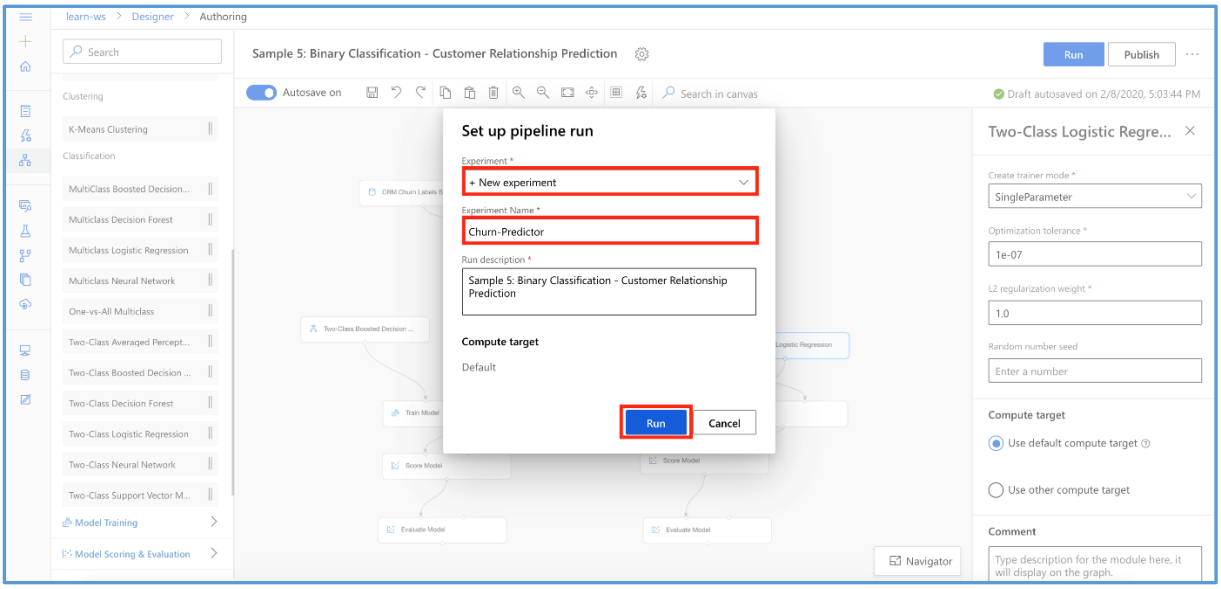
Task 1: Create Experiment and Submit Pipeline

1. Select **Submit** to open the Setup pipeline run editor.



Please note that the button name in the UI is changed from **Run** to **Submit**.

2. In the Setup pipeline run editor, select **Experiment, Create new** and provide New experiment name: **Churn-Predictor**, and then select **Submit**.



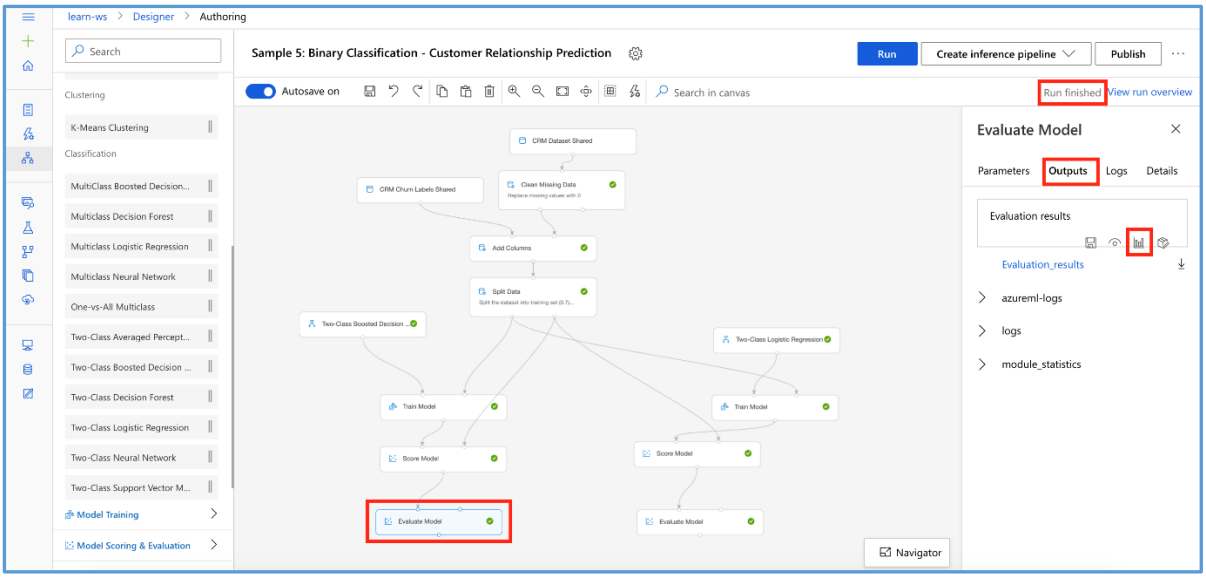
3. Wait for pipeline run to complete. It will take around **5 minutes** to complete the run.

4. While you wait for the model training to complete, you can learn more about the evaluation metrics for the classification algorithm used in this lab by selecting Metrics for classification models.

Exercise 3: Compare Model Performance

Task 1: Open Evaluation Results for Two-Class Boosted Decision Tree

1. From the left-hand-side of the pipeline, select **Evaluate Model, Outputs, Visualize** to open the Evaluate Model result visualization dialog for the Two-Class Boosted Decision Tree module.

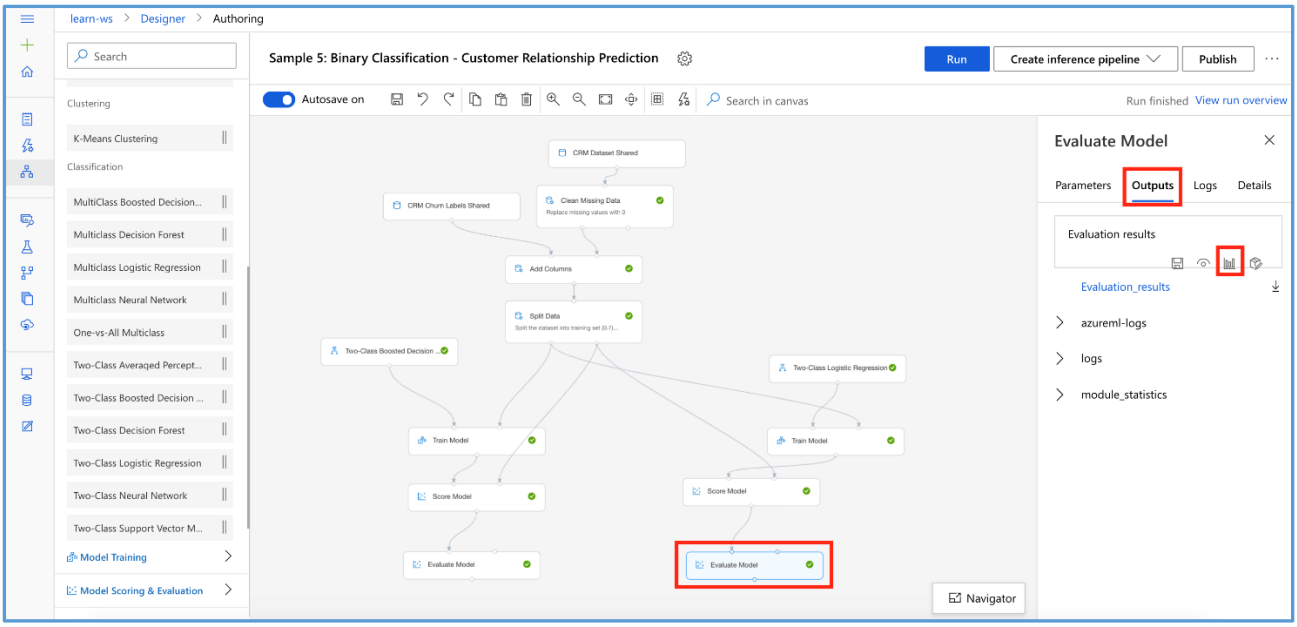


Task 2: Evaluate Two-Class Boosted Decision Tree Performance

1. Scroll down to review model performance metrics for Two-Class Boosted Decision Tree. Observe that the **Precision** value is around **0.7**.

Task 3: Open Evaluation Results for Two-Class Logistic Regression

1. From the right-hand-side of the pipeline, select **Evaluate Model, Outputs, Visualize** to open the Evaluate Model result visualization dialog for the Two-Class Logistic Regression module.



Task 4: Evaluate Two-Class Logistic Regression Performance

1. Scroll down to review model performance metrics for Two-Class Logistic Regression. Observe that the **Precision** value is around **0.3**.

Task 5: Conclusion

1. Based on the primary performance metric, Precision, it shows that the Two-Class Boosted Decision Tree algorithm outperforms the Two-Class Logistic Regression algorithm.

Next Steps

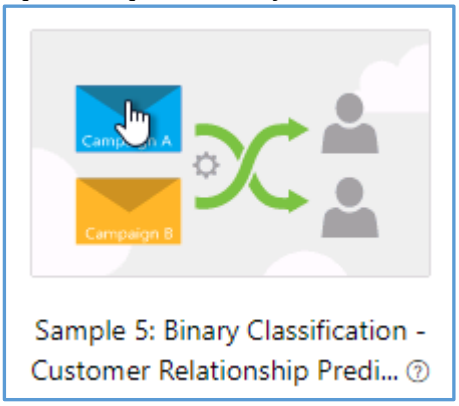
Congratulations! You have trained and compared performance of two different classification machine-learning models. You can continue to experiment in the environment but are free to close the lab environment tab and return to the Udacity portal to continue with the lesson.

Chapter 5: Walkthrough - Two Class Classifiers Performance

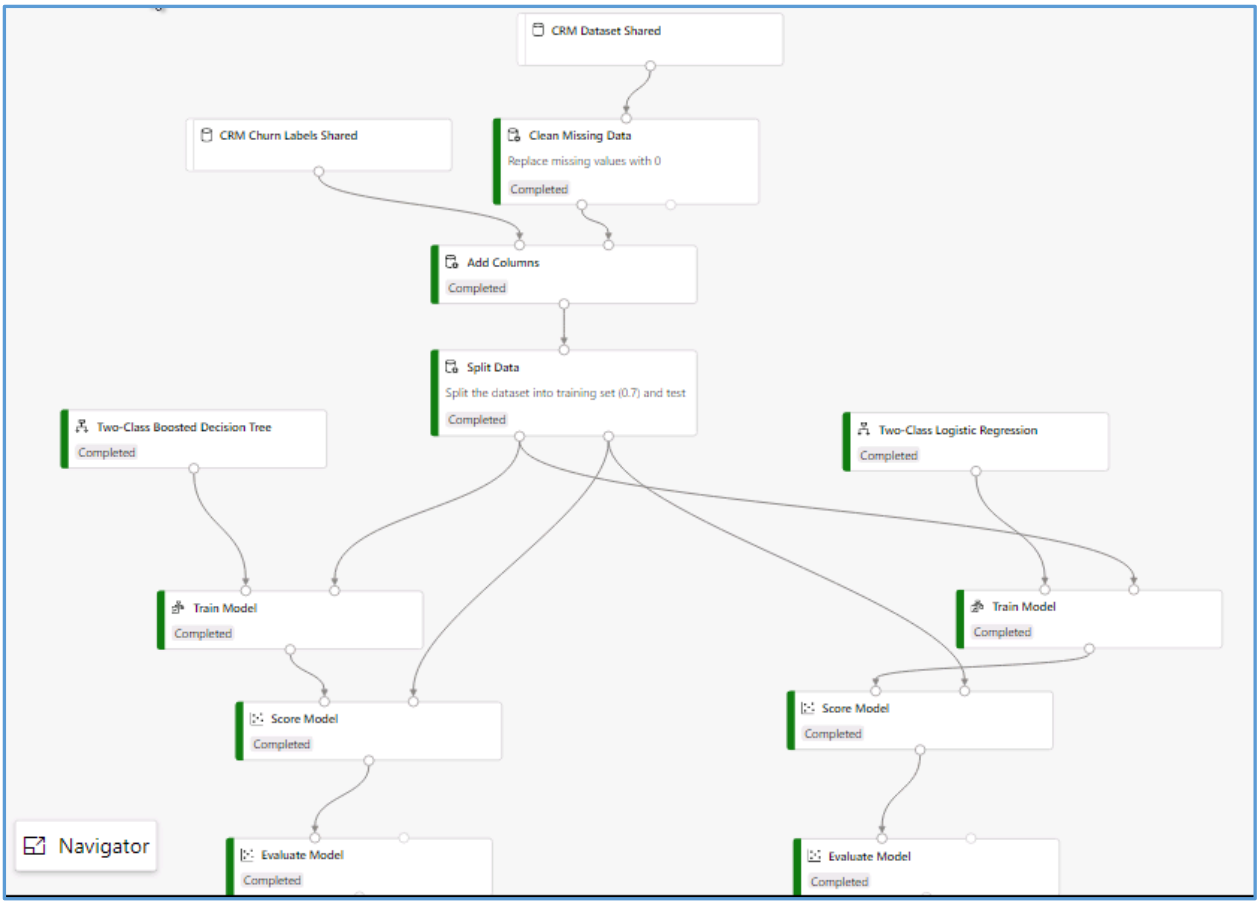
The goal is to compare the performance of two binary classifiers, two class boosted decision tree and two-class logistic regression for predicting customer churn.

Precision metric would be, used to evaluate the performance of these two algorithms.

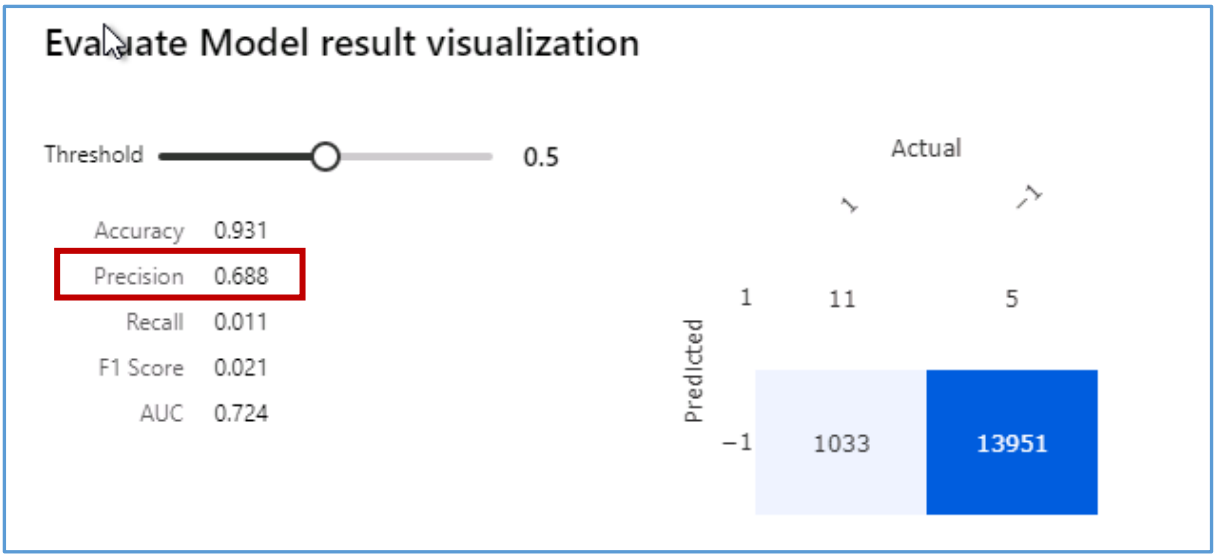
- Setup the pipeline and select Compute Target.[Scales automatically as per the experiment's need]
- Open Sample 5: Binary Classification – Customer Relationship Prediction from Designer Tab.



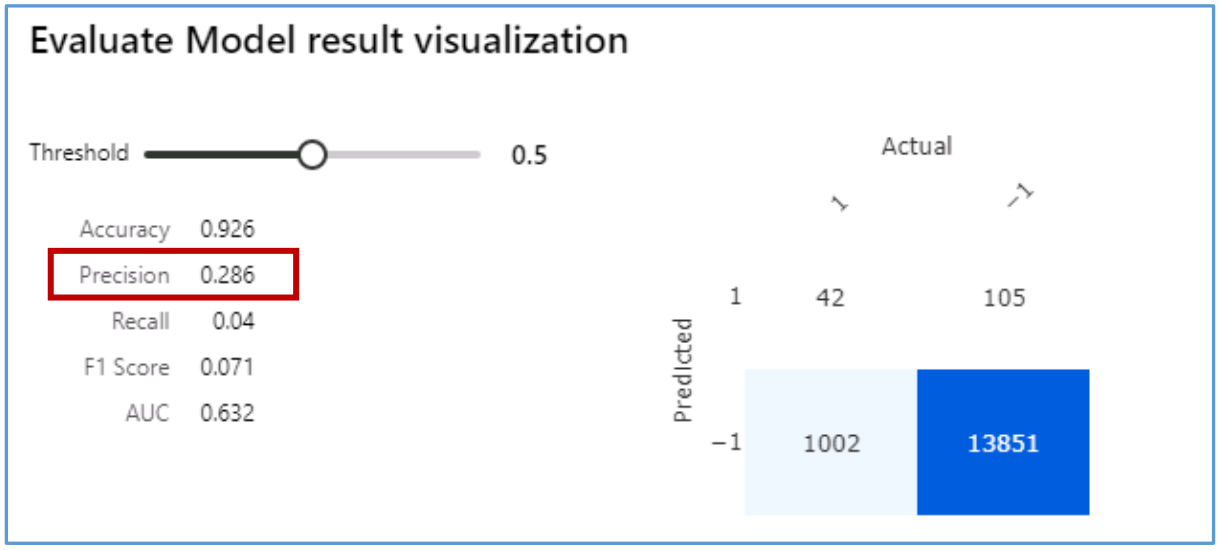
- From the right-hand-side of the pipeline, select the Two-Class Boosted Decision Tree module and then select the Delete Icon.
- From the right-hand-side of the pipeline, select the SMOTE module and then select the Delete Icon.
- From the right-hand-side of the pipeline, select the SMOTE module and then select the Delete Icon.
  a) Connect the Two-Class Logistic Regression module to the first input of the Train Model module
  b) Connect the first output of the Split Data module to the second input of the Train Model module
- Select Submit to open the Setup pipeline run editor.



Two Class Boosted Decision Tree



Two Class Logistic Regression



- Based on the primary performance metric, **precision**, it shows that the Two-Class Boosted Decision Tree algorithm **[Precision = 0.7]** outperforms the Two-Class Logistic Regression algorithm **[Precision = 0.3]**.

Chapter 6: Prelaunch Lab

Prelaunch your lab environment.

NOTE: When you click "Prelaunch Lab", we will begin preparing a lab environment for you. When it is time to access the lab environment, this ensures you will be able to start working. Once your lab is reserved, you can continue on to the next concepts.

Your lab has been, reserved! Please continue to the next concept.

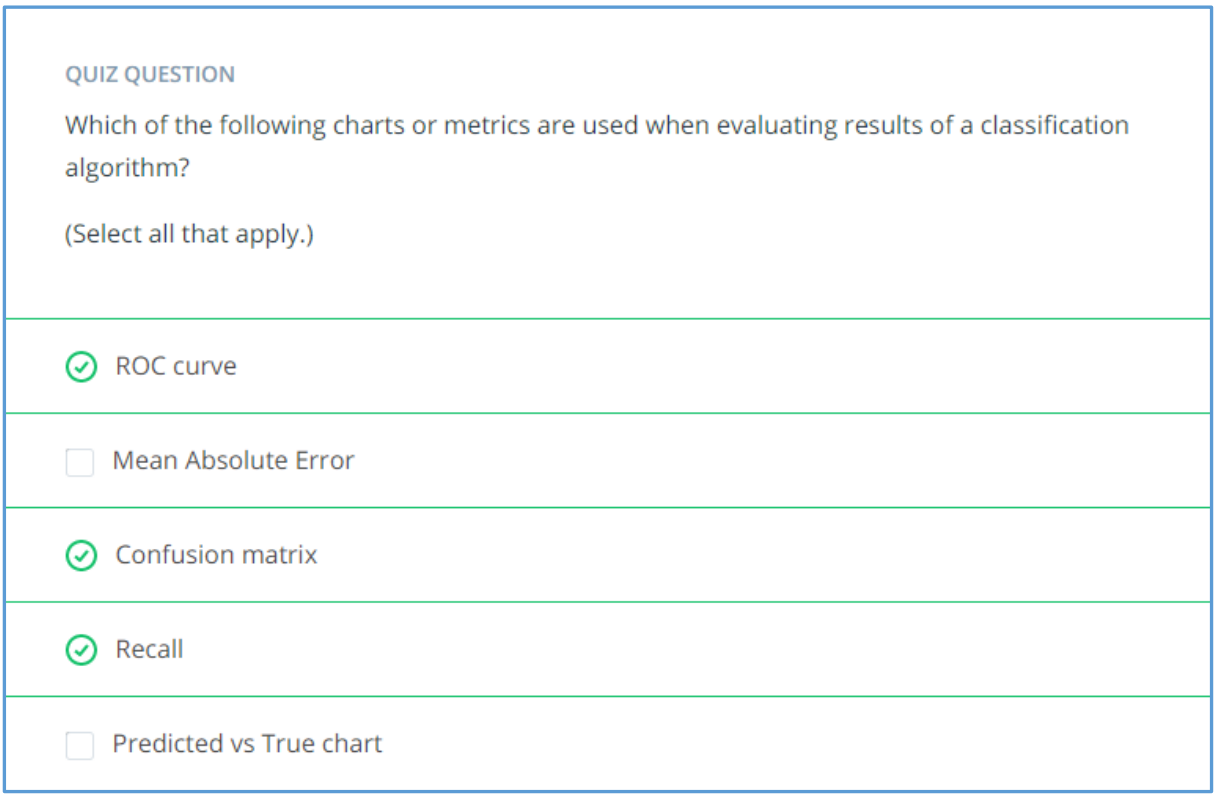
Chapter 7: Multi-Class Algorithms

Learn about Multi class classifier algorithm and the various hyperparameters used to tune the performance of these algorithms.

- **Multi Class Logistic Regression Algorithm** – Logistic regression is, used to predict the probability of an outcome & it is popular for classification tasks. Two key parameters to configure this algorithm are :
  a) **Optimization Tolerance** – Controls when to stop the iterations. If the improvement between the iterations is less than the threshold, the algorithm stops and returns the current model.

b) **Regularization Weight** – This is a method used for preventing overfitting by penalizing models with extreme coefficients values. The regularization weight controls how much to penalize the models at each iterations.

- **Multi Class Neural Network** – Example of a multi class neural network used in multi class classification includes the Input Layer one Hidden Layer and an Output Layer. The relationship between input and output is learned from training the Neural network on Input data. Three key parameters used to configure the algorithm are:
  a) **Number of Hidden Nodes** – Helps in customizing the number of hidden nodes and the neural network
  b) **Learning Rate** – This controls the size of the step taken at each iteration before correction.
  c) **Number of Learning Iterations** – The maximum number of times the algorithm should process the training cases.
- **Multi Class Decision Forest** – This is an ensemble of decision trees. It works by building multiple decision trees and then voting on the most popular output class. Five key parameters used to configure the algorithm are-
  a) **Resampling method** – This controls the method used to create the individual trees.
  b) **Number of decision trees** – Maximum number of decision trees that can be, created in the ensemble.
  c) **Maximum depth of the decision trees** – Number to limit the depth of any decision tree.
  d) **Number of random splits per node** – Number of splits to use while building each node of the tree.
  e) **Minimum number of samples per leaf node** – Minimum number of cases that are required to create any terminal node in a tree.



Chapter 8: Lab - Multi-Class Classifiers Performance

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.

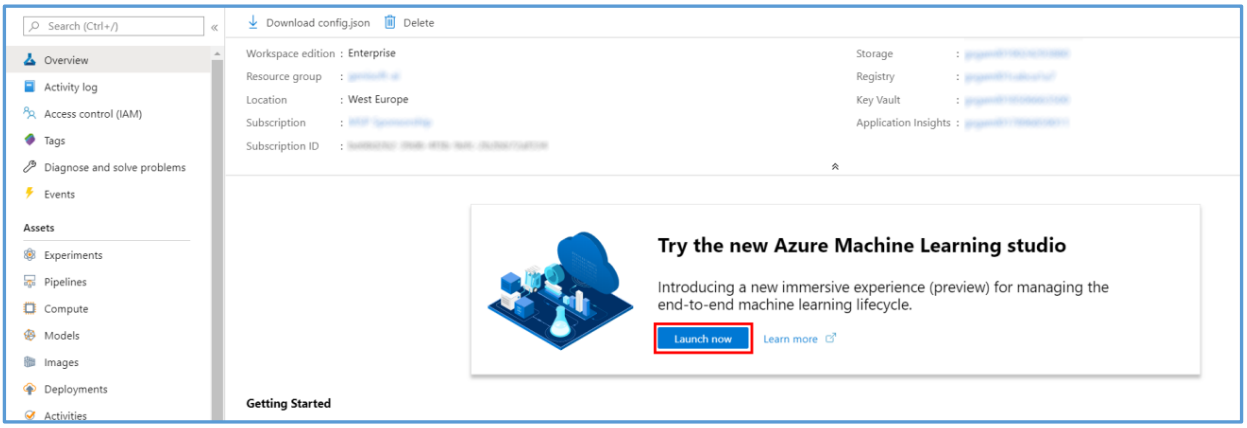
In this lab, we will be compare the performance of two different multiclass classification approaches: `Two-Class Support Vector Machine` used with `One-vs-All Multiclass` module vs `Multiclass Decision Forest`. We will apply the two approaches for the letter recognition problem and compare their performance. We will do all of this from the Azure Machine Learning designer without writing a single line of code.

Exercise 1: Create Training Pipeline

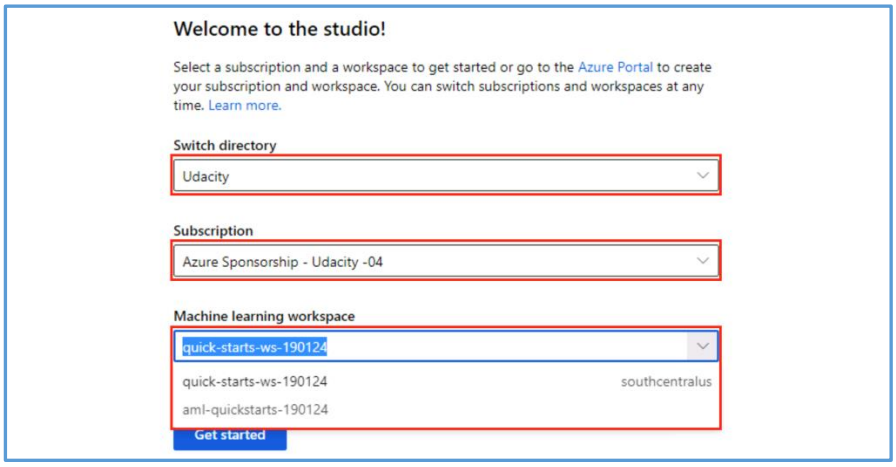
Task 1: Open Sample 12: Multiclass Classification - Letter Recognition

1. In Azure portal, open the available machine learning workspace.

2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

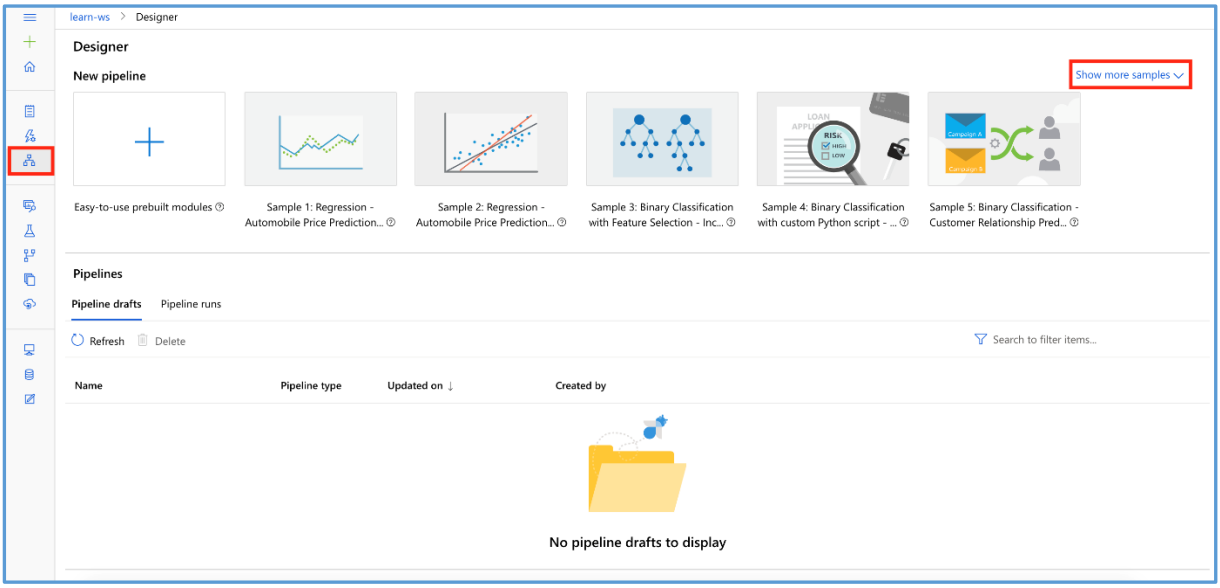


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:

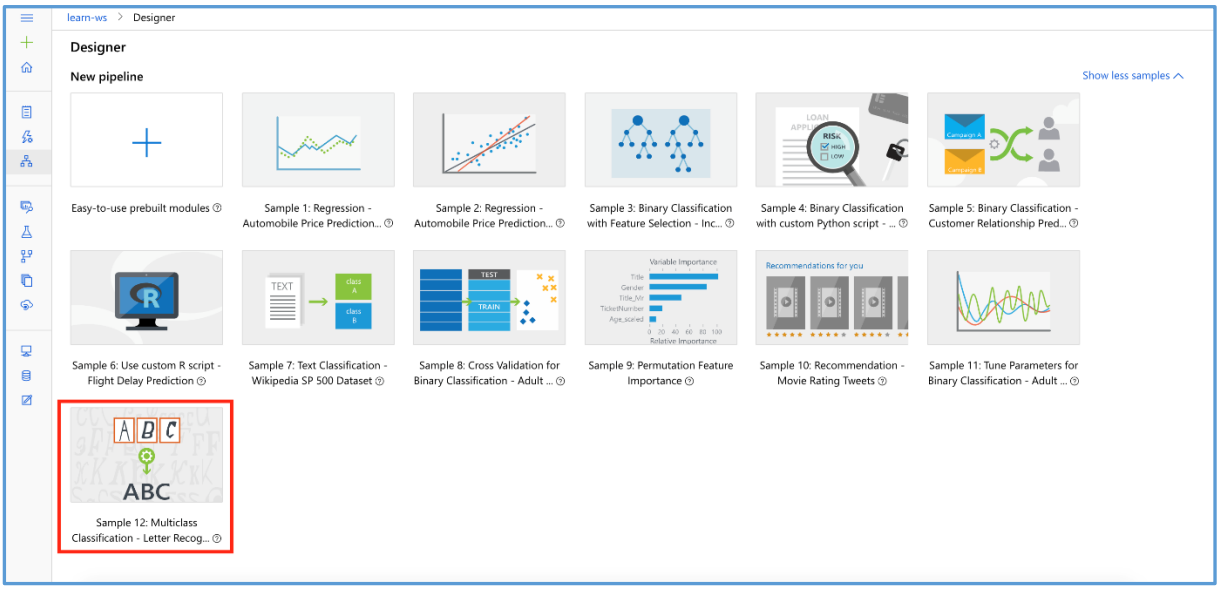


For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

4. From the studio, select **Designer, Show more samples**.

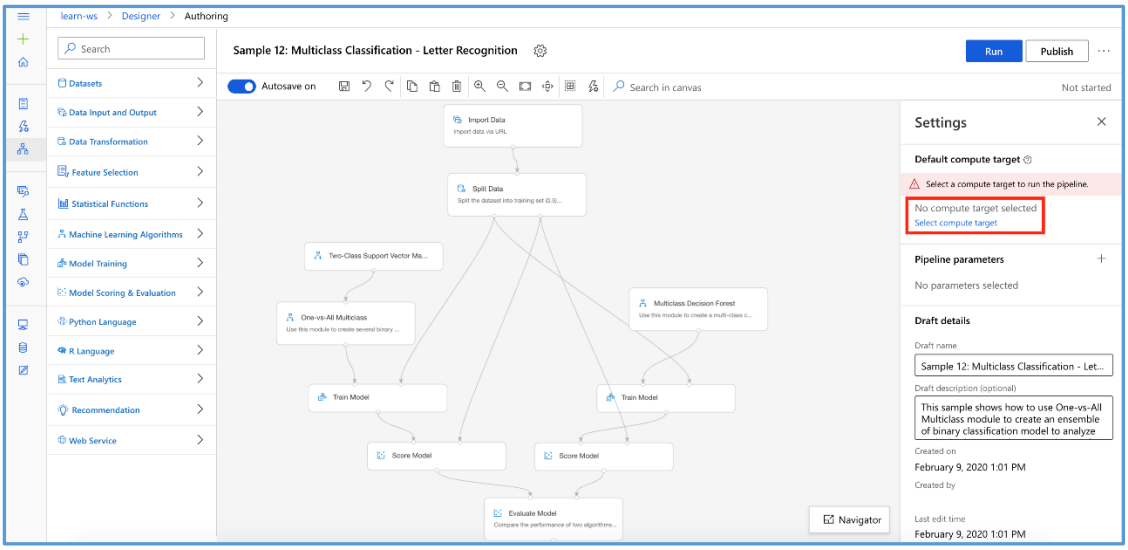


5. Select **Sample 12: Multiclass Classification - Letter Recognition**.



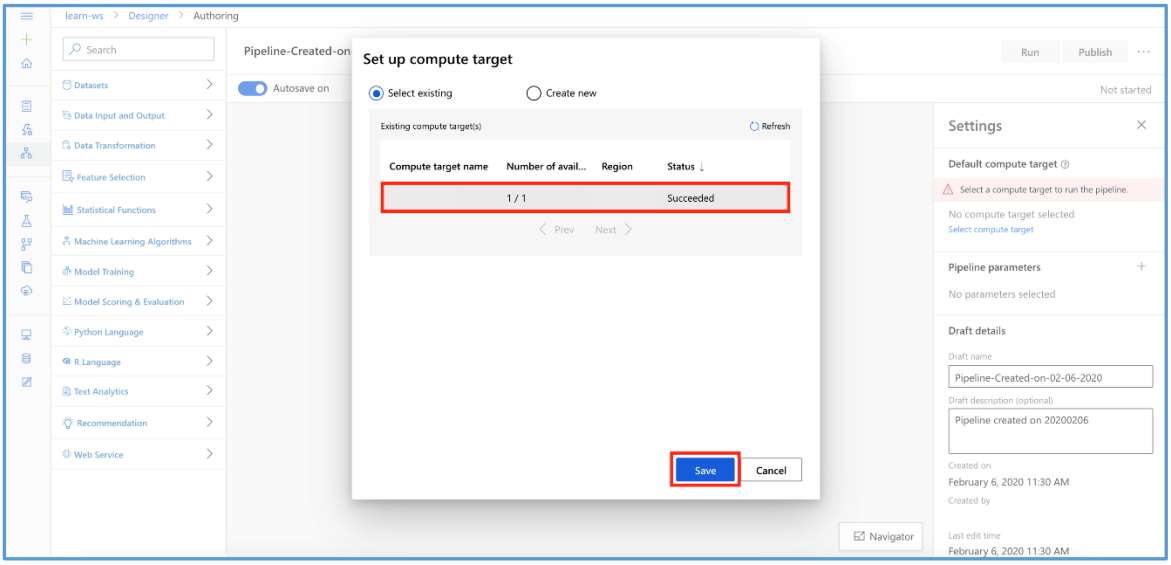
Task 2: Setup Compute Target

1. In the settings panel on the right, select **Select compute target**.



2. In the Set up compute target editor, select the available compute, and then select **Save**.

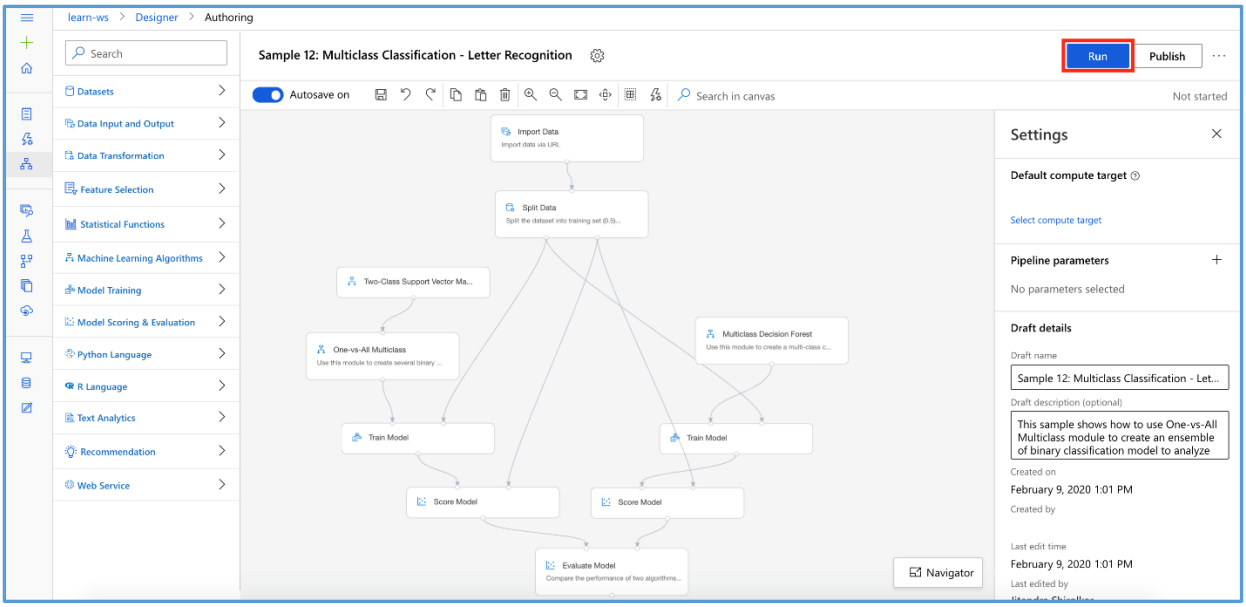
Note: If you are facing difficulties in accessing pop-up windows or buttons in the user interface, please refer to the Help section in the lab environment.



Exercise 2: Submit Training Pipeline

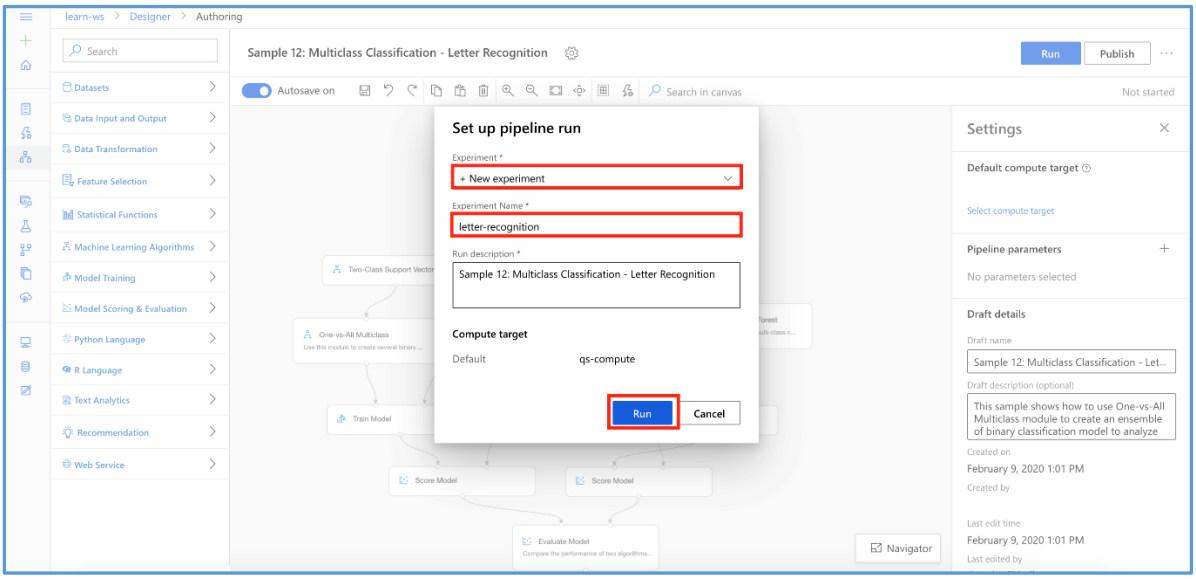
Task 1: Create Experiment and Submit Pipeline

1. Select **Submit** to open the Setup pipeline run editor.



Please note that the button name in the UI is, changed from **Run** to **Submit**.

2. In the Setup pipeline run editor, select **Experiment, Create new** and provide New experiment name: **letter-recognition**, and then select **Submit**.



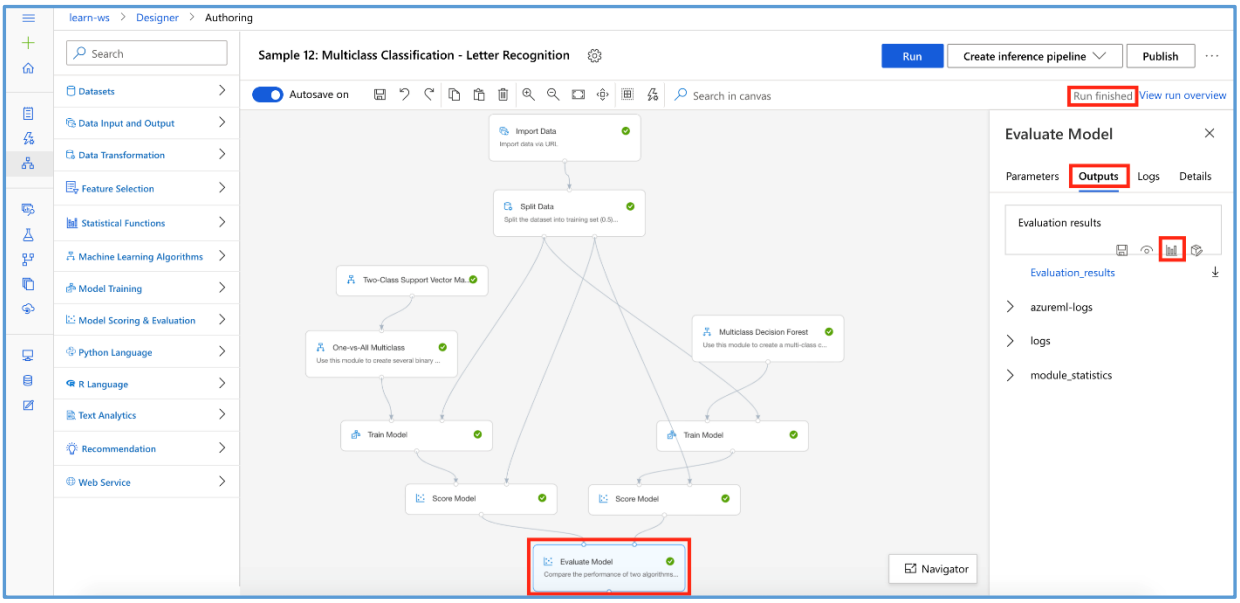
3. Wait for pipeline run to complete. It will take around **10 minutes** to complete the run.

4. While you wait for the model training to complete, you can learn more about the One-vs-All Multiclass module used in this lab by selecting One-vs-All Multiclass.

Exercise 3: Compare Model Performance

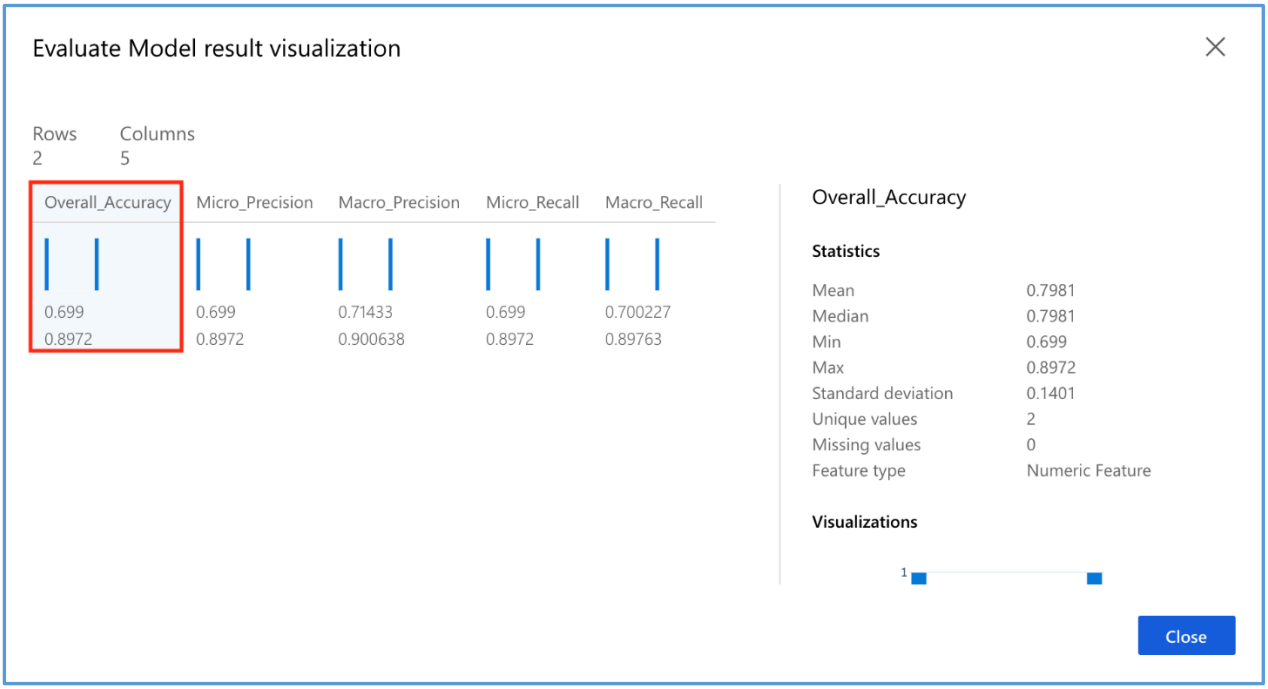
Task 1: Open Evaluation Results

1. Select **Evaluate Model, Outputs, Visualize** to open the Evaluate Model result visualization dialog.



Task 2: Compare Performance Metrics

1. Select the regression performance metric **Overall_Accuracy** and compare performance of the two algorithms: Two-Class Support Vector Machine and Multiclass Decision Forest.



Task 3: Conclusion

1. The Two-Class Support Vector Machine algorithm is extended for multiclass classification problem by using the One-vs-All Multiclass module.

2. As you can observe that the native multiclass algorithm Multiclass Decision Forest outperforms the Two-Class Support Vector Machine across all key performance metrics.

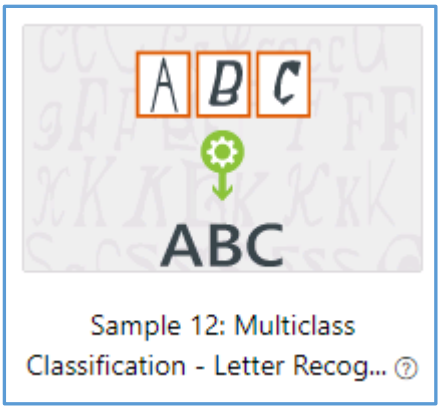
3. One recommendation for next steps is to increase the Number of iterations parameter for the Two-Class Support Vector Machine module to an higher value like **100** and observe its impact on the performance metrics.

Next Steps

Congratulations! You have trained and compared performance of two different multiclass classification machine-learning models. You can continue to experiment in the environment but are free to close the lab environment tab and return to the Udacity portal to continue with the lesson.

Chapter 9: Walkthrough - Multi-Class Classifiers Performance

- The goal is to compare the performance of two different Multi Class classification approaches two class, support vector machines used with one versus all multi class module versus multi class, decision forest. This would be, applied for letter recognition problem & compare their performance.
- Overall accuracy would be, used as the primary metric.
- Open Sample 12: Multi Class Classification – Letter recognition problem from Designer Tab.
- Setup the pipeline and select Compute Target.[Scales automatically as per the experiment's need]



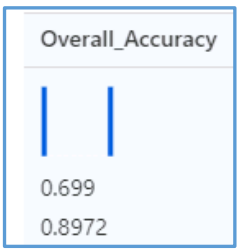
- Two algorithms used for comparison - The Two-Class Support Vector Machine algorithm is, extended for multiclass classification problem by using the One-vs-All Multiclass module vs. Multiclass Decision Forest
- Select Submit to open the Setup pipeline run editor.

Conclusion

1. The Two-Class Support Vector Machine algorithm is, extended for multiclass classification problem by using the One-vs-All Multiclass module.

2. As you can observe that, the native multiclass algorithm Multiclass Decision Forest outperforms the Two-Class Support Vector Machine across all key performance metrics.

3. One recommendation for next steps is to increase the Number of iterations parameter for the Two-Class Support Vector Machine module to a higher value like **100** and observe its impact on the performance metrics.



Chapter 10: Prelaunch Lab

Prelaunch your lab environment.

NOTE: When you click "Prelaunch Lab", we will begin preparing a lab environment for you. When it is time to access the lab environment, this ensures you will be able to start working. Once your lab is reserved, you can continue on to the next concepts.

Your lab has been, reserved! Please continue to the next concept.

Chapter 11: Lab: Train a Classifier Using Automated Machine Learning

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.

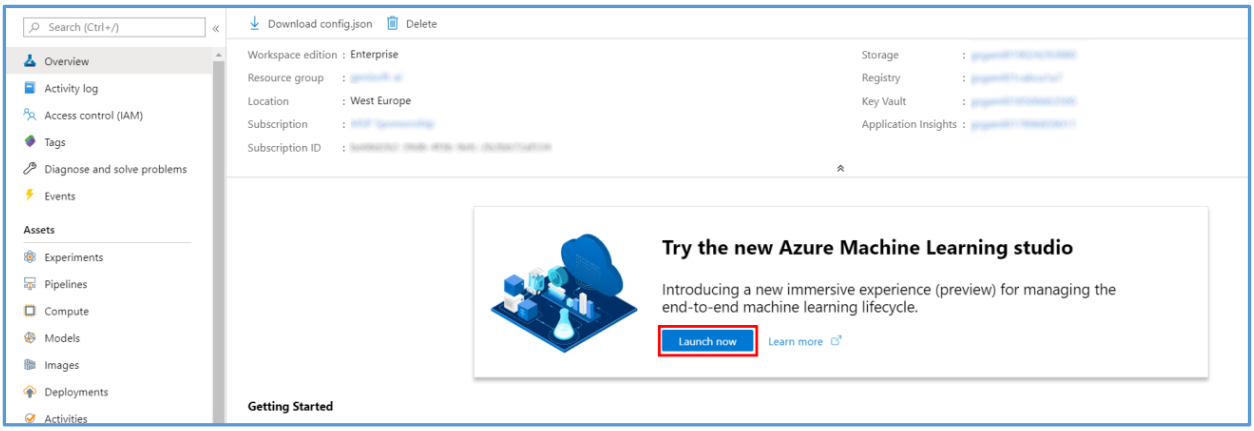
In this lab, we will use Automated Machine Learning to find the best performing binary classification model for predicting customer churn. We will do all of this from the Azure Machine Learning Studio without writing a single line of code.

Exercise 1: Register Dataset with Azure Machine Learning studio

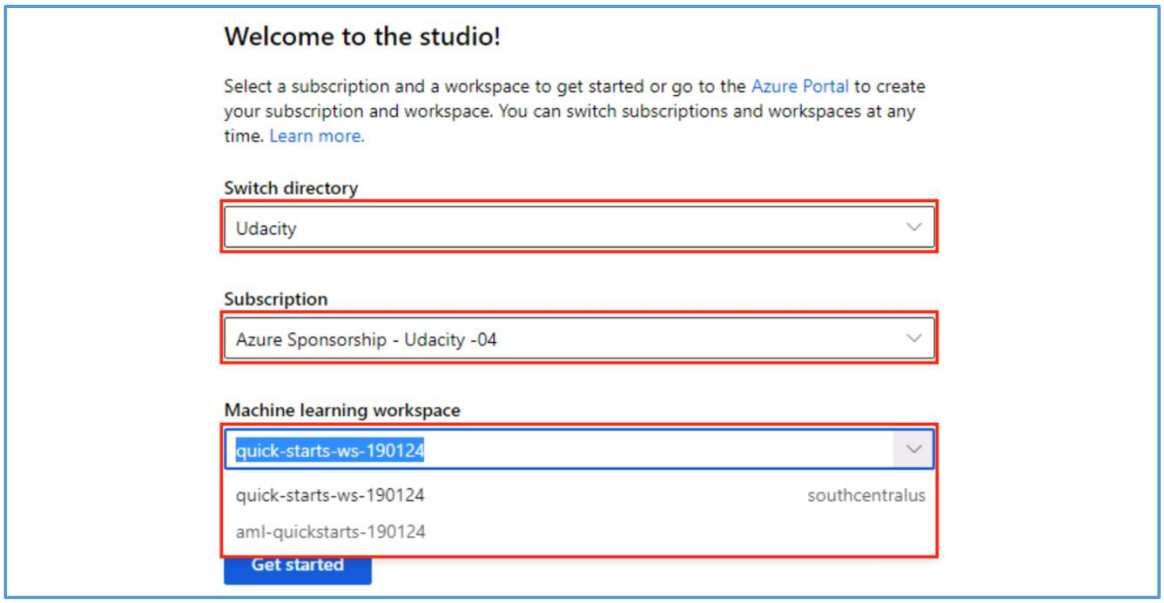
Task 1: Upload Dataset

1. In Azure portal, open the available machine learning workspace.

2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

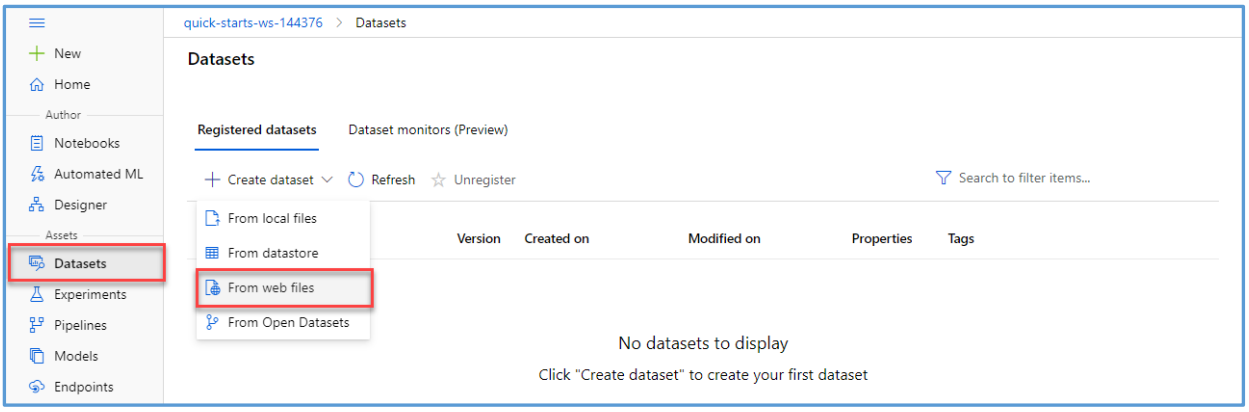


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it does not matter which) and then click **Get started**.

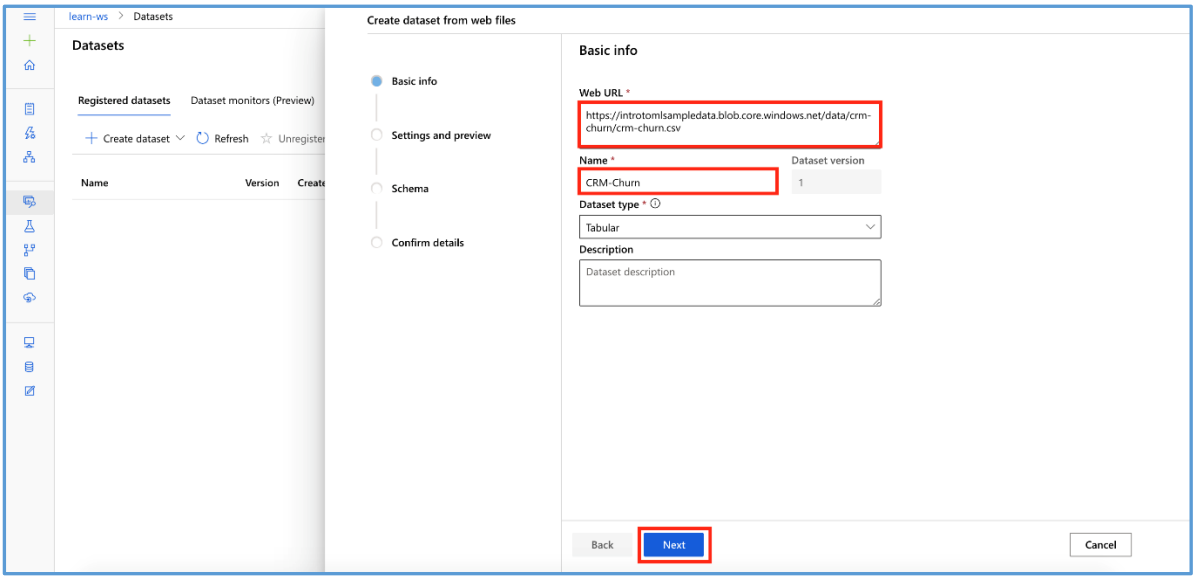
4. From the studio, select **Datasets, + Create dataset, from web files**. This will open the Create dataset from web files dialog on the right.



5. In the Web URL field provide the following URL for the training data file:

```
https://introtomlsampledata.blob.core.windows.net/data/crm-churn/crm-churn.csv
```

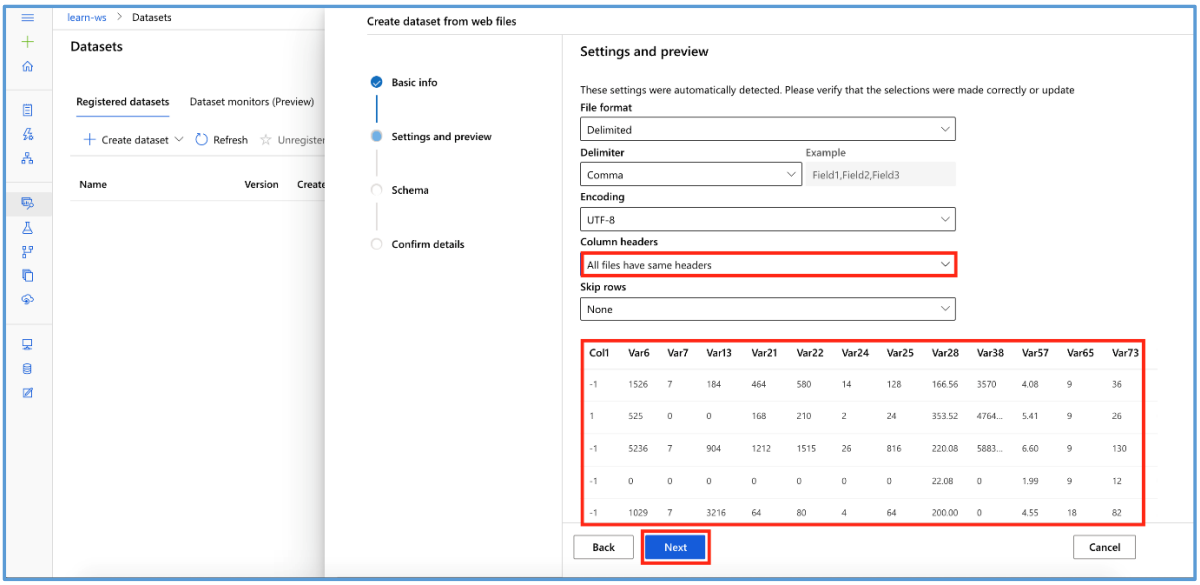
6. Provide `CRM-Churn` as the Name, leave the remaining values at their defaults and select **Next**.



Task 2: Preview Dataset

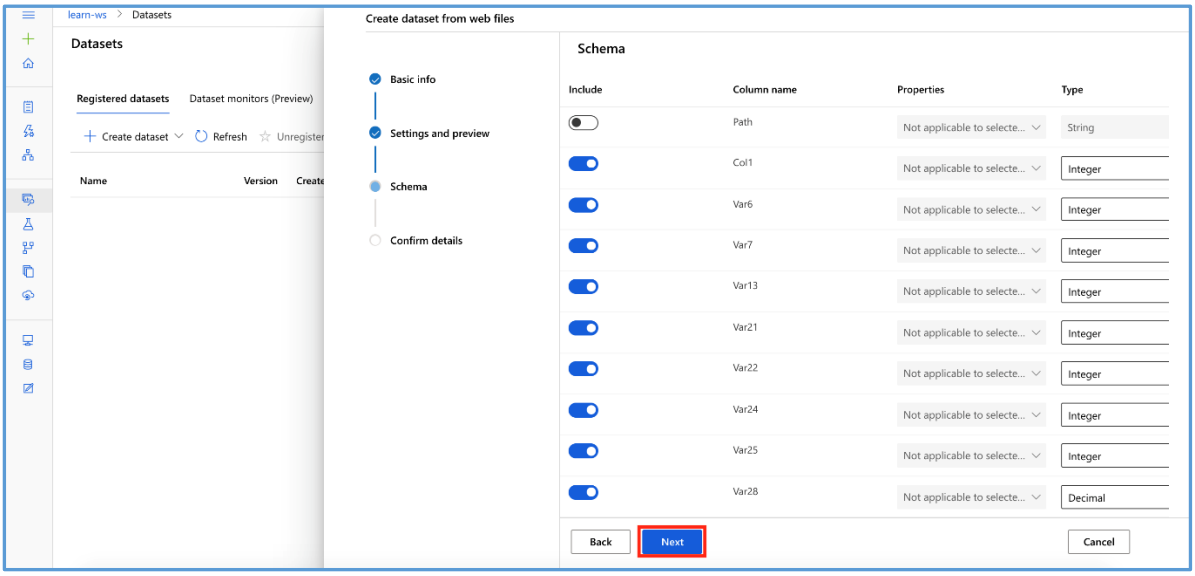
1. On the Settings and preview panel, set the column headers drop down to `All files have same headers`.

2. Review the dataset and then select **Next**



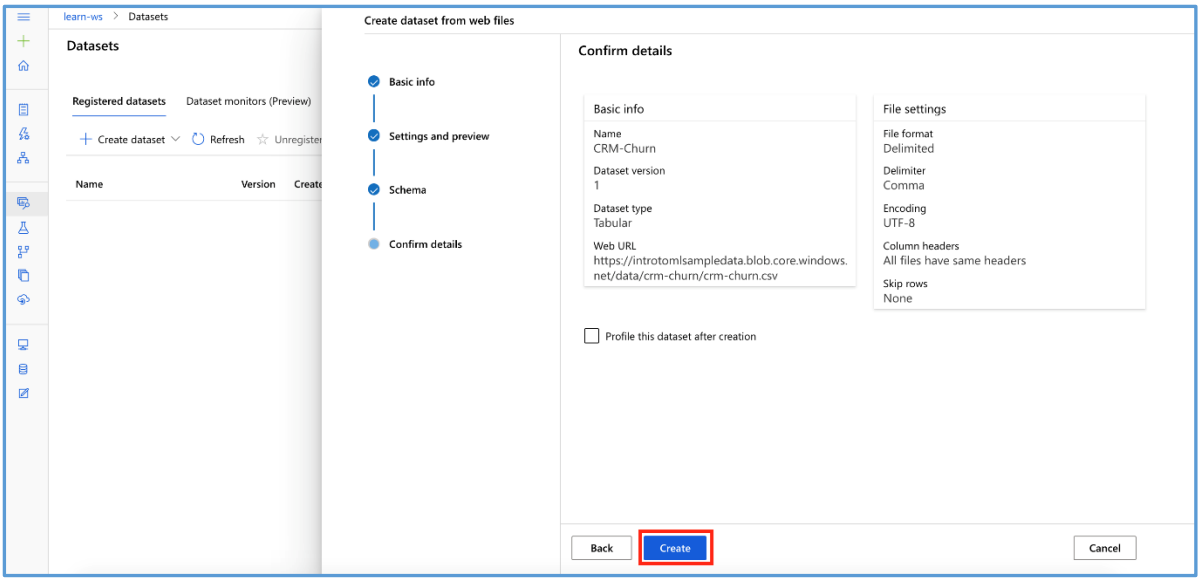
Task 3: Select Columns

1. Keep the default selections, and select **Next**



Task 4: Create Dataset

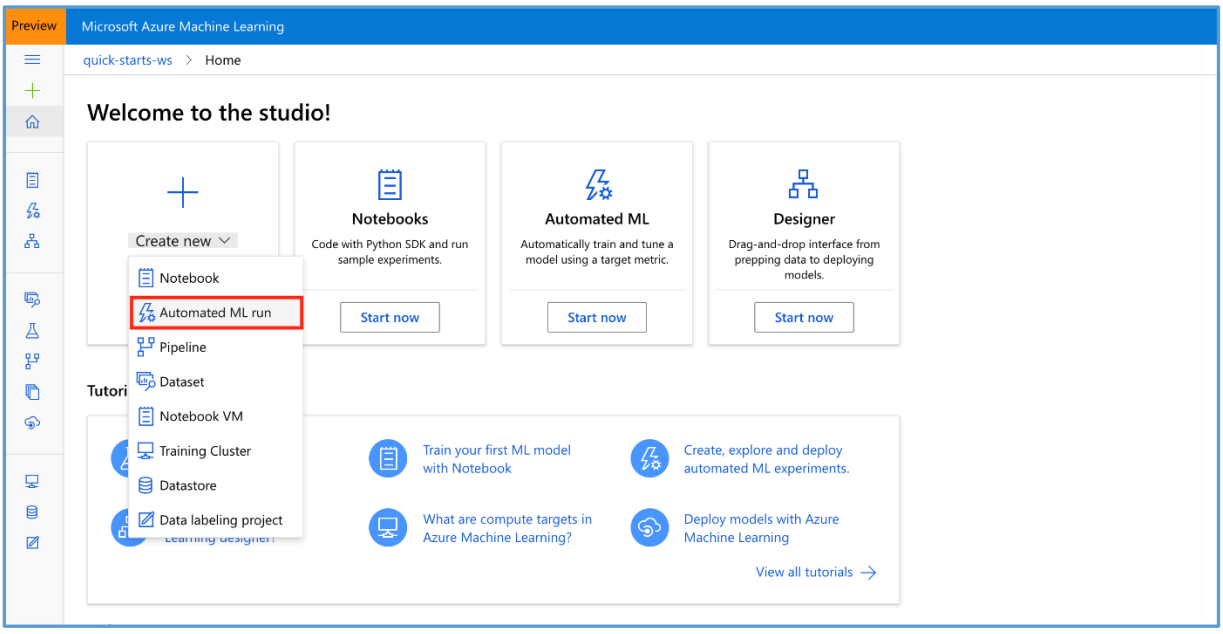
1. Confirm the dataset details and select **Create**



Exercise 2: Setup New Automated Machine Learning Experiment

Task 1: Create New Automated Machine Learning Experiment

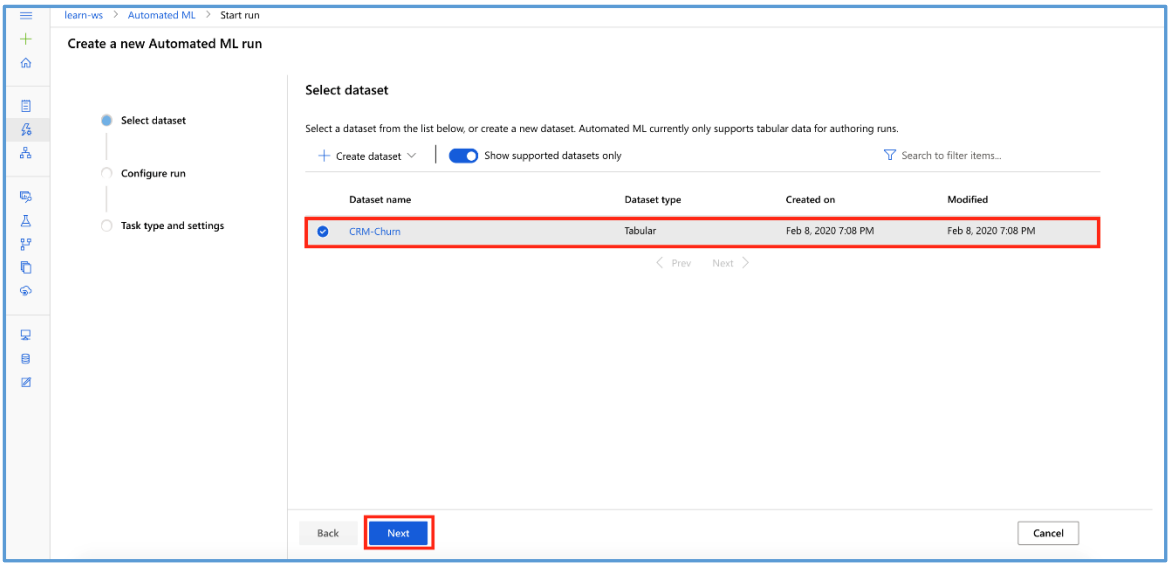
1. From the studio home, select **Create new, Automated ML run**



2. This will open a Create a new automated machine learning experiment page

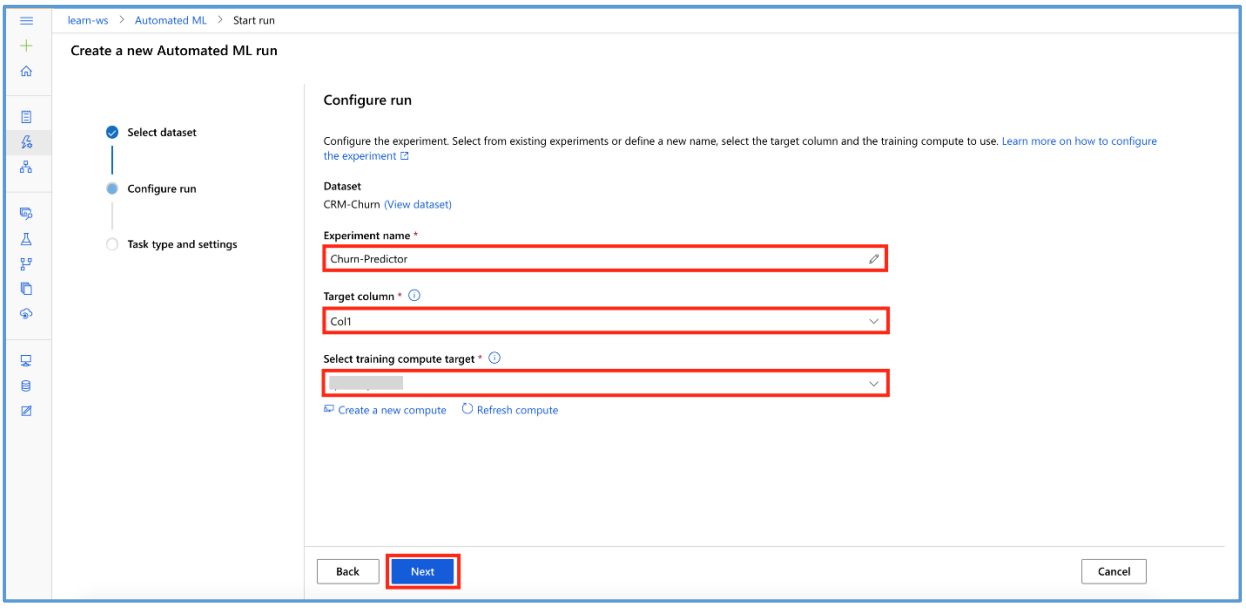
Task 2: Select Training Data

1. Select the dataset CRM-Churn and then select **Next**



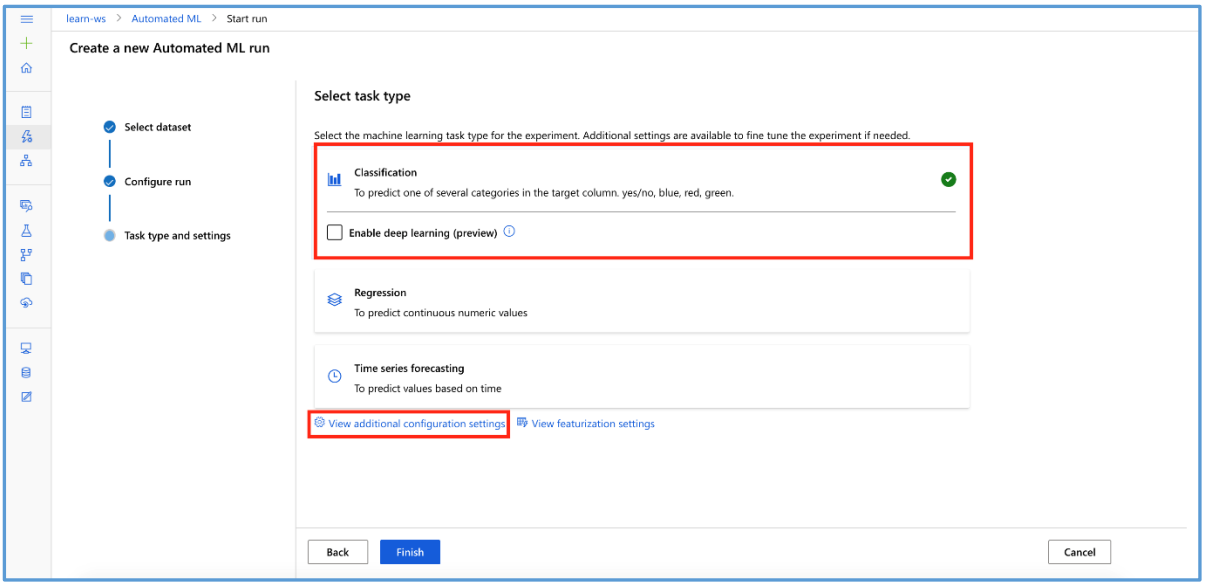
Task 3: Create a new Automated ML run

1. Provide an experiment name: **Churn-Predictor**
2. Select target column: **Col1**
3. Select compute target: **select the available compute**
4. Select **Next**



Task 4: Setup Task type and Settings

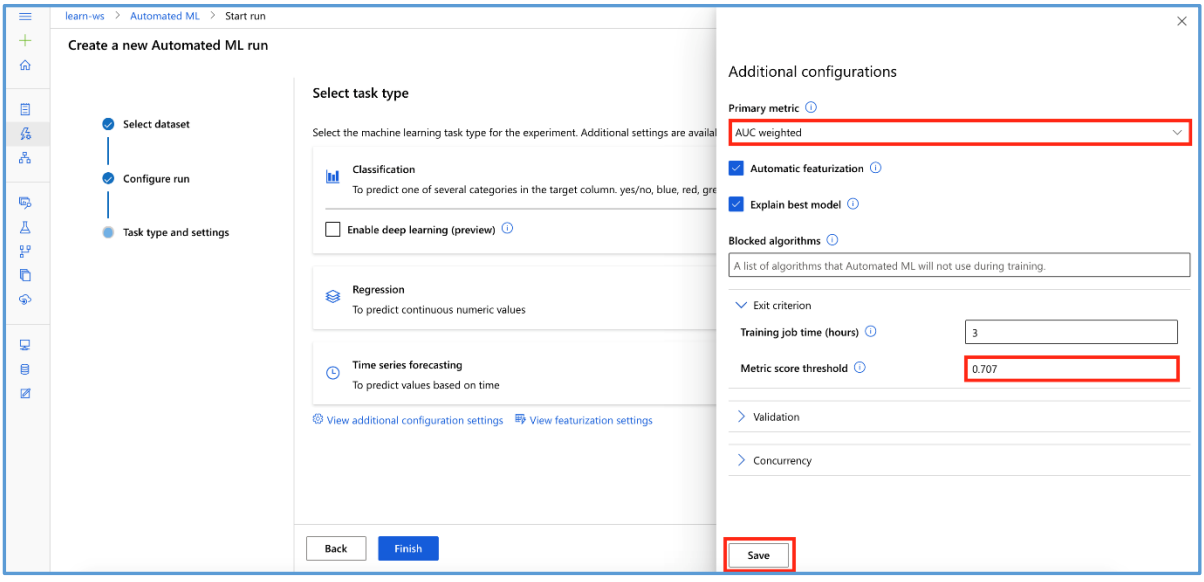
1. Select task type: **Classification**, and then select **View additional configuration settings**



2. This will open the Additional configurations dialog.

3. Provide the following information and then select **Save**

    1. Primary metric: **AUC weighted**
    2. Exit criteria, Metric score threshold: 0.707

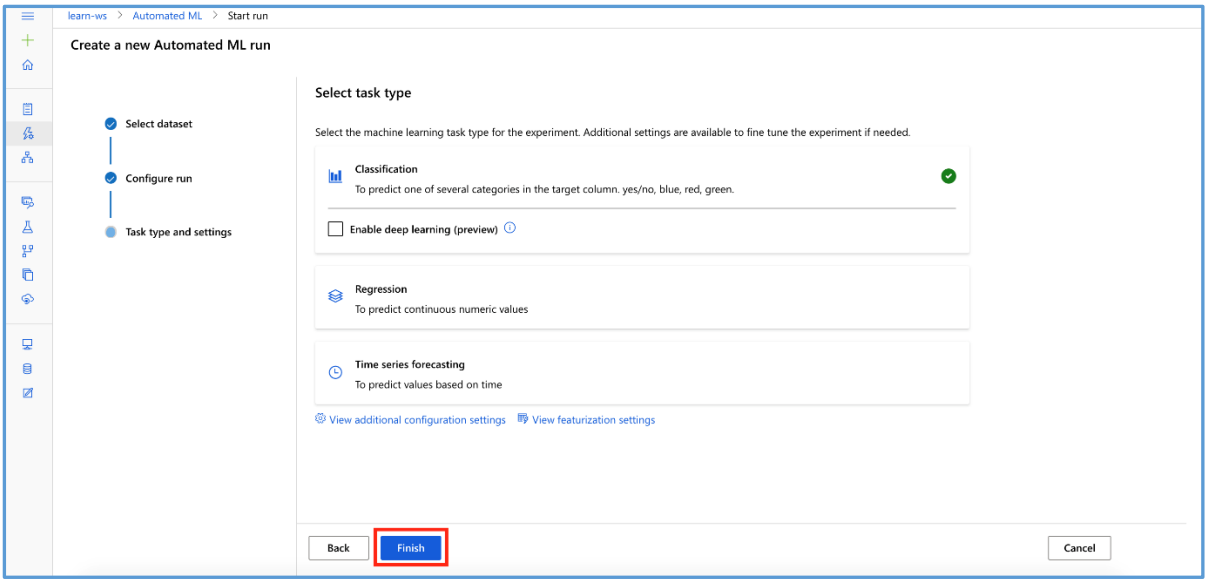


Note that we are setting a metric score threshold to limit the training time. In practice, for initial experiments, you will typically only set the training job time to allow AutoML to discover the best algorithm to use for your specific data.

Exercise 3: Start and Monitor Experiment

Task 1: Start Experiment

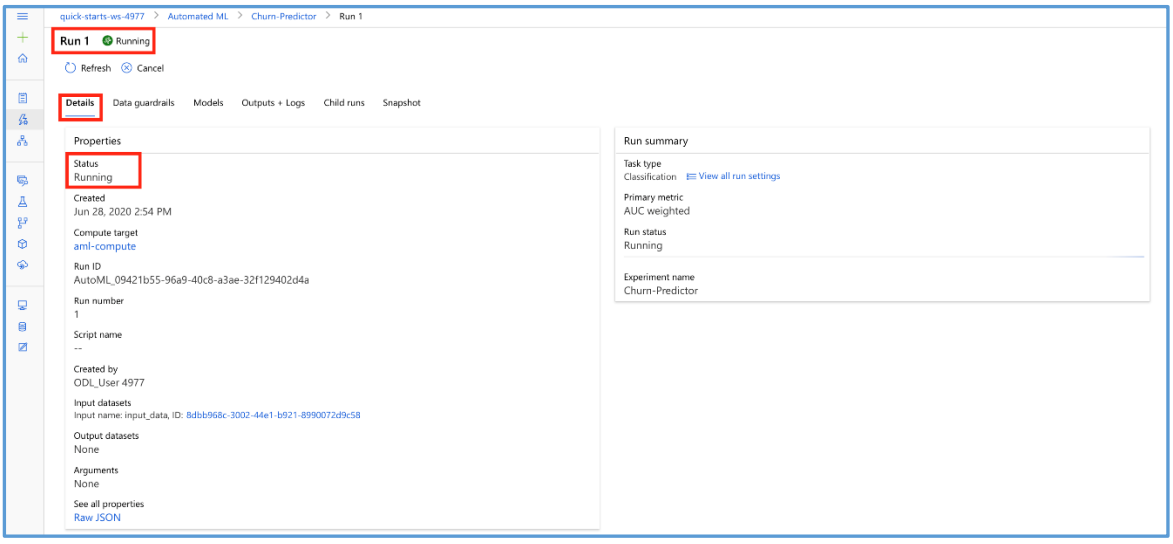
1. Select **Finish** to start running the experiment



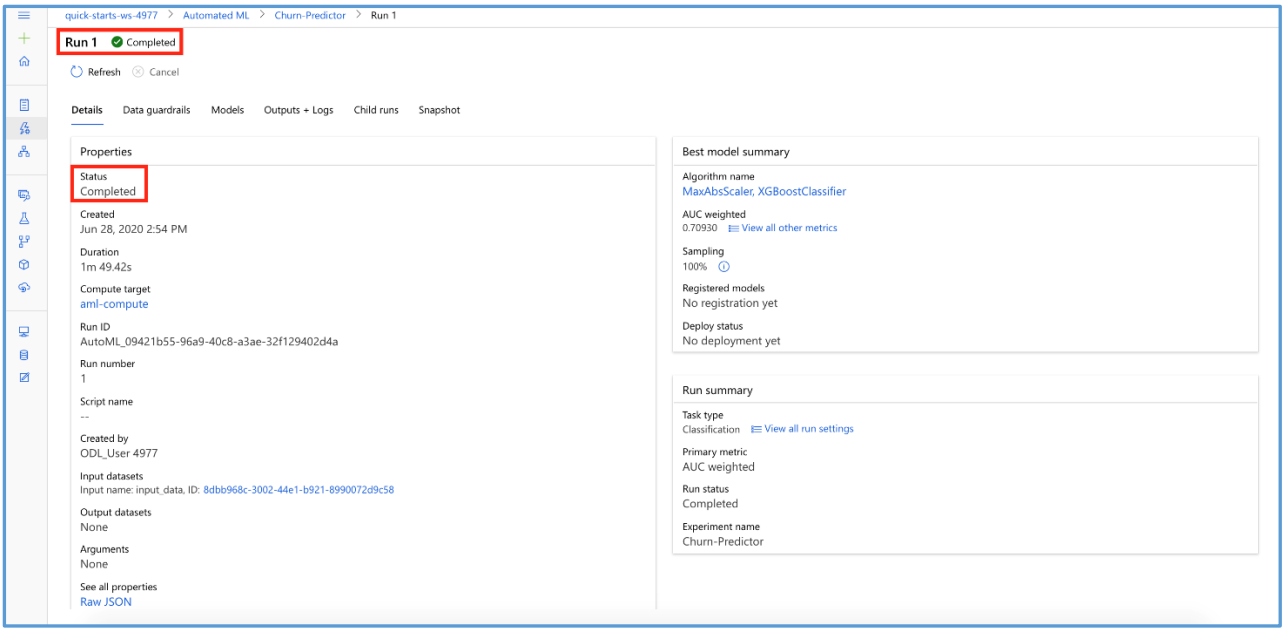
Task 2: Monitor Experiment

1. The experiment will run for about *5 min*

2. In the **Details** tab, observe the **run status** of the job.



3. Wait till the run status becomes **Completed**.

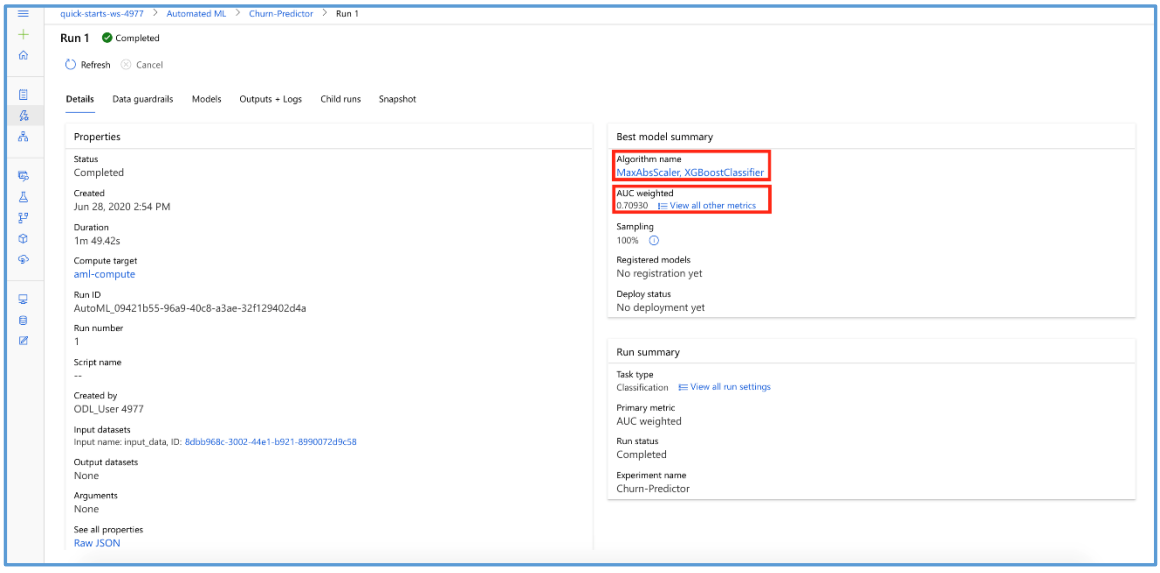


4. While you wait for the model training to complete, you can learn to view and understand the charts and metrics for your automated machine learning run by selecting Understand automated machine learning classification results.

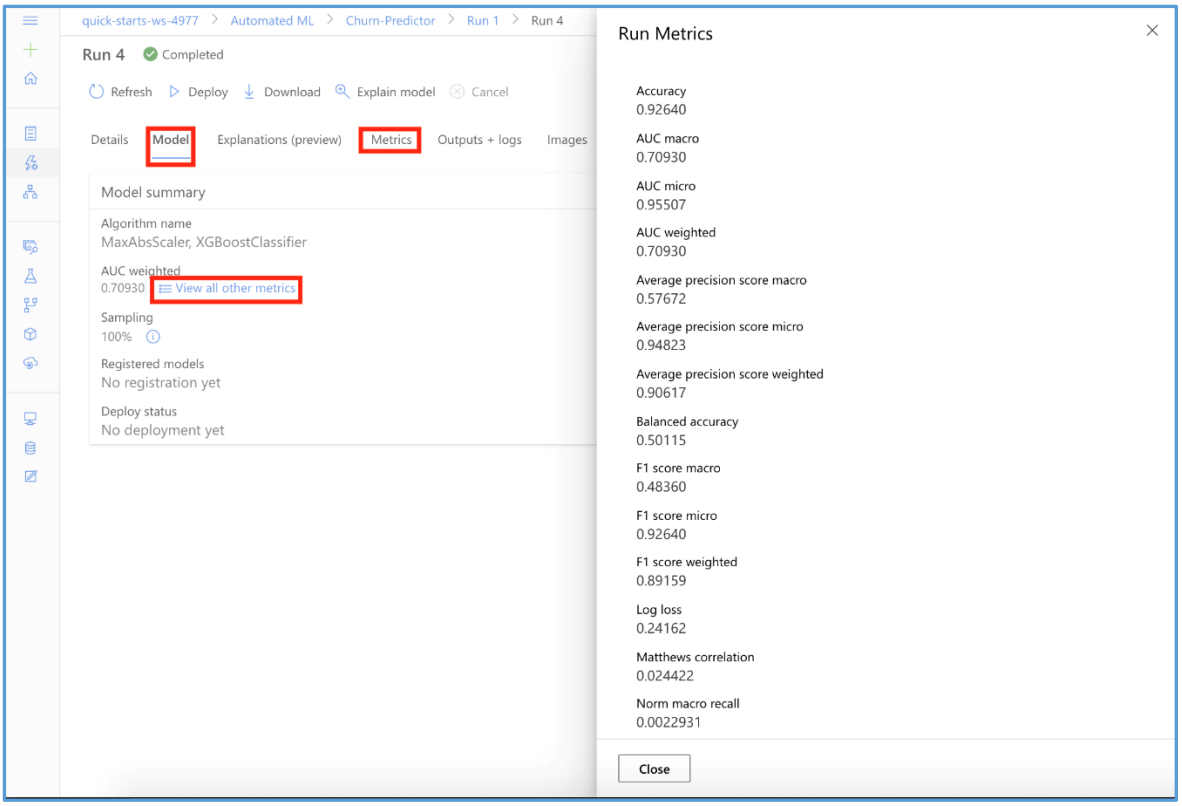
Exercise 4: Review Best Model's Performance

Task 1: Review Best Model Performance

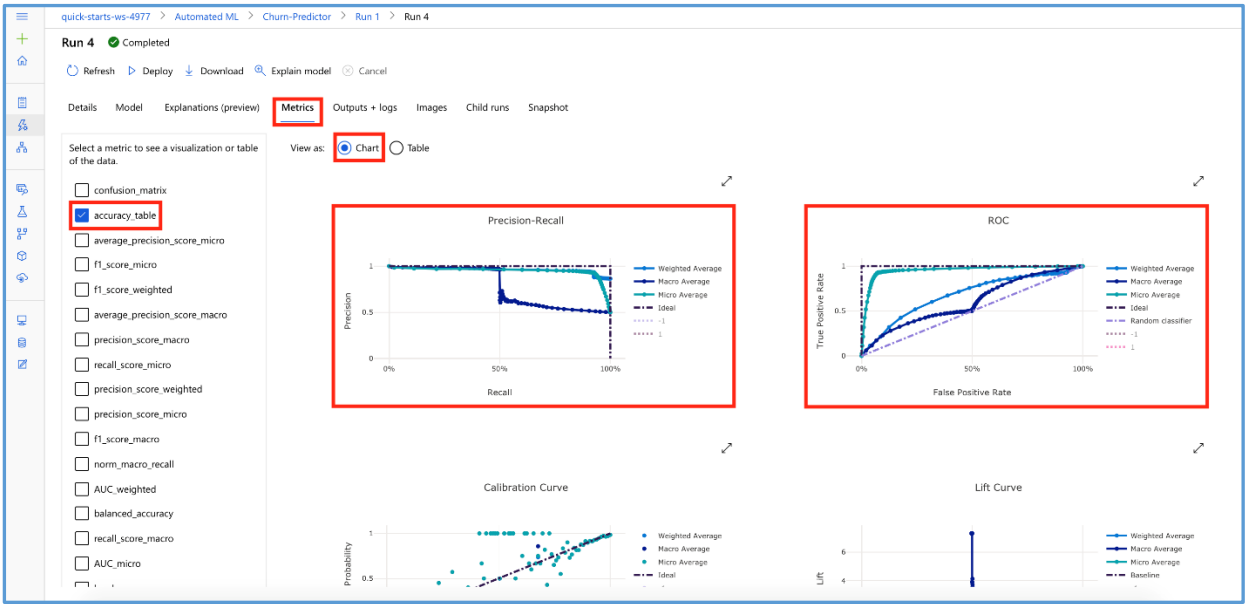
1. From the Details tab review the best model's Algorithm name and its corresponding AUC weighted score. Next, select the best model's **Algorithm name**



2. Select **View all other metrics** to review the various Run Metrics to evaluate the model performance. Next, select **Metrics**



3. Select **accuracy_table, Chart** to review the various model performance curves, such as Precision-Recall, ROC, Calibration curve, and Gain & Lift curves.

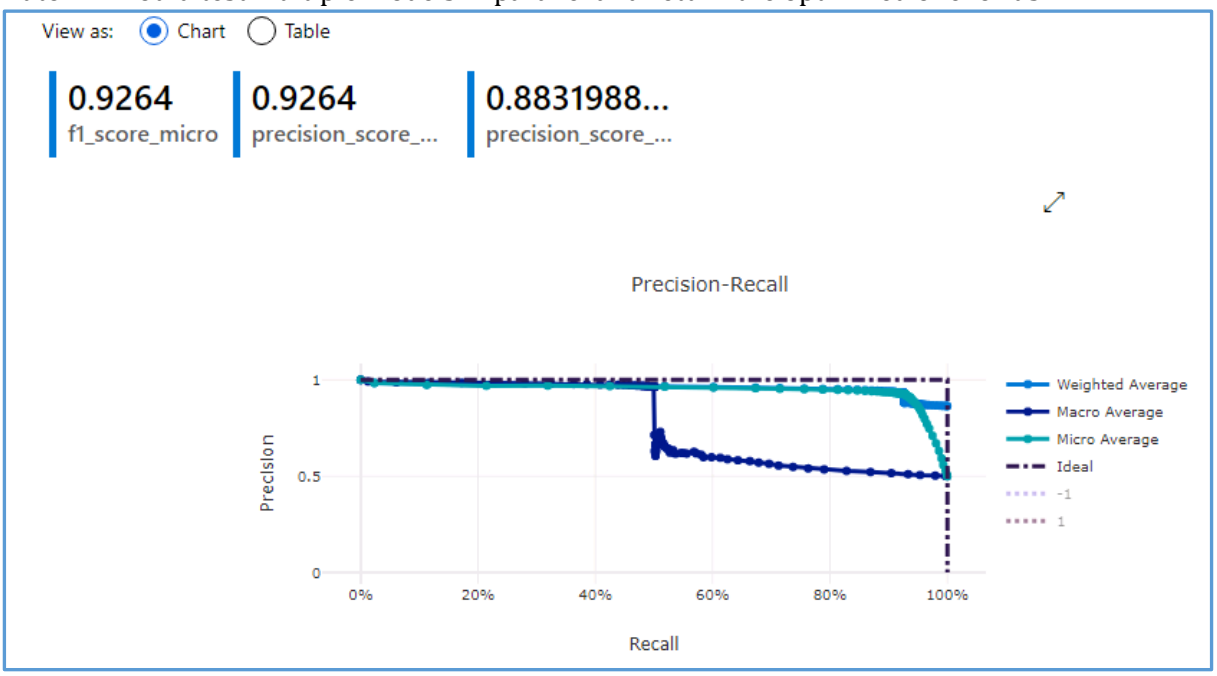


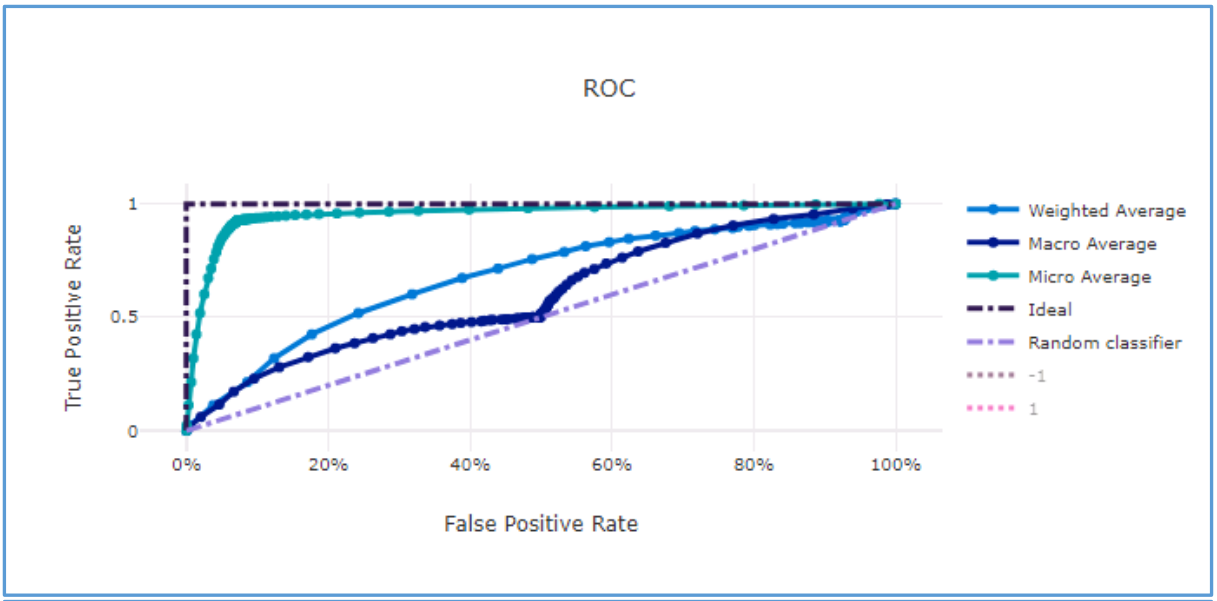
Next Steps

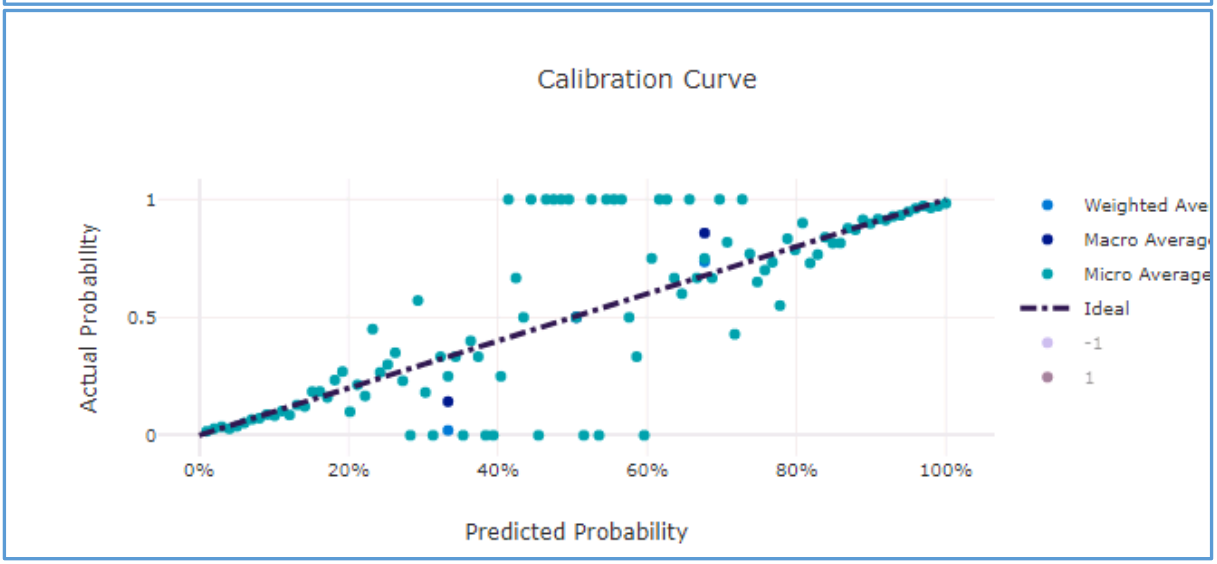
Congratulations! You have trained and evaluated a binary classification model using automated machine learning. You can continue to experiment in the environment but are free to close the lab environment tab and return to the Udacity portal to continue with the lesson.

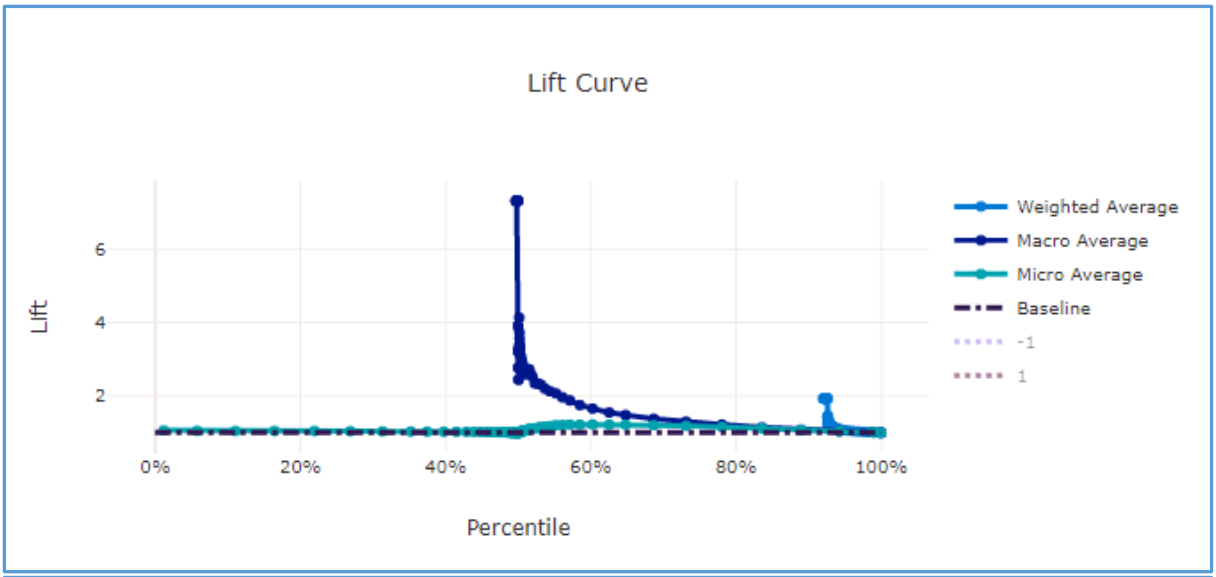
Chapter 12: Walkthrough: Train a Classifier Using Automated Machine Learning

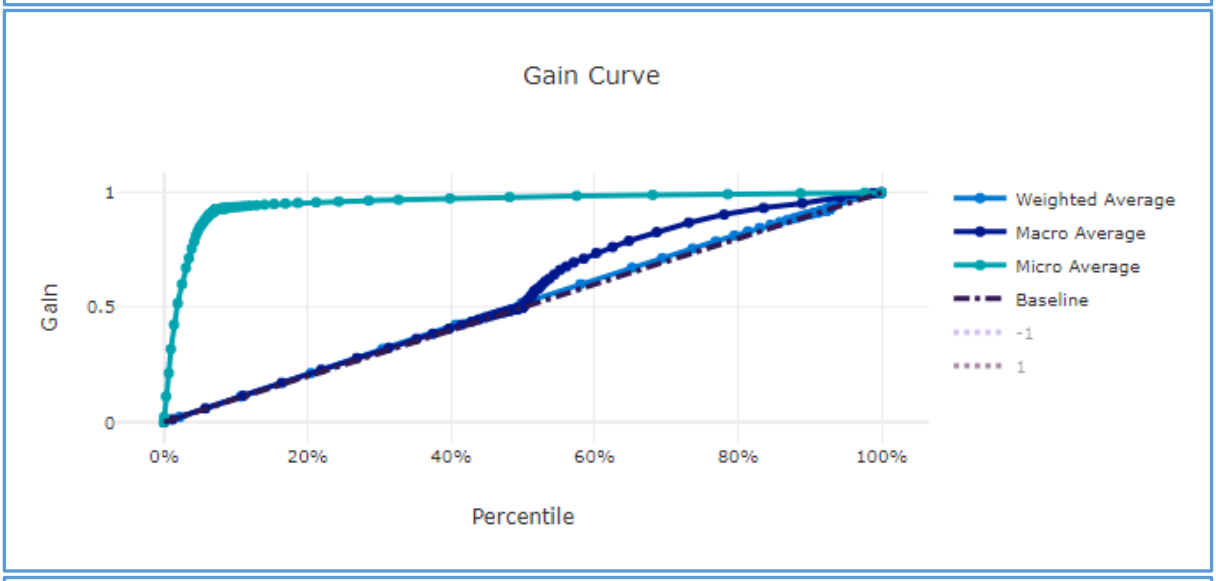
- We will use Auto ML to find the best performing binary classification model for predicting customer churn.
- Setup the dataset. Then initiate a new Automated ML run. Give an experiment name and choose the Target column. Select a Compute cluster, which would scale as per training needs.
- Task type would be "Classification"
- Then under "view additional configuration settings", we set "Primary metric" to "AUC weighted"
- Under Exit criteria set "Metric score threshold" at 0.707.
- Auto ML would test multiple models in parallel and return the optimized one for us.

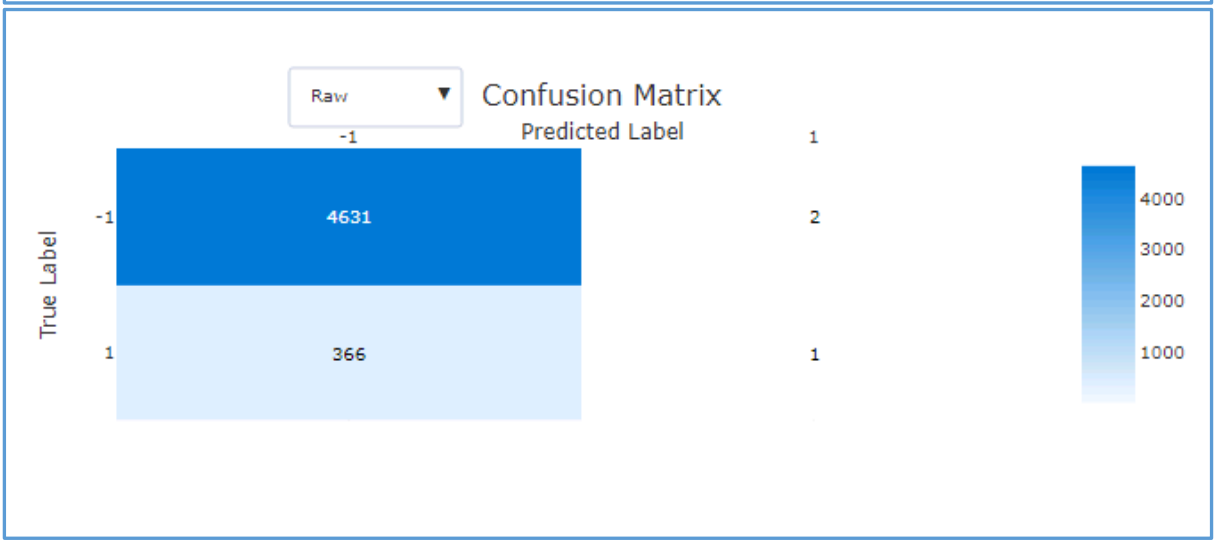



ROC
True Positive Rate
1
0.5
0
0%
20%
40%
60%
80%
100%
False Positive Rate
Weighted Average
Macro Average
Micro Average
Ideal
Random classifier
-1
1


Calibration Curve
Actual Probability
1
0.5
0
0%
20%
40%
60%
80%
100%
Predicted Probability
Weighted Ave
Macro Averag
Micro Average
Ideal
-1
1


Lift Curve
Lift
6
4
2
0%
20%
40%
60%
80%
100%
Percentile
Weighted Average
Macro Average
Micro Average
Baseline
-1
1


Gain Curve
Gain
1
0.5
0
0%
20%
40%
60%
80%
100%
Percentile
Weighted Average
Macro Average
Micro Average
Baseline
-1
1


Raw
Confusion Matrix
Predicted Label
True Label
-1
1
4631
366
4000
3000
2000
1000

Chapter 13: Prelaunch Lab

Prelaunch your lab environment.

NOTE: When you click "Prelaunch Lab", we will begin preparing a lab environment for you. When it is time to access the lab environment, this ensures you will be able to start working. Once your lab is reserved, you can continue on to the next concepts.

Your lab has been, reserved! Please continue to the next concept.

Chapter 14: Supervised Learning: Regression

The first type of *supervised learning* that we will look at is *Regression*. Again, the main distinguishing characteristic of regression is the type of output it produces:

*In a **regression** problem, the output is numerical or continuous.*

Introduction to Regression

Common types of regression problems include:

- ***Regression on tabular data:*** The data is available in the form of rows and columns, potentially originating from a wide variety of data sources.
- ***Regression on image or sound data:*** Training data consists of images/sounds whose numerical scores are already, known. Several steps need to be, performed during the preparation phase to transform images/sounds into numerical vectors accepted by the algorithms.
- ***Regression on text data:*** Training data consists of texts whose numerical scores are already, known. Several steps need to be, performed during the preparation phase to transform text into numerical vectors accepted by the algorithms.

Examples:

- Housing prices
- Customer churn
- Customer Lifetime Value
- Forecasting (time series)
- Anomaly detection

Categories of Algorithms

- **Linear Regression – Linear relationship between independent variables and a numeric outcome or dependent variable.**
- **Approaches – Two approaches to measure error and fit the regression line**
  a) **Ordinary Least square method – Computes error as the sum of squares of distance from the actual value to the predicted line. It fits the model by minimizing the squared error. Assumes strong linear relationship between the inputs and the dependent variables.**
  b) **Gradient descent – Approach is to minimize the amount of error at each step of the model training process.**

Decision Forest Regression

- Ensemble learning method using multiple decision trees
- Each tree outputs a distribution as a prediction
- Aggregate to find a distribution closest to the combined distribution

- Five key parameters used to configure the algorithm are-
  - **f) Resampling method** – This controls the method used to create the individual trees.
  - **g) Number of decision trees** – Maximum number of decision trees that can be, created in the ensemble.
  - **h) Maximum depth of the decision trees** – Number to limit the depth of any decision tree.
  - **i) Number of random splits per node** – Number of splits to use while building each node of the tree.
  - **j) Minimum number of samples per leaf node** – Minimum number of cases that are required to create any terminal node in a tree.

Neural Network Regression

- Supervised learning method
- Label column must be a numerical data type
- Input layer + One hidden layer + Output layer

- Three key parameters used to configure the algorithm are:
  - **d) Number of Hidden Nodes** – Helps in customizing the number of hidden nodes and the neural network
  - **e) Learning Rate** – This controls the size of the step taken at each iteration before correction.
  - **f) Number of Learning Iterations** – The maximum number of times the algorithm should process the training cases.

Common machine learning algorithms for regression problems include:

- Linear Regression
  - Fast training, linear model
- Decision Forest Regression
  - Accurate, fast training times
- Neural Net Regression
  - Accurate, long training times

QUESTION 1 OF 2

Can you match each of these types of regression with the appropriate characteristics?

Submit to check your answer choices!

CHARACTERISTICS	REGRESSION TYPE
Accurate, fast training times	Decision Forest Regression
Accurate, long training times	Neural Net Regression
Fast training, linear model	Linear Regression

QUESTION 2 OF 2

Match the following hyperparameters with their respective descriptions.

Submit to check your answer choices!

DESCRIPTION	HYPERPARAMETER
A value that defines the step taken at each iteration, before correction.	Learning rate
Penalize models to prevent overfitting.	L2 regularization weight
The maximum number of times the algorithm processes the training cases.	Number of learning iterations
A method that minimizes the amount of error at each step of the model training process.	Gradient descent

Chapter 15: Lab – Regressors Performance

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.

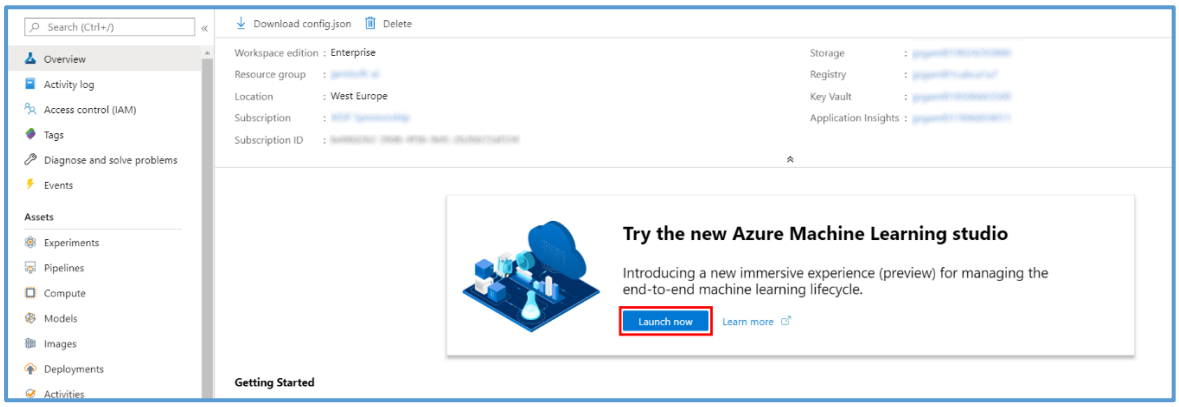
In this lab, we will be compare the performance of two regression algorithms: Boosted Decision Tree Regression and Neural Net Regression for predicting automobile prices. We will do all of this from the Azure Machine Learning designer without writing a single line of code.

Exercise 1: Create Training Pipeline

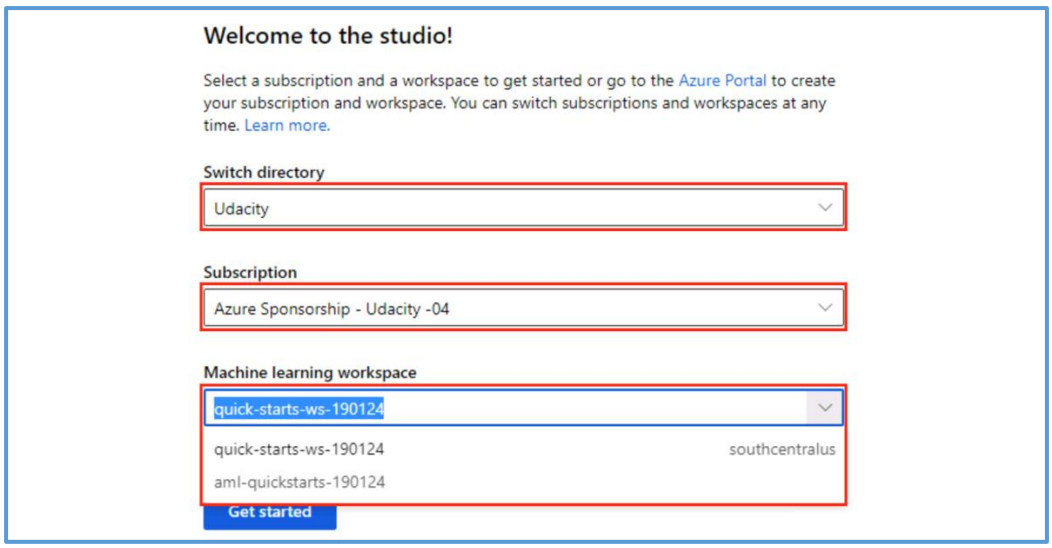
Task 1: Open Sample 2: Regression - Automobile Price Prediction (Compare algorithms)

1. In Azure portal, open the available machine learning workspace.

2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

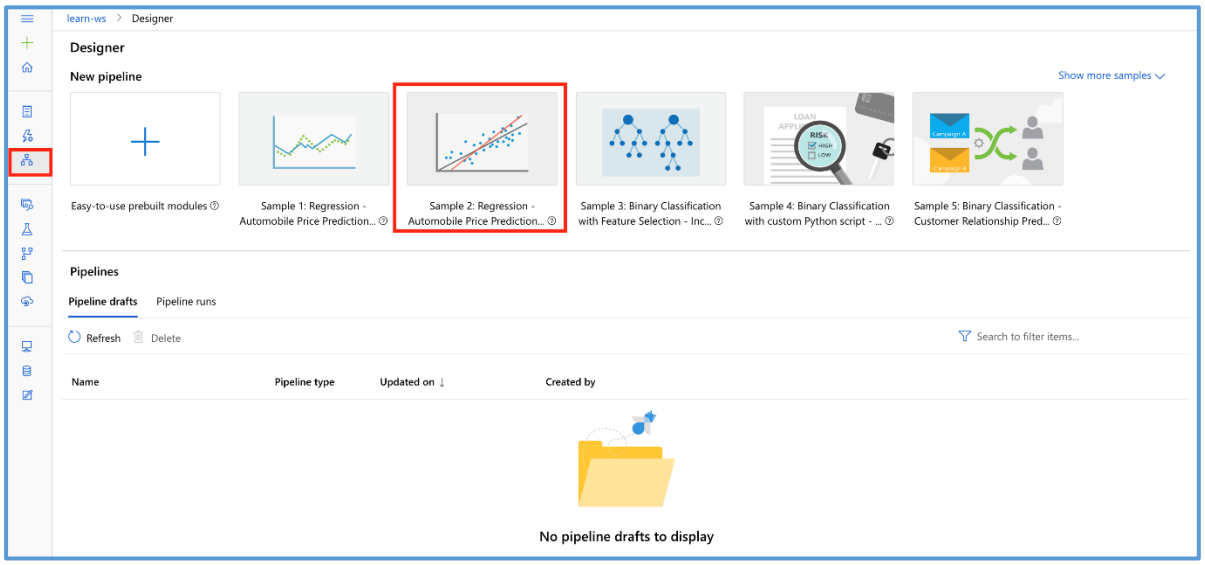


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



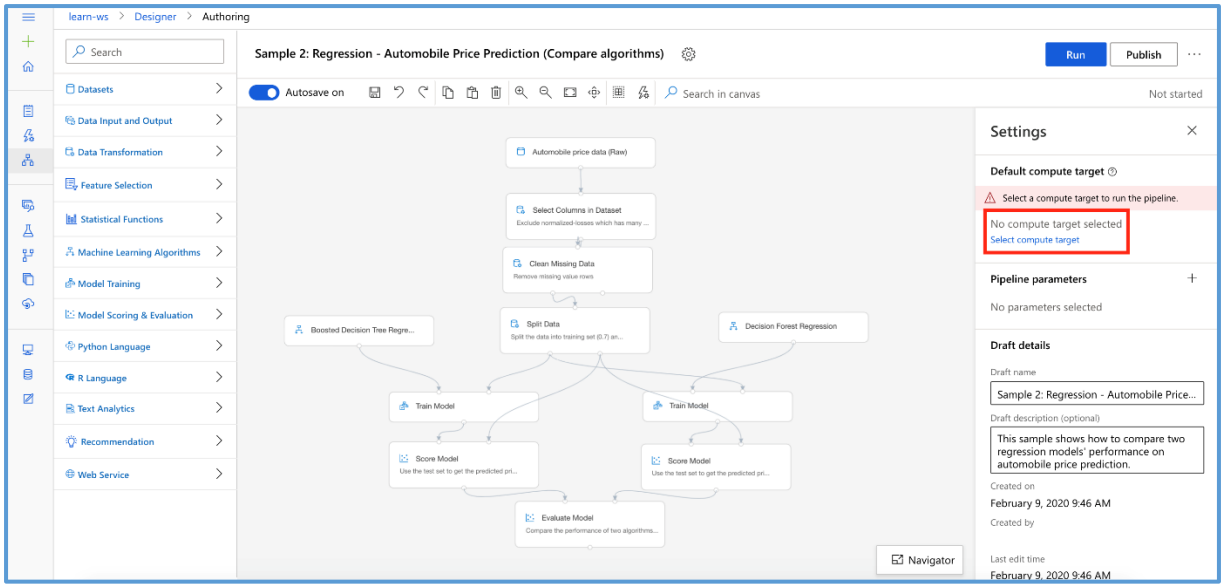
For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it does not matter which) and then click **Get started**.

4. From the studio, select **Designer, Sample 2: Regression - Automobile Price Prediction (Compare algorithms)**.



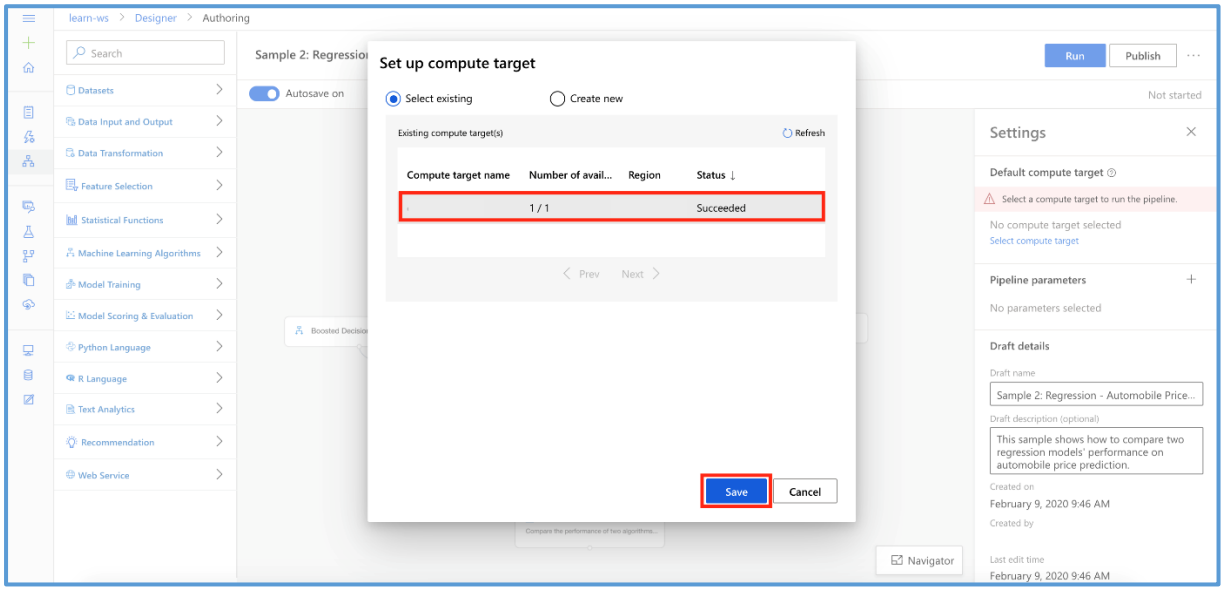
Task 2: Setup Compute Target

1. In the settings panel on the right, select **Select compute target**.



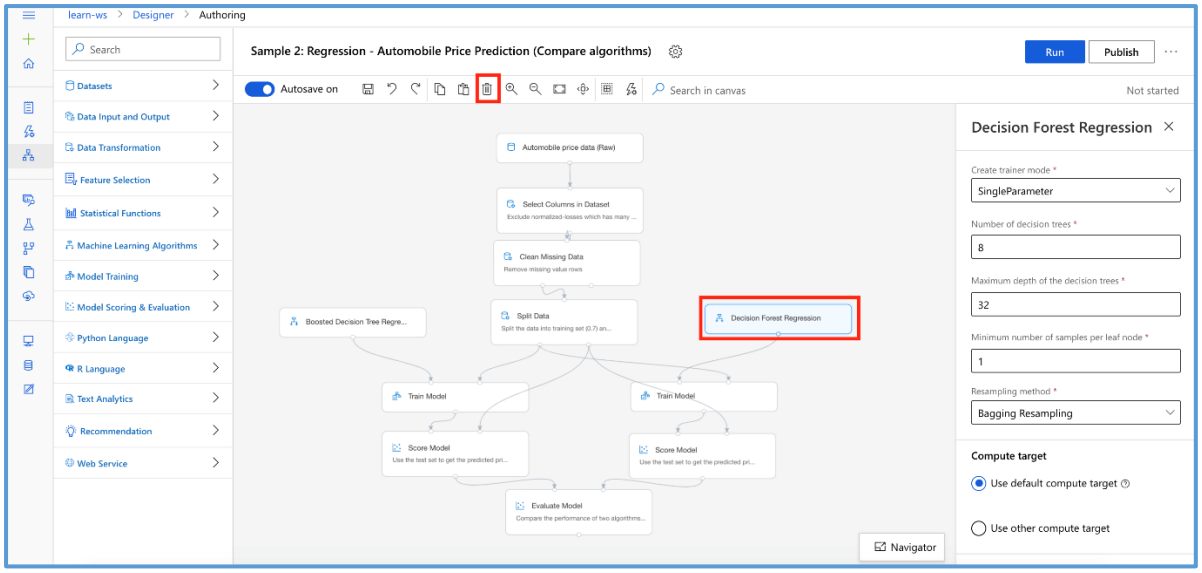
2. In the Set up compute target editor, select the available compute, and then select **Save**.

Note: If you are facing difficulties in accessing pop-up windows or buttons in the user interface, please refer to the Help section in the lab environment.



Task 3: Delete Pipeline Modules

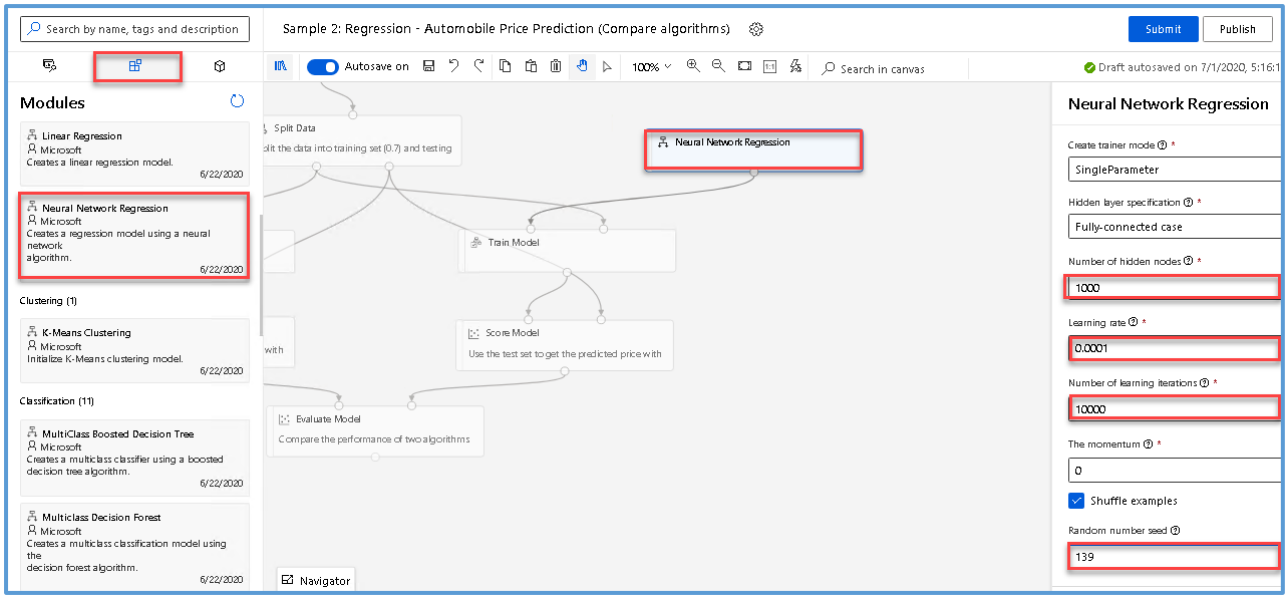
1. From the right-hand-side of the pipeline, select the **Decision Forest Regression module** and then select the **Delete Icon**.



Task 4: Setup the Neural Net Regression Module

1. Select **Machine Learning Algorithms** section in the left navigation. Follow the steps outlined below:

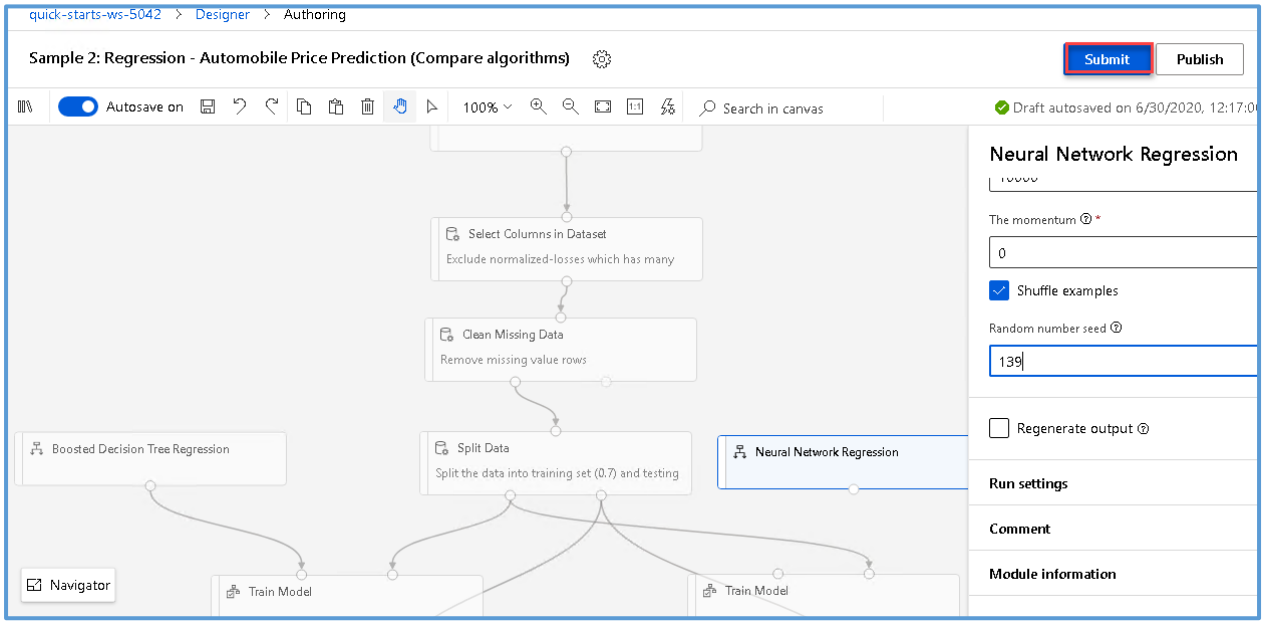
    1. Select the **Neural Net Regression** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Set `Number of hidden nodes` to **1000**
    4. Set `Learning rate` to **0.0001**
    5. Set `Number of learning iterations` to **10000**
    6. Set `Random number seed` to **139**
    7. Connect the `Neural Net Regression` module to the first input of the `Train Model` module



Exercise 2: Submit Training Pipeline

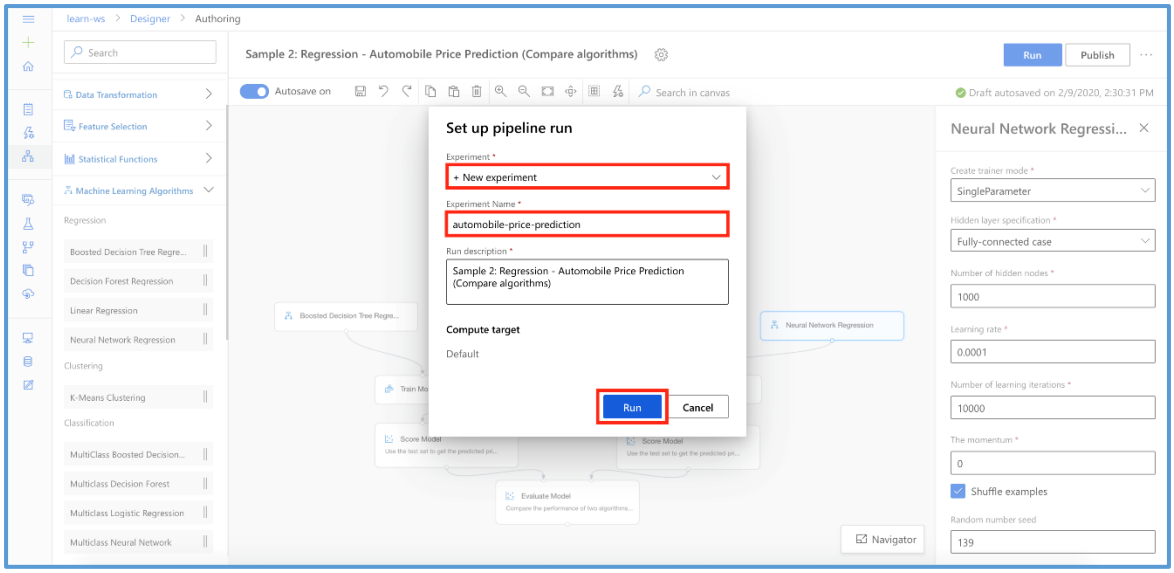
Task 1: Create Experiment and Submit Pipeline

1. Select **Submit** to open the Setup pipeline run editor.



Please note that the button name in the UI is, changed from **Run** to **Submit**.

2. In the Setup pipeline run editor, select **Experiment, Create new** and provide New experiment name: **automobile-price-prediction**, and then select **Submit**.



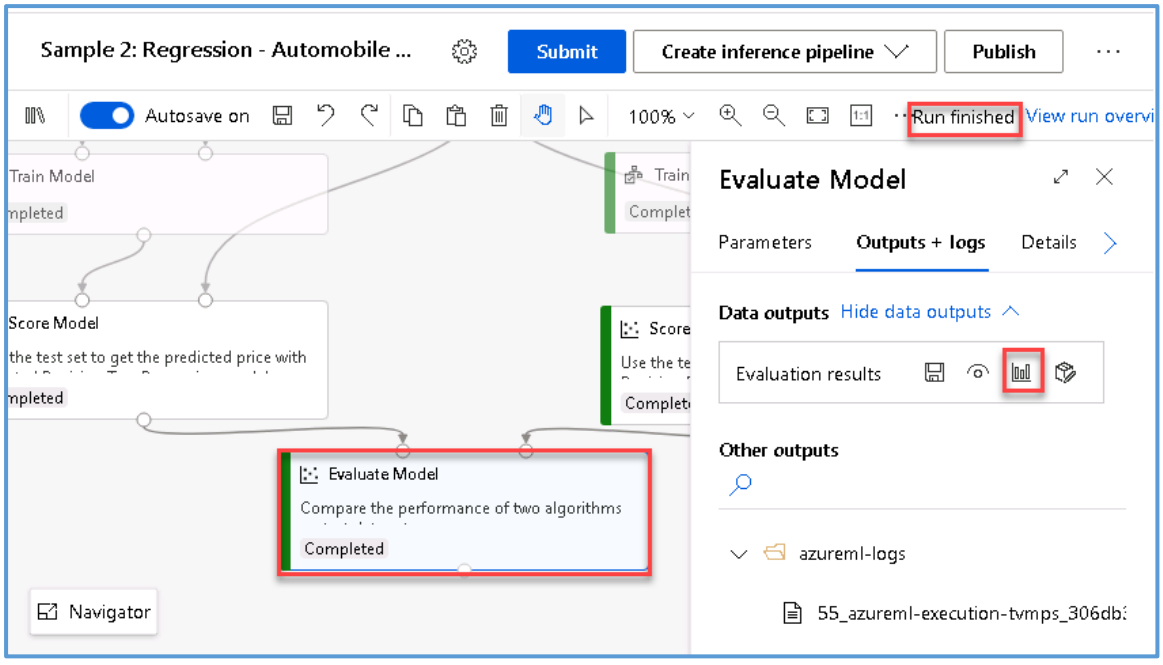
3. Wait for pipeline run to complete. It will take around **10 minutes** to complete the run.

4. While you wait for the model training to complete, you can learn more about the evaluation metrics for the regression algorithm used in this lab by selecting Metrics for regression models.

Exercise 3: Compare Model Performance

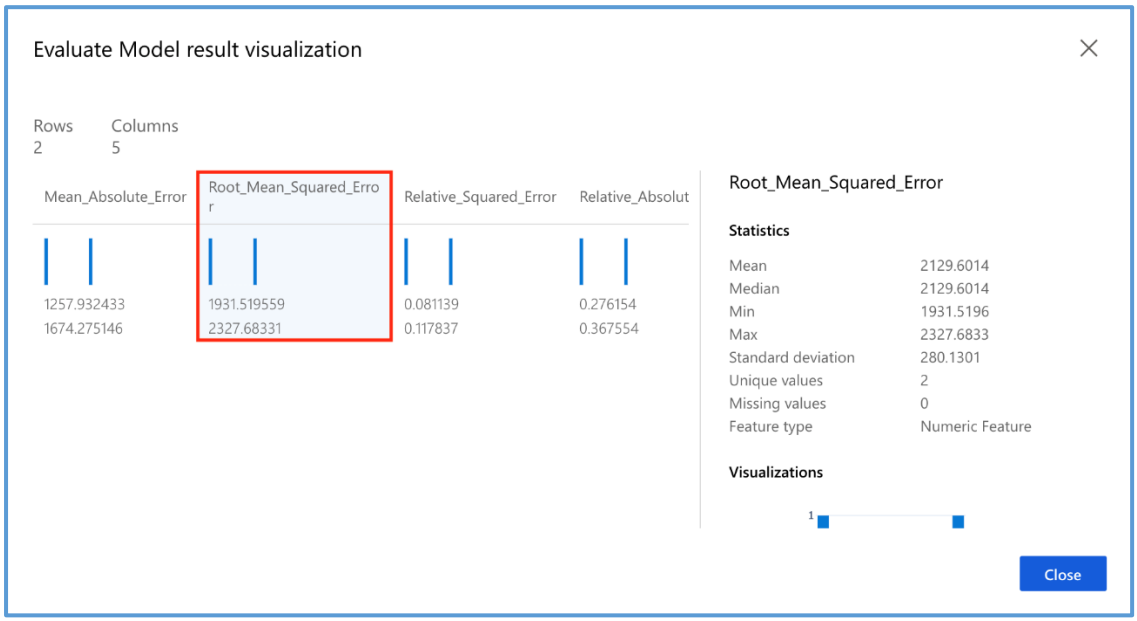
Task 1: Open Evaluation Results

1. Select **Evaluate Model, Outputs, Visualize** to open the Evaluate Model result visualization dialog.



Task 2: Compare Performance Metrics

1. Select the regression performance metric **Root_Mean_Squared_Error** and compare performance of the two algorithms: `Boosted Decision Tree Regression` and `Neural Net Regression`. Note that smaller value for `Root_Mean_Squared_Error` implies better performance.



Task 3: Conclusion

1. Based on the performance metric, `Root_Mean_Squared_Error`, it shows that the `Boosted Decision Tree Regression` algorithm outperforms the `Neural Net Regression` algorithm. One recommendation for next steps is to tune the hyperparameters for the `Neural Net Regression` module to see if we can improve its performance.

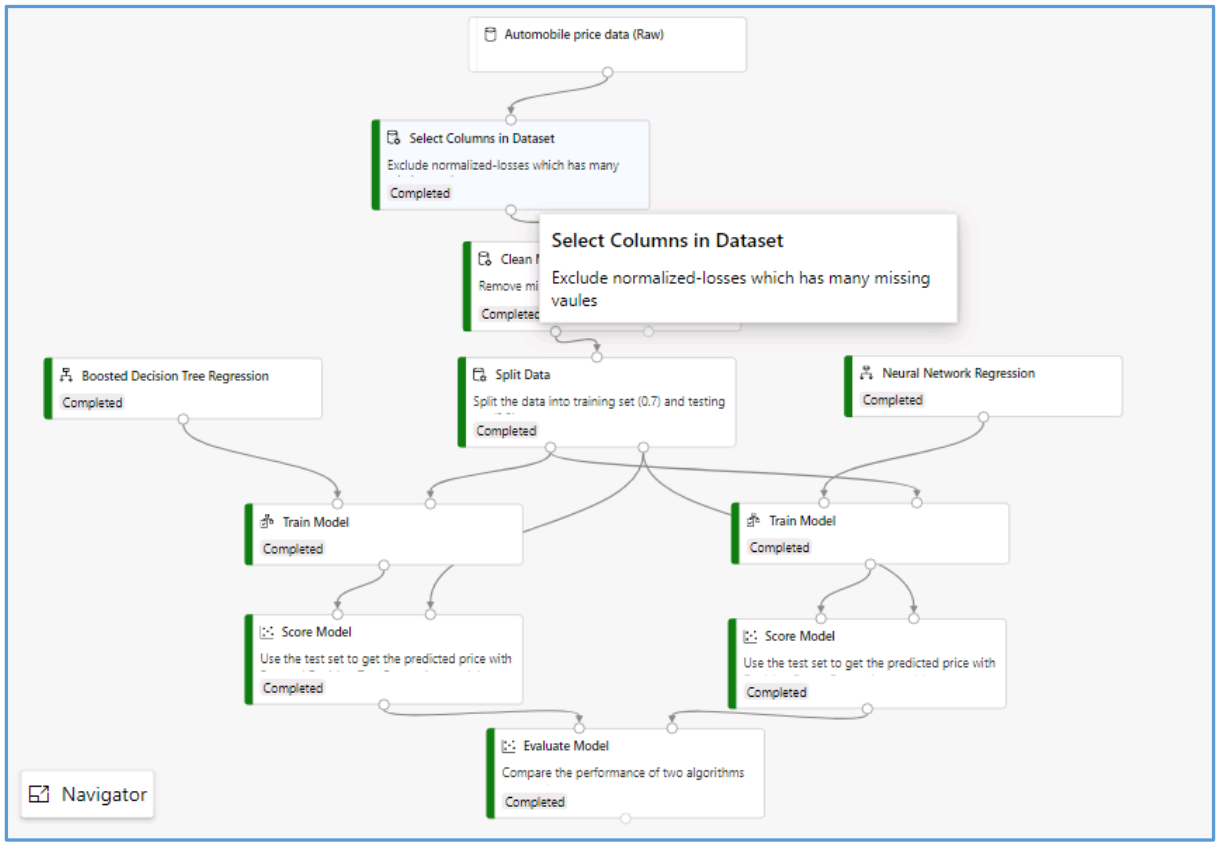
Next Steps

Congratulations! You have trained and compared performance of two different regression machine-learning models. You can continue to experiment in the environment but are free to close the lab environment tab and return to the Udacity portal to continue with the lesson.

Chapter 16: Walkthrough – Regressors Performance

- We will compare the performance of two regression algorithms.
- Boosted Decision Tree and Neural Net Regression for predicting Automobile price.
- Use Sample 2 regression automobile price pipeline from the designer
- Setup the compute resource which would scale automatically as per the requirement
- Hyperparameters related to neural net needs to be, specified.

- Both models are, compared using the RMSE value. Value shows that Boosted Decision Tree regression outperforms Neural Net Regression.
- We can tune the hyperparameters for Neural Net Regression to improve model's performance



Chapter 17: Prelaunch Lab

Prelaunch your lab environment.

NOTE: When you click "Prelaunch Lab", we will begin preparing a lab environment for you. When it is time to access the lab environment, this ensures you will be able to start working. Once your lab is reserved, you can continue on to the next concepts.

Your lab has been, reserved! Please continue to the next concept.

Chapter 18: Automate the Training of Regressors

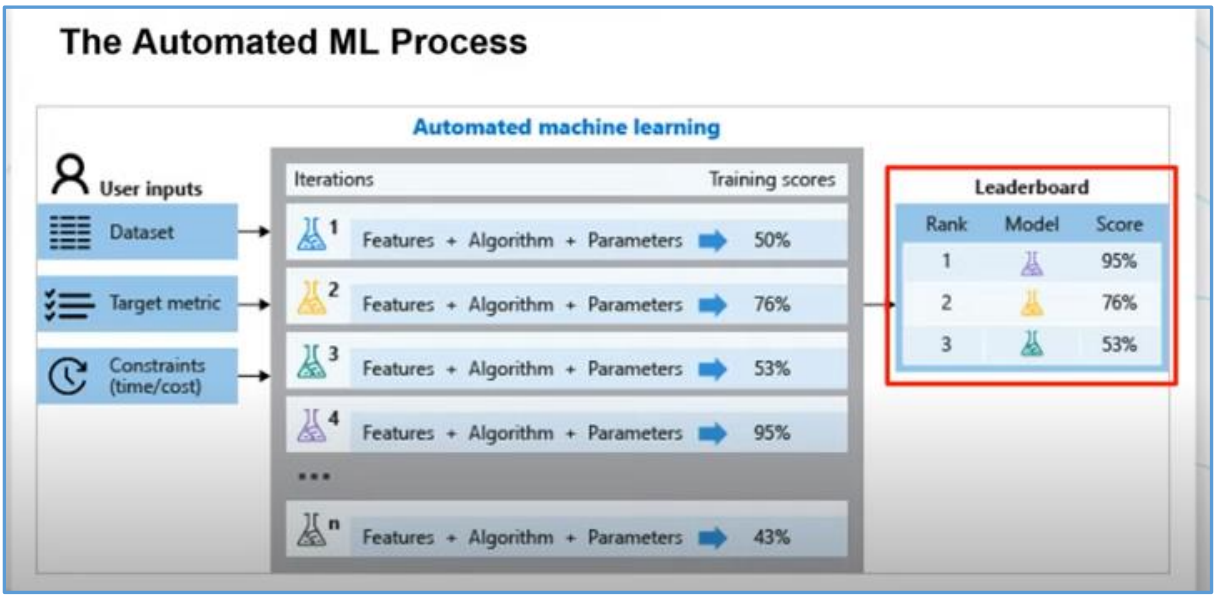
Key challenges in successfully training a machine-learning model: [Entire process is pretty, iterative]

- Features available in the data sets
- Algorithms that are suitable for the task
- Hyperparameters tuning
- Evaluation metrics

Auto ML is to enable the automated exploration of the combinations needed to successfully, produce a training model. It allows to, build ML models with high scale efficiency and productivity all while sustaining model quality.

Intelligently test **multiple** algorithms and hyper-parameters in parallel and return the best one

- Deploy into production
- Further customize or refine



QUIZ QUESTION

Automated Machine Learning gives users the option to automatically scale and normalize input features. It also gives users the ability to enable additional featurization, such as missing values imputation, encoding, and transforms.

What is the default behavior in Automated Machine Learning to deal with missing values for categorical features?

- ○ Impute with average value
- ✓ Impute with most frequent value
- ○ Custom substitution value
- ○ Remove entire row

Chapter 19: Lab - Train a Regressor using Automated Machine Learning

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.

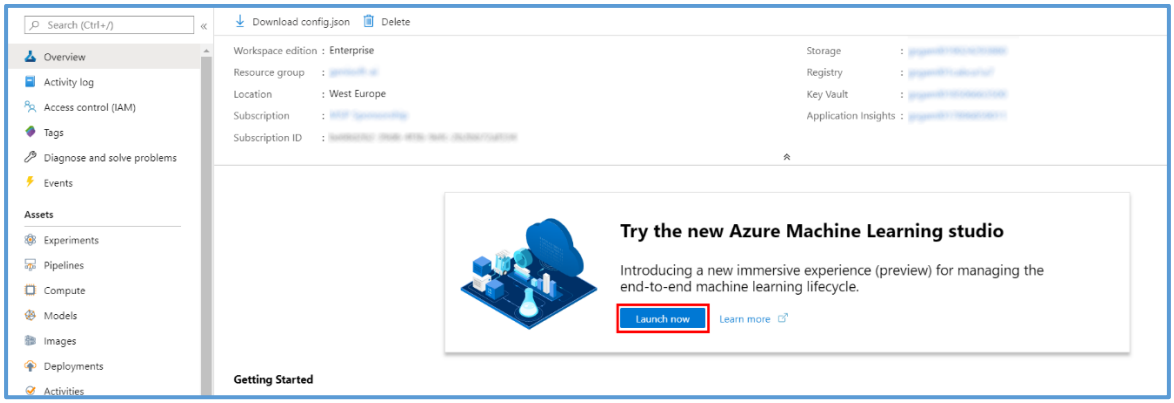
In this lab, we will use Automated Machine Learning to find the best performing regression model for predicting automobile prices. We will do all of this from the Azure Machine Learning Studio without writing a single line of code.

Exercise 1: Register Dataset with Azure Machine Learning studio

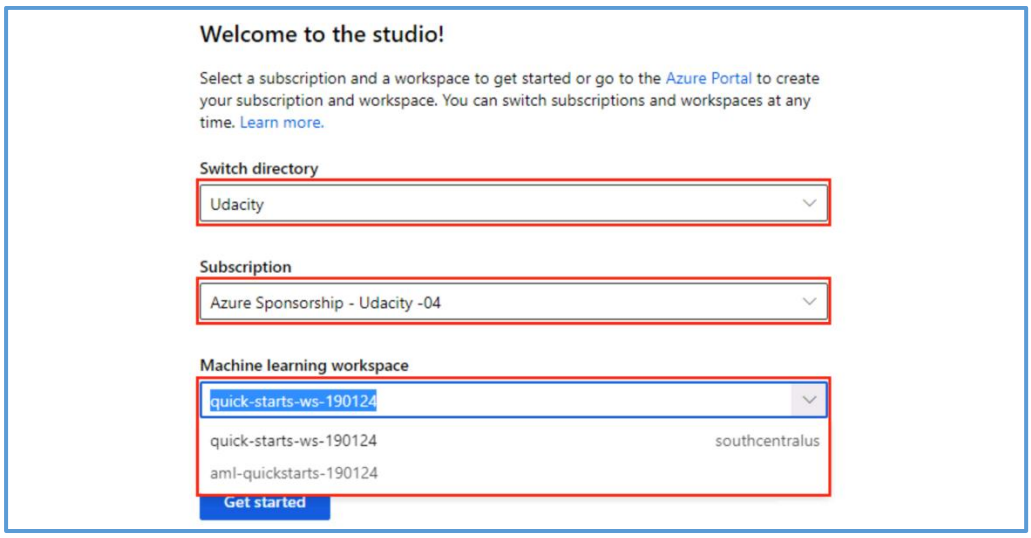
Task 1: Upload Dataset

1. In Azure portal, open the available machine learning workspace.

2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

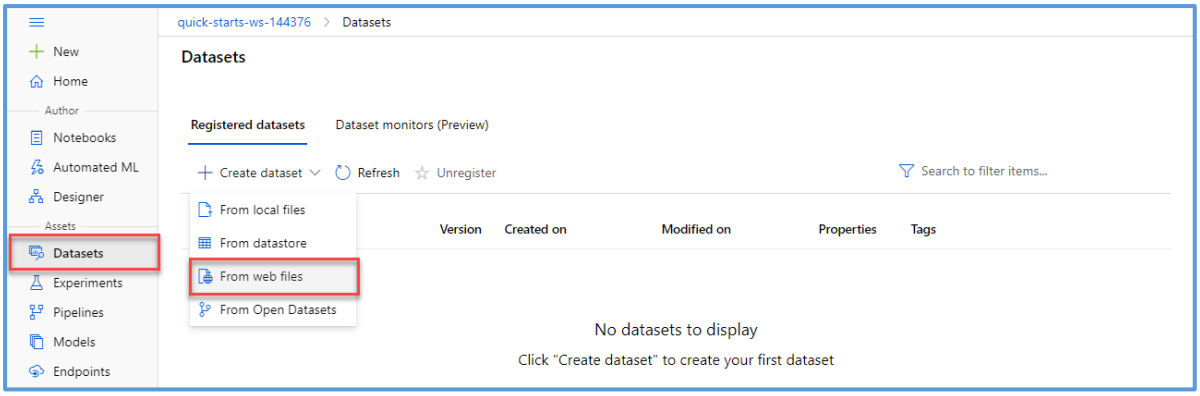


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

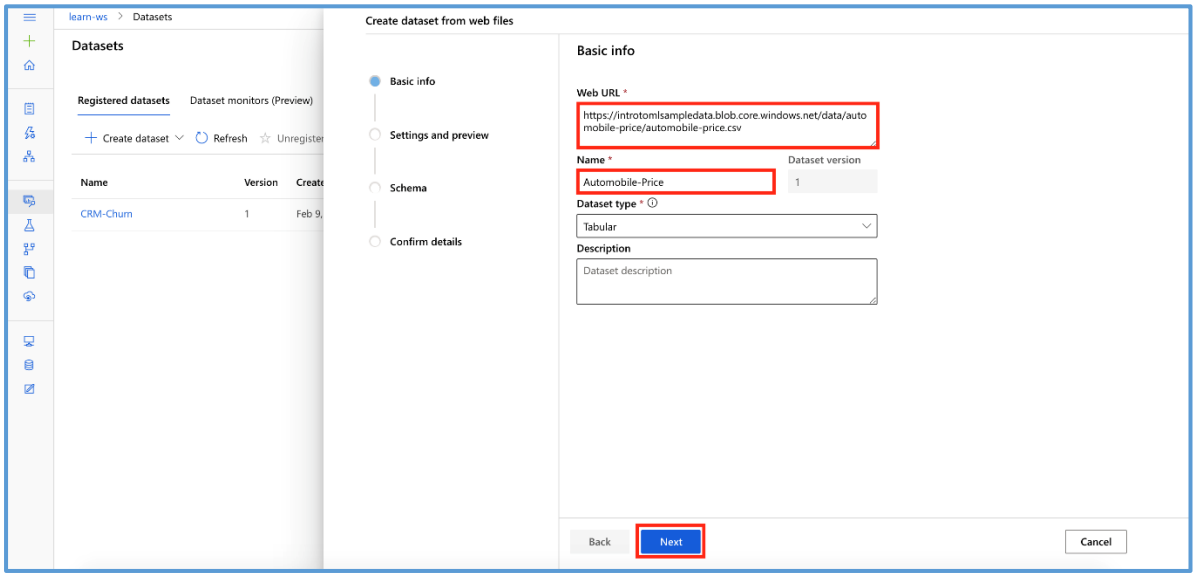
4. From the studio, select **Datasets, + Create dataset, From web files**. This will open the Create dataset from web files dialog on the right.



5. In the Web URL field provide the following URL for the training data file:

```
https://introtomlsampledata.blob.core.windows.net/data/automobile-price/automobile-price.csv
```

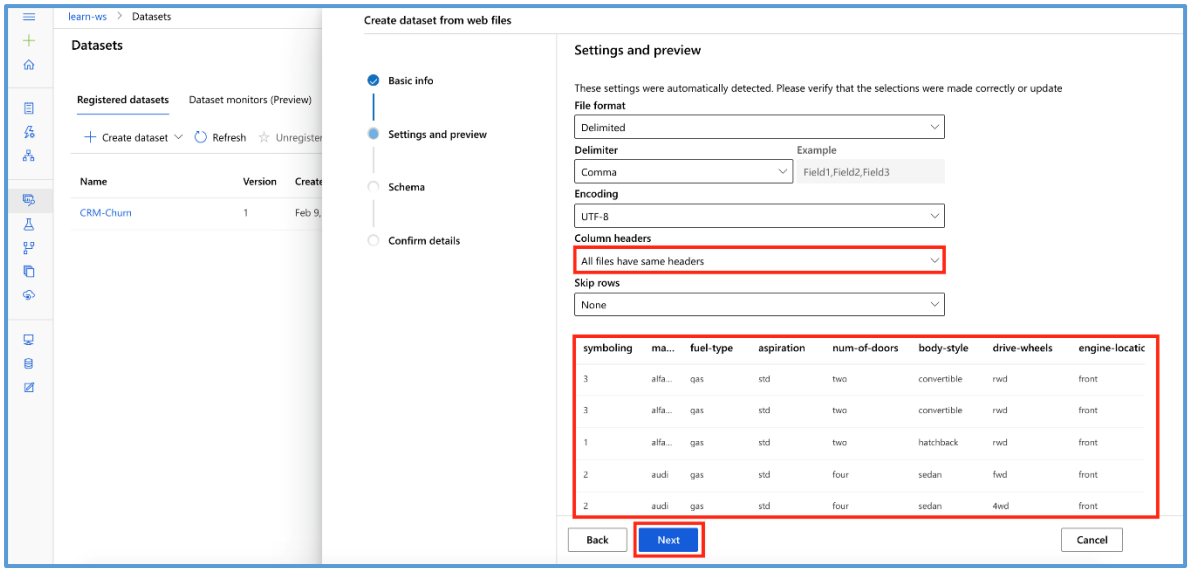
6. Provide `Automobile-Price` as the Name, leave the remaining values at their defaults and select **Next**.



Task 2: Preview Dataset

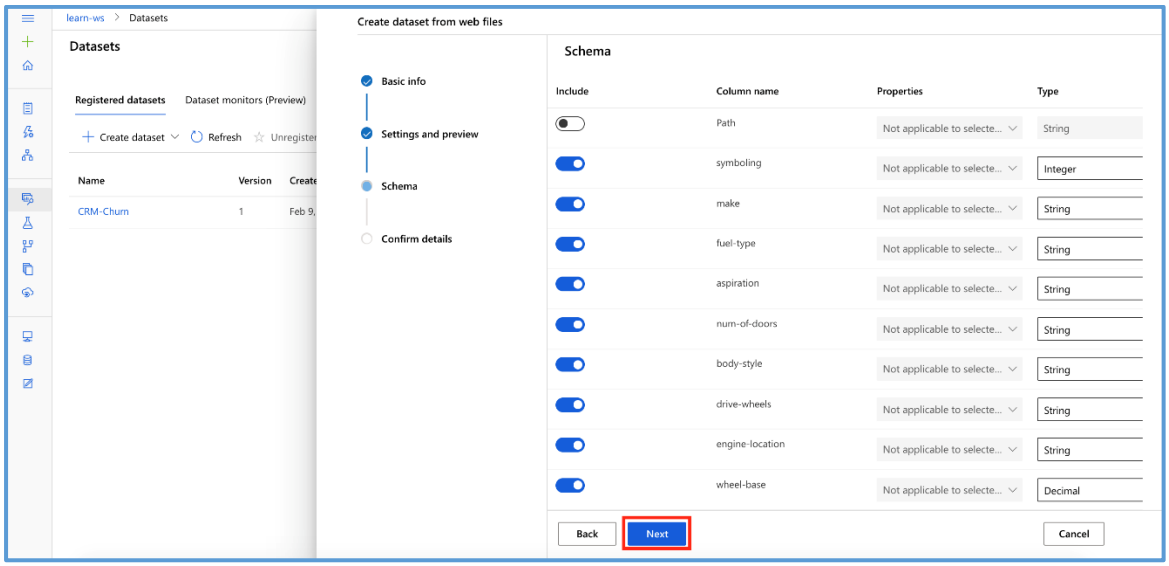
1. On the Settings and preview panel, set the column headers drop down to `All files have same headers`.

2. Review the dataset and then select **Next**



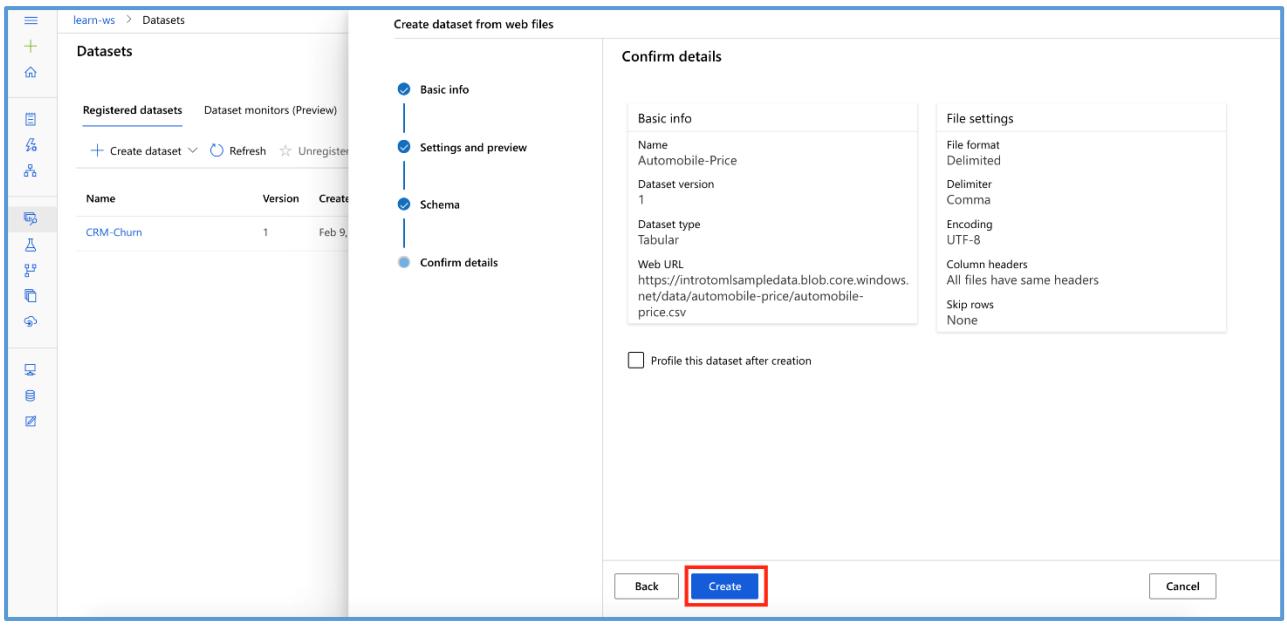
Task 3: Select Columns

1. Keep the default selections, and select **Next**



Task 4: Create Dataset

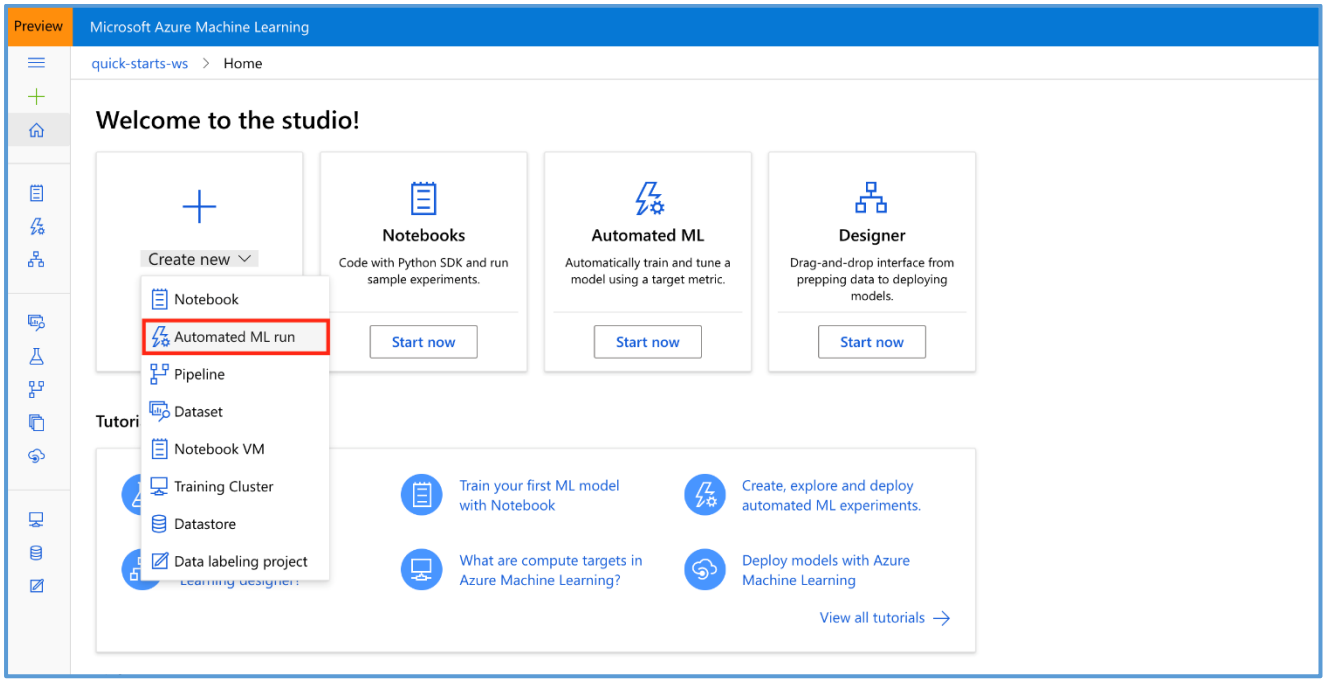
1. Confirm the dataset details and select **Create**



Exercise 2: Setup New Automated Machine Learning Experiment

Task 1: Create New Automated Machine Learning Experiment

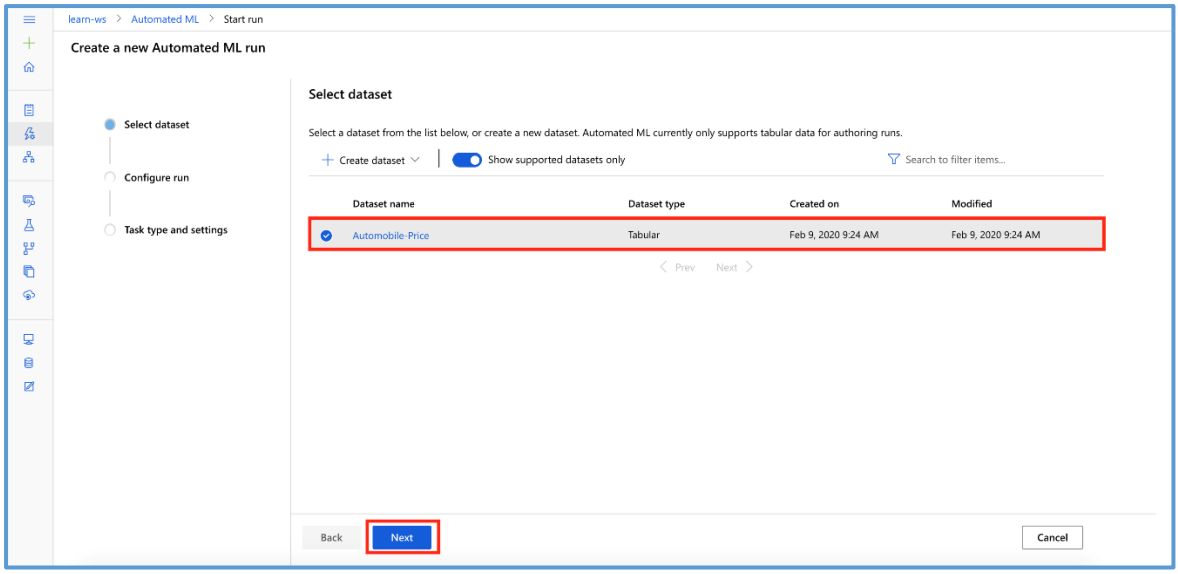
1. From the studio home, select **Create new, Automated ML run**



2. This will open a Create a new automated machine learning experiment page

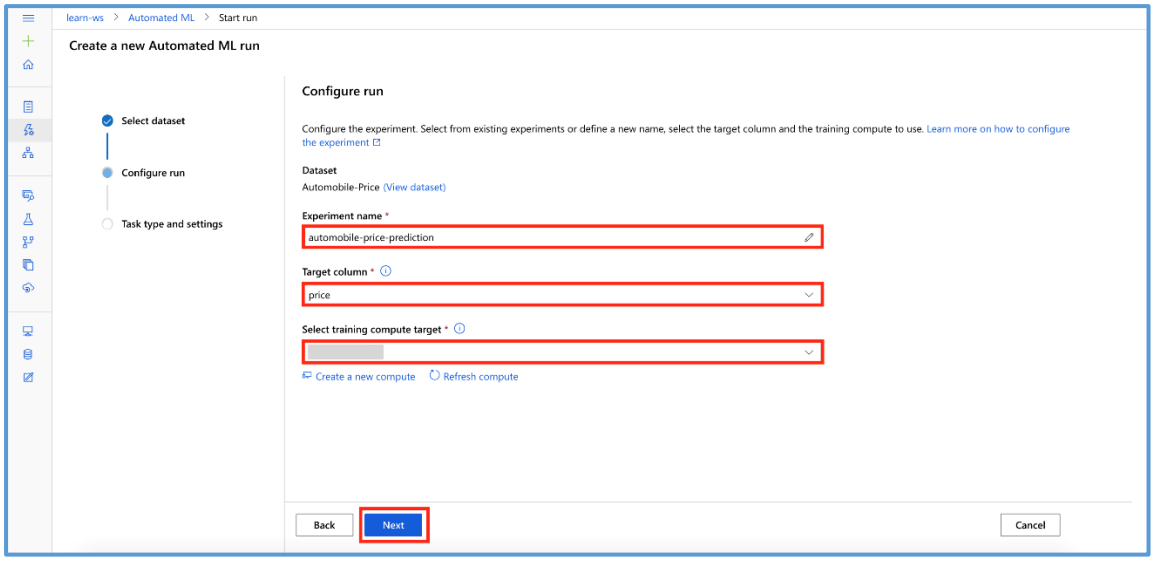
Task 2: Select Training Data

1. Select the dataset Automobile-Price and then select **Next**



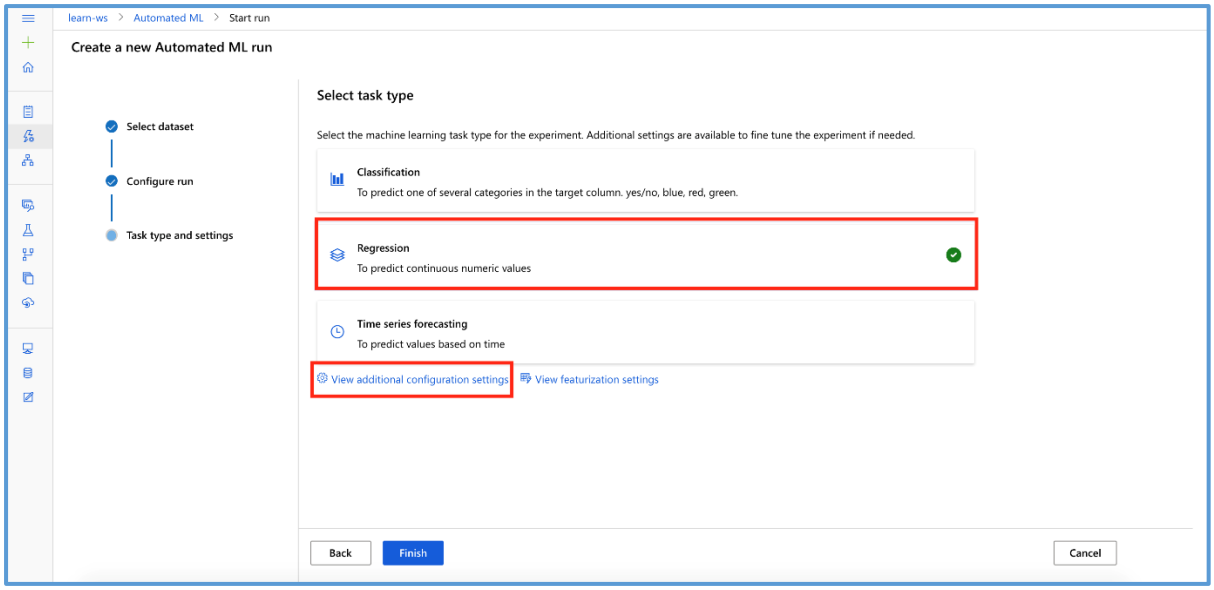
Task 3: Create a new Automated ML run

1. Provide an experiment name: **automobile-price-prediction**
2. Select target column: **price**
3. Select compute target: **select the available compute**
4. Select **Next**



Task 4: Setup Task type and Settings

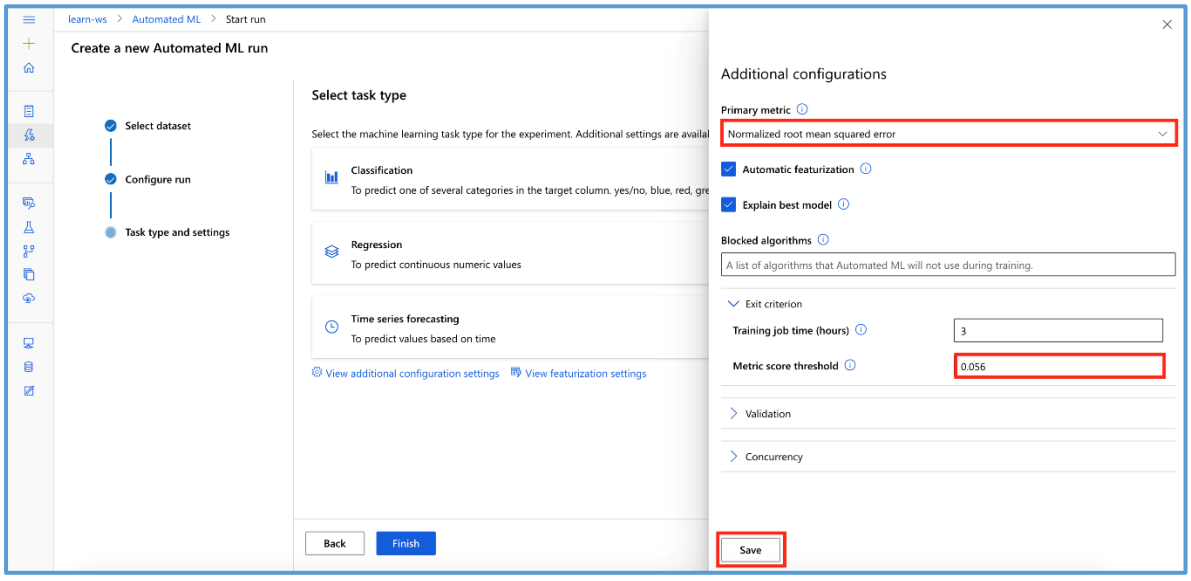
1. Select task type: **Regression**, and then select **View additional configuration settings**



2. This will open the Additional configurations dialog.

3. Provide the following information and then select **Save**

    1. Primary metric: **Normalized root mean squared error**
    2. Exit criteria, Metric score threshold: 0.056

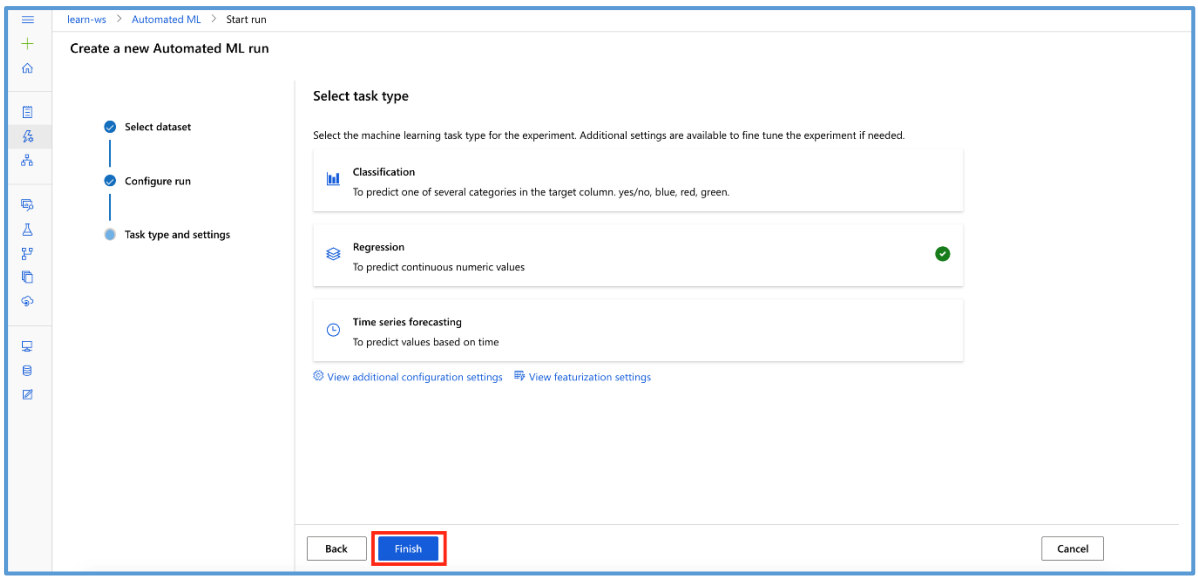


Note that we are setting a metric score threshold to limit the training time. In practice, for initial experiments, you will typically only set the training job time to allow AutoML to discover the best algorithm to use for your specific data.

Exercise 3: Start and Monitor Experiment

Task 1: Start Experiment

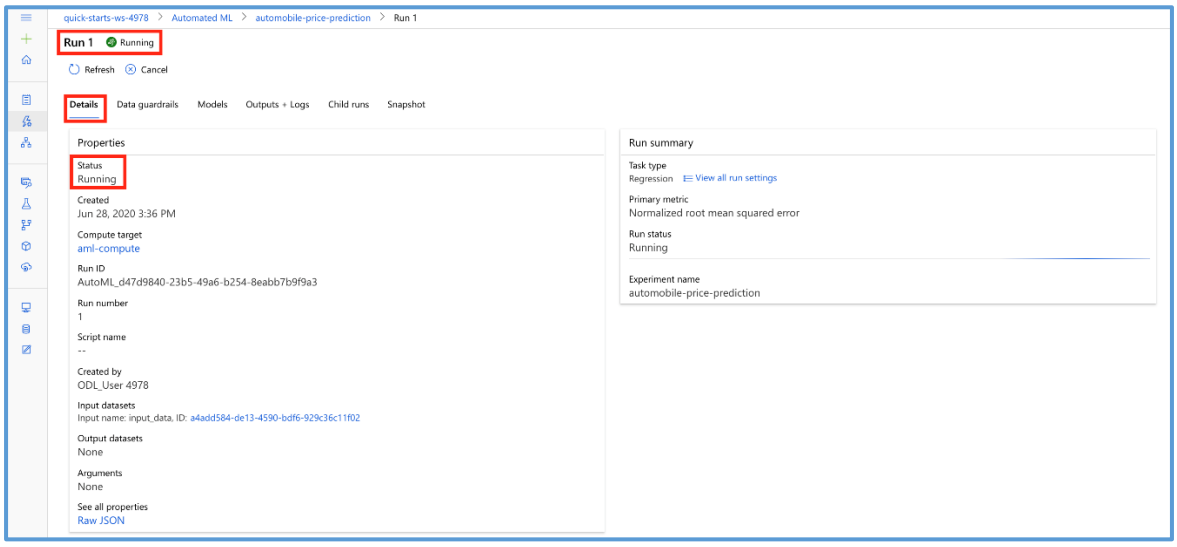
1. Select **Finish** to start running the experiment



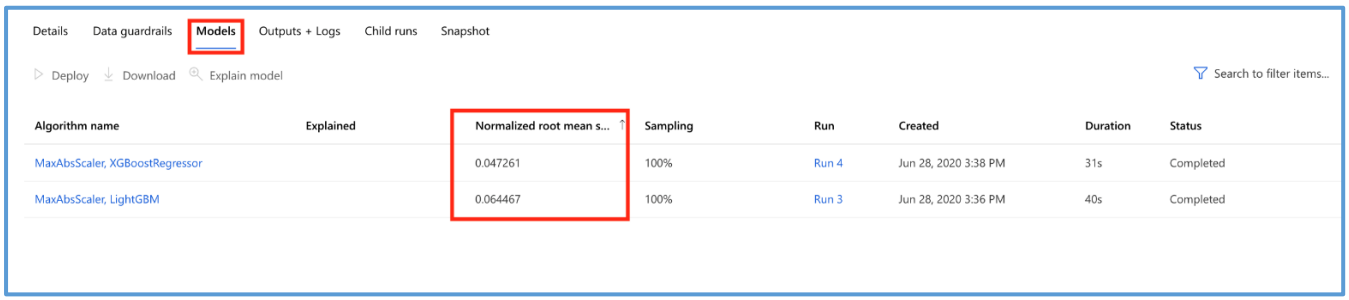
Task 2: Monitor Experiment

1. The experiment will run for about *10 min*.

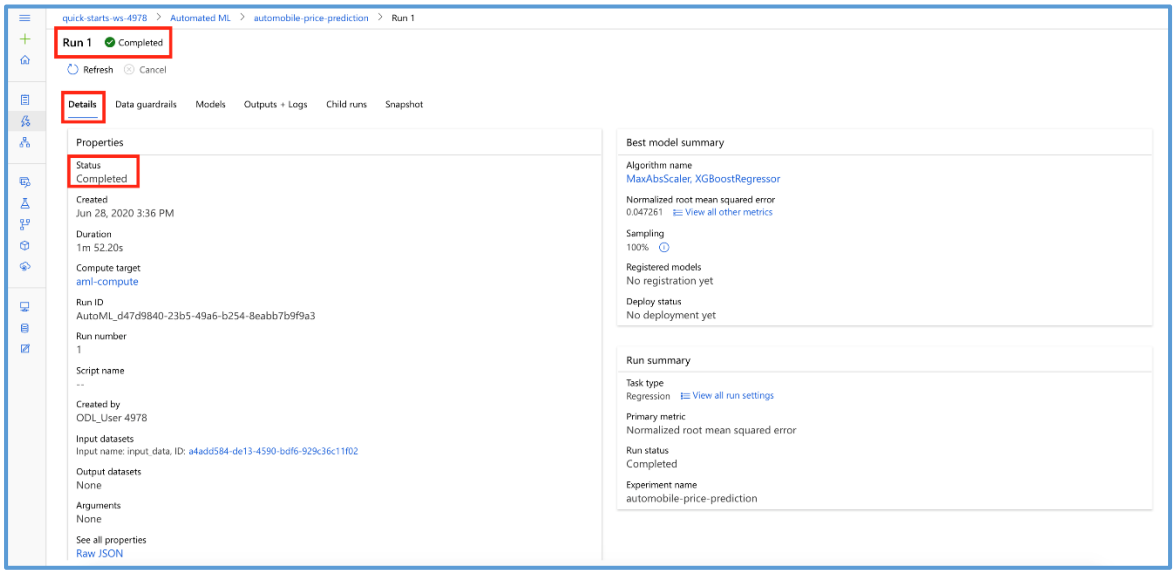
2. In the **Details** tab, observe the **run status** of the job.



3. Select the **Models** tab, and observe the various algorithms the AutoML is evaluating. You can also observe the corresponding **Normalized root mean squared error** scores for each algorithm.



4. Select **Details** and wait till the run status becomes **Completed**.

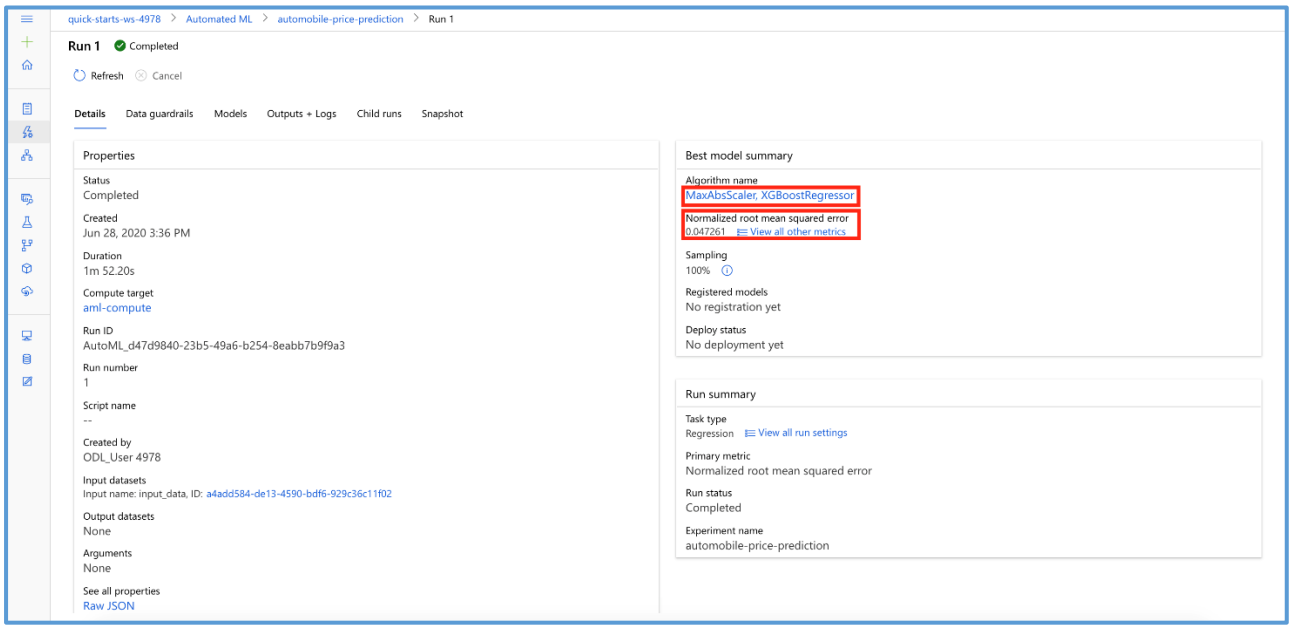


5. While you wait for the model training to complete, you can learn more about how Automated Machine Learning offers preprocessing and data guardrails automatically by selecting Automatic featurization.

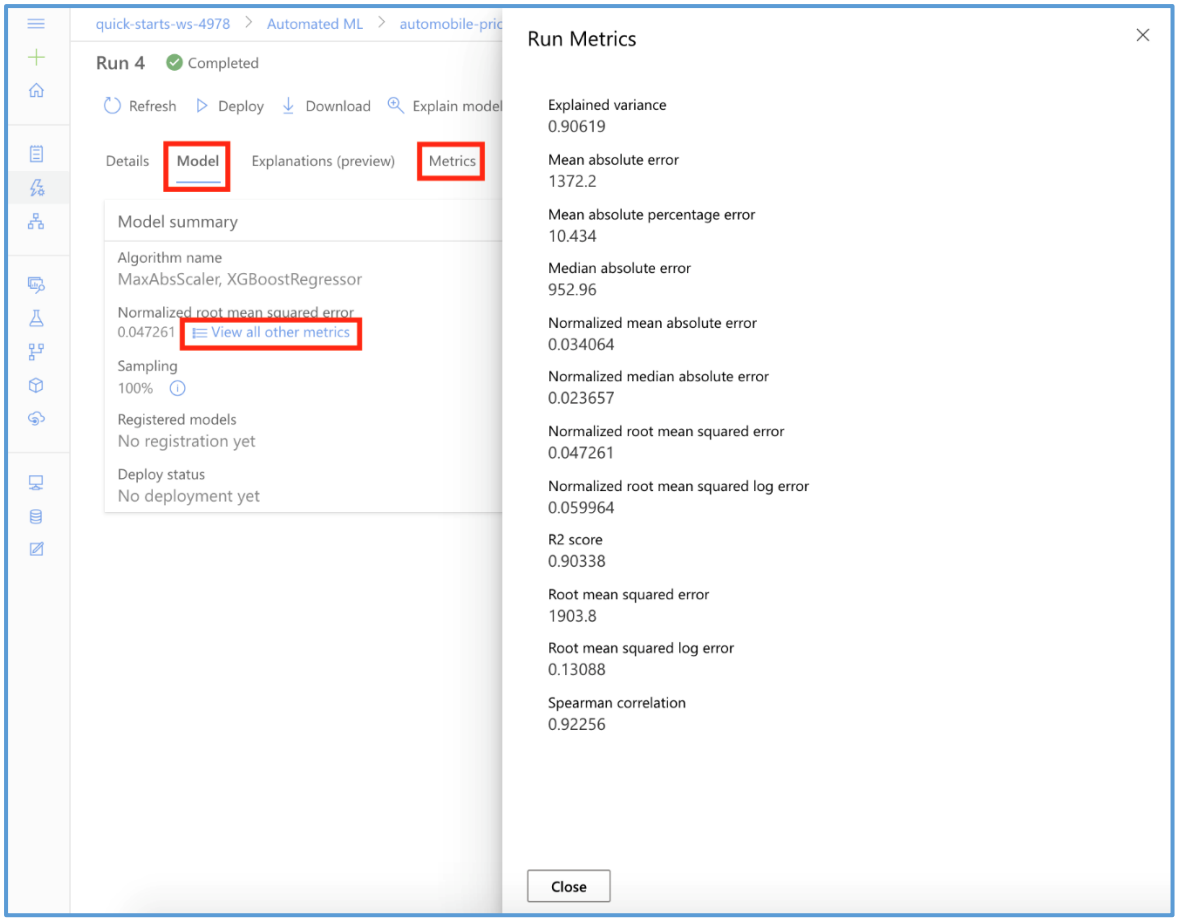
Exercise 4: Review Best Model's Performance

Task 1: Review Best Model Performance

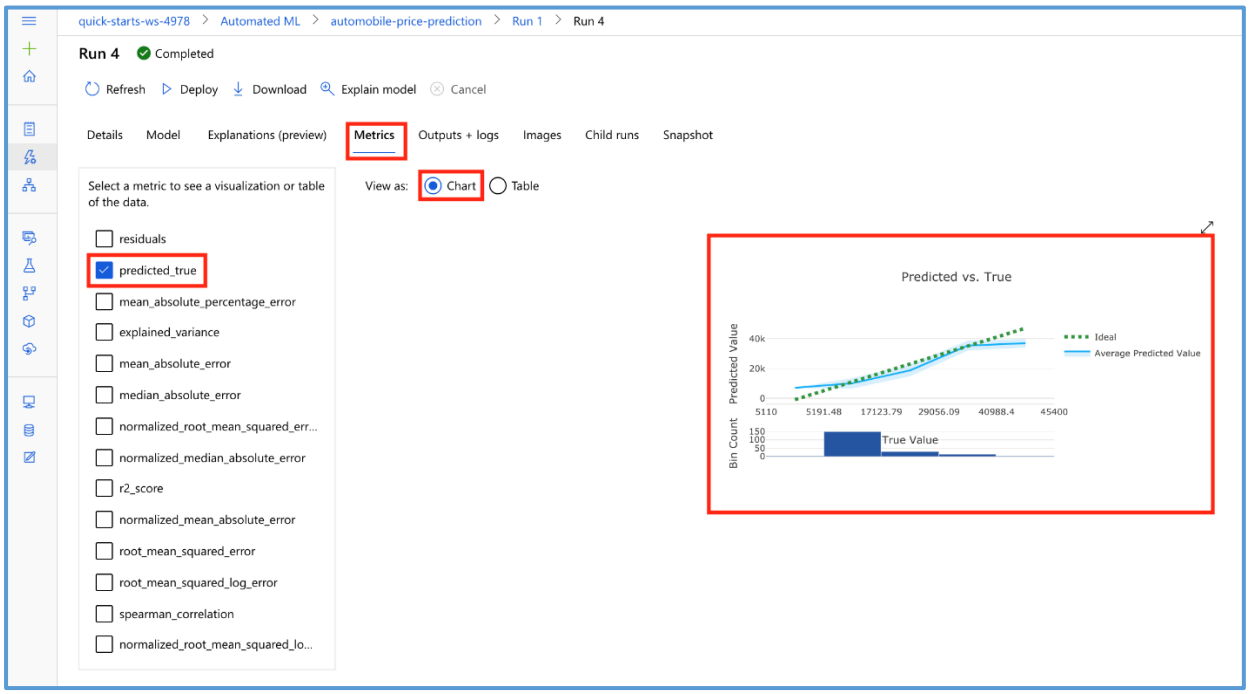
1. From the Details tab review the best model's Algorithm name and its corresponding Normalized root mean squared error score. Next, select the best model's **Algorithm name**



2. Select **View all other metrics** to review the various Run Metrics to evaluate the model performance. Next, select **Metrics**



3. Select **predicted_true, Chart** to review the Predicted vs. True curve.

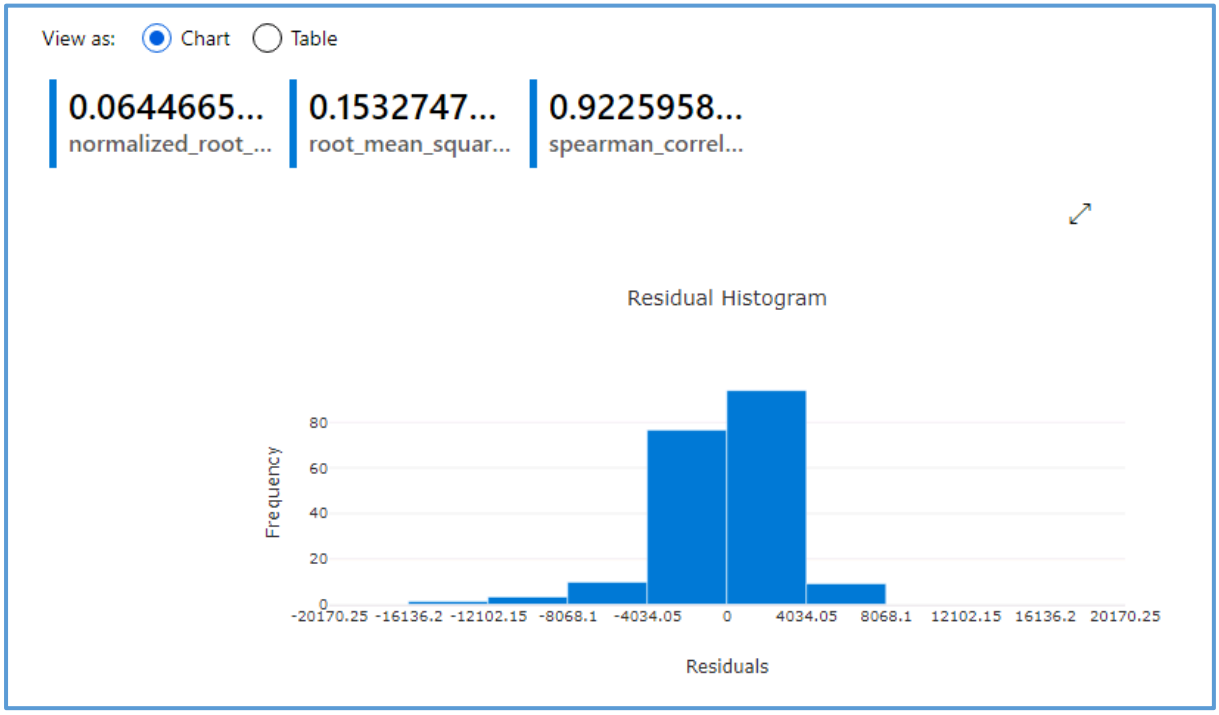


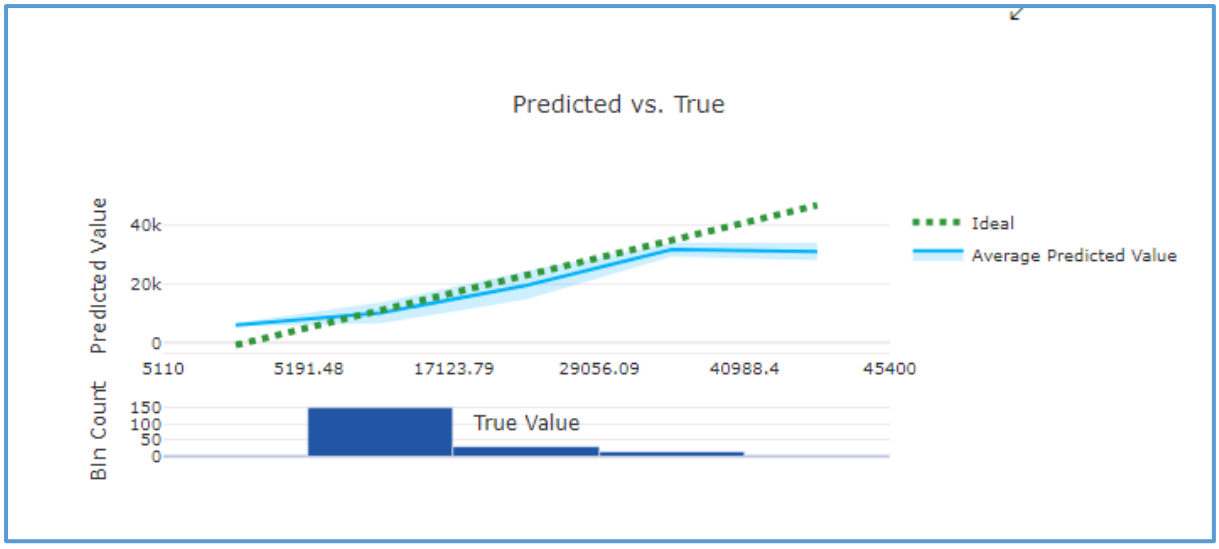
Next Steps

Congratulations! You have trained and evaluated a regression model using automated machine learning. You can continue to experiment in the environment but are free to close the lab environment tab and return to the Udacity portal to continue with the lesson.

Chapter 20: Walkthrough - Train a Regressor using Automated Machine Learning

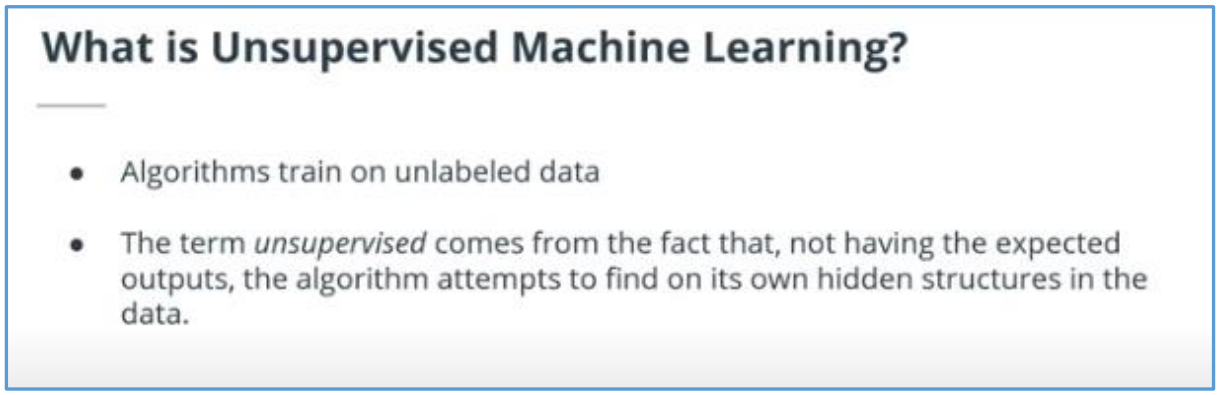
- We will use AutoML learning to find the best performing regression model.
- First setup the dataset
- Initiate a new AutoML run using the created dataset.
- The target column is “price”
- Task type would be Regression.
- Primary metric would be Normalized Root mean squared error.
- Under Exit criteria – Metric Score Threshold would be 0.056. As soon as the model reaches this score, it would complete the run.
- It would test multiple models for us and give back the optimized model.





Chapter 21: Unsupervised Learning

Unsupervised Learning groups those algorithms, which rely only on inputs in the training process.



Training process identifies common aspects between entities & then use a presence or absence of these to produce various results.

All of the algorithms we have looked at so far are examples of *supervised learning*, in which the training data is, *labelled*.
For example, if we are training a classifier to recognize an image of a cat, we might have some images in our training dataset that we already know have cats—and are labelled as such.
However, the cost of obtaining labelled data can be high & sometimes not scalable. So now, let us turn our attention to the second main type of machine learning explored in this lesson: **unsupervised learning**.
*In **unsupervised learning**, algorithms learn from unlabelled data by looking for hidden structures in the data.*

Obtaining unlabelled data is comparatively inexpensive and unsupervised learning can be, used to uncover very useful information in such data.

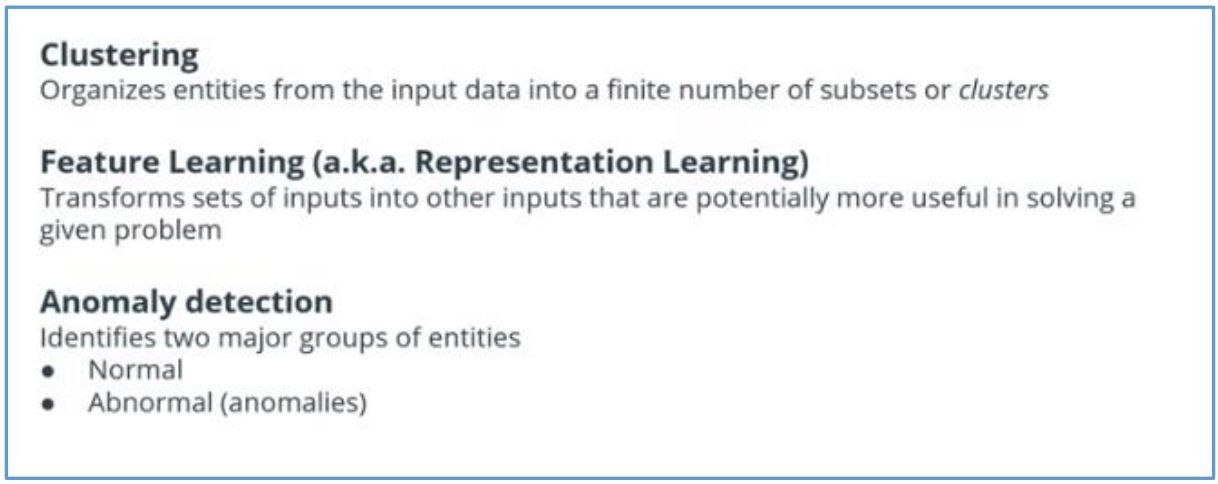
For example, we can use **clustering** algorithms to discover implicit grouping within the data, or use **association** algorithms to discover hidden rules that are governing the data (e.g., people who buy product A also tend to buy product B).

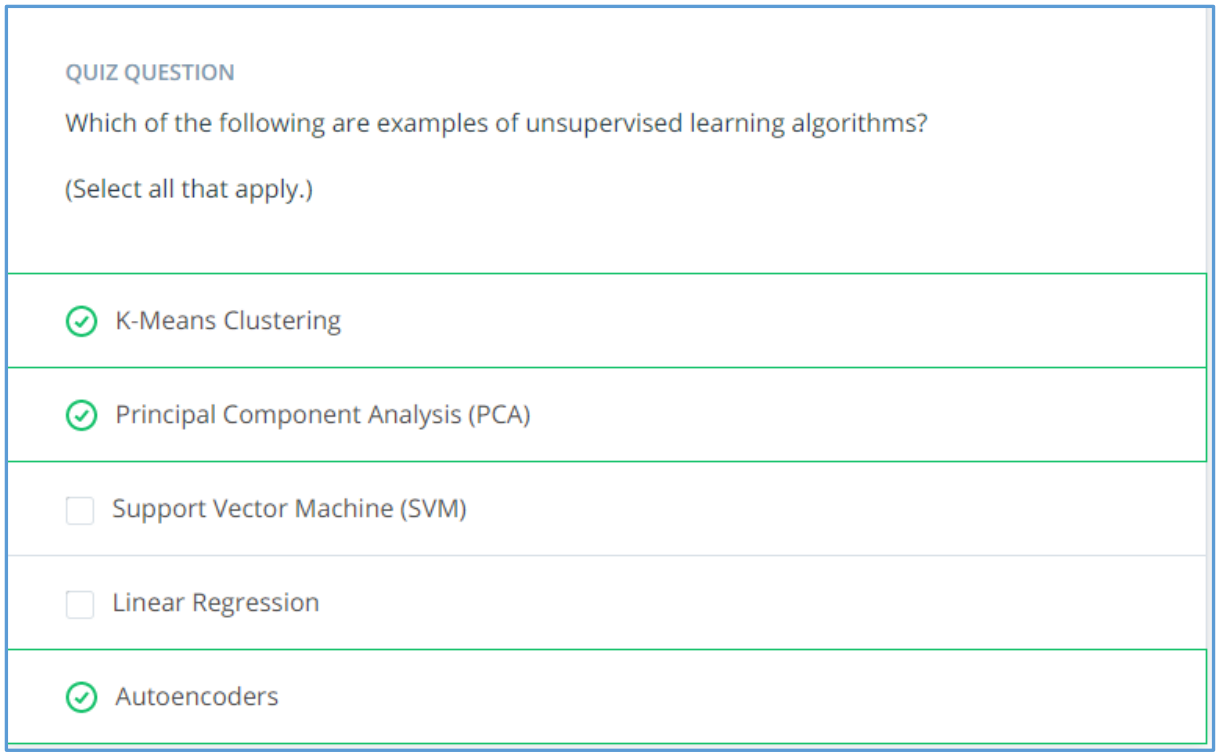
Additional things we can do are Dimensionality reduction, Feature extraction and Anomaly detection.

Types of unsupervised learning algorithms includes Clustering and Feature Learning.

Additionally Neural networks, PCA, matrix factorization and anomaly detection are other unsupervised algorithms.

Types of Unsupervised Machine Learning

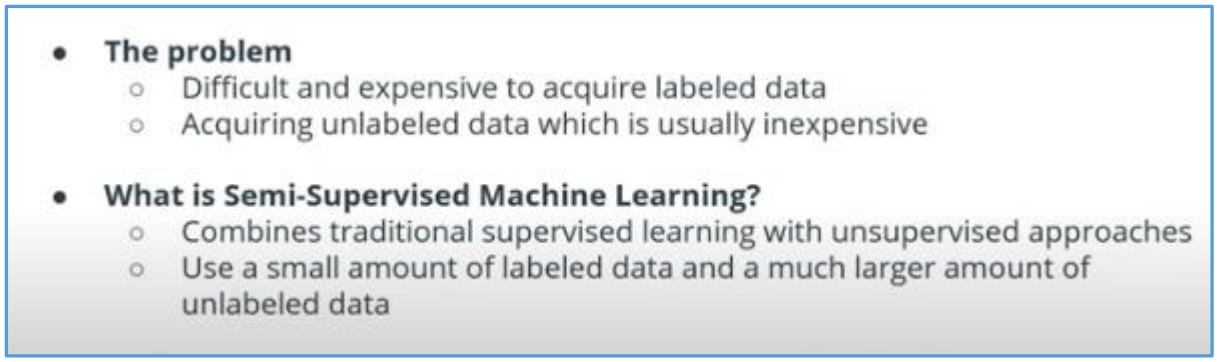




Chapter 22: Semi Supervised Learning

Sometimes fully labelled data cannot be obtained, or is too expensive—but at the same time, it may be possible to get partially labelled data. This is where **semi-supervised learning** is useful.

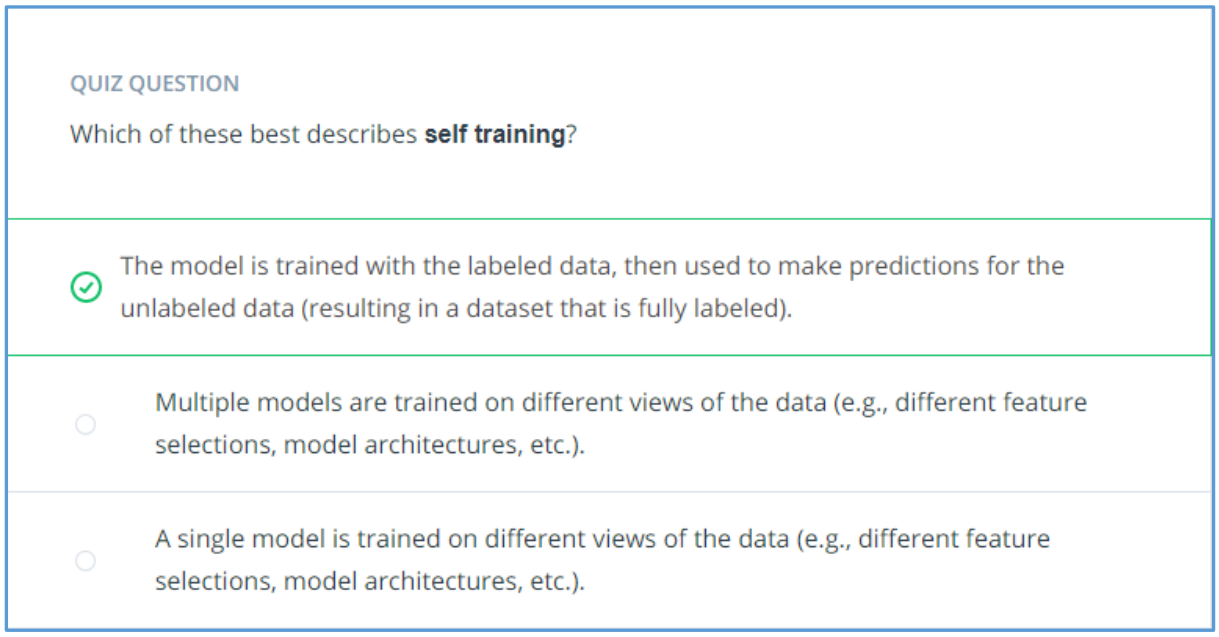
Semi-supervised learning *combines the supervised and unsupervised approaches; typically, it involves having small amounts of labelled data and large amounts of unlabelled data.*



We use the advantage of the labelled part of the training data to make unlabelled part useful.

There are 3 major approaches:

- **Self-Training** – Train the model using labelled data and then use it to make predictions for the unlabelled data. Result is a dataset that is fully, labelled which can be used in supervised learning approach.
- **Multi view Training** – Train multiple models on different views of the data that include various feature selection, parts of training data or various model architectures
- **Self-ensemble Training** – This is similar to Multi view training except we use a single base model and different hyperparameter settings.



Chapter 23: Prelaunch Lab

Prelaunch your lab environment.

NOTE: When you click "Prelaunch Lab", we will begin preparing a lab environment for you. When it is time to access the lab environment, this ensures you will be able to start working. Once your lab is reserved, you can continue on to the next concepts.

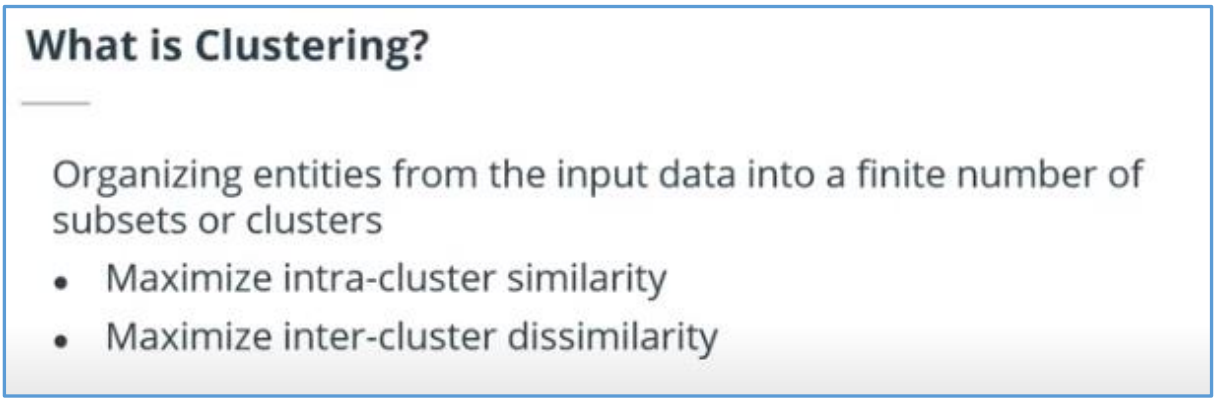
Your lab has been, reserved! Please continue to the next concept.

Chapter 24: Clustering

On this page, we will discuss the unsupervised approach of *clustering* in more detail.

*As the name suggests, **clustering** is the problem of organizing entities from the input data into a finite number of subsets or clusters; the goal is to maximize both intra-cluster similarity and inter-cluster differences.*

Clustering Algorithms



Applications of Clustering Algorithms

- **Personalization and Target Marketing** – Group customers based on similar characteristics, such as spending habits to develop target campaigns.
- **Document Classification –** Plan is to cluster or tag similar document based on its content. This can be, used to improve document search or create digest or similarities.
- **Fraud Detection –** Isolate new cases based on proximity with historical clusters that represent fraudulent behaviour.
- **Medical Imaging –** Clustering can be, used to differentiate different types of tissues in 3D images for many different purposes.
- **City Planning –** Identifying groups of houses according to their house types, value and geographical locations.

Clustering Algorithms

- **Centroid based clustering** – Organizes data into clusters based on the distance of the members from the centroid of the clusters. [k –Means]
- **Density based clustering** – Algorithm clusters those members who are closely, packed together. Advantage is that we can learn clusters of arbitrary shapes.
- **Distribution based clustering** – Assumption is that data has an inherent distribution type such as Normal distribution. Algorithm clusters based on the probability of a member belonging to a particular distribution.
- **Hierarchical clustering** – Algorithm builds a tree of clusters. Best suited for hierarchical data such as taxonomies.

QUESTION 1 OF 2

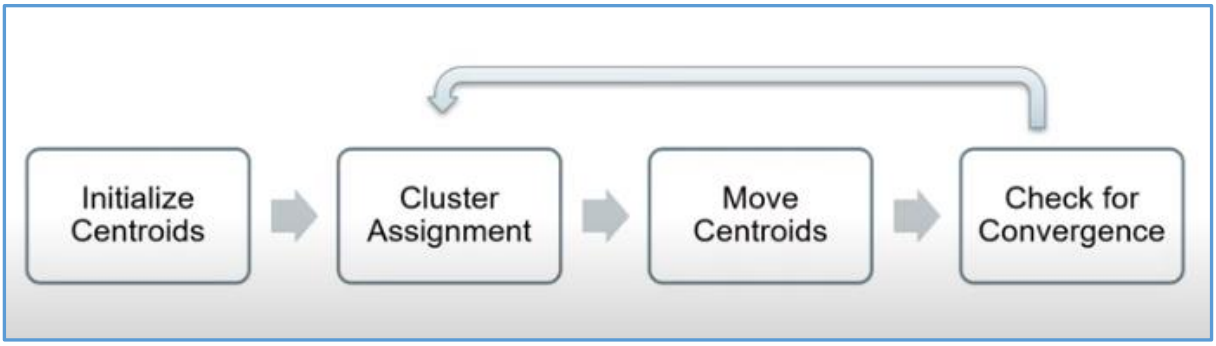
Here are the main types of clustering algorithms we just discussed. Can you match each one with its description?

Submit to check your answer choices!

DESCRIPTION	TYPE OF CLUSTERING
Groups members based on how closely they are packed together; can learn clusters of arbitrary shape.	Density-based clustering
Builds a tree of clusters.	Hierarchical clustering
Groups members based on their distance from the center of the cluster.	Centroid-based clustering
Groups members based on the probability of a member belonging to a particular distribution.	Distribution-based clustering

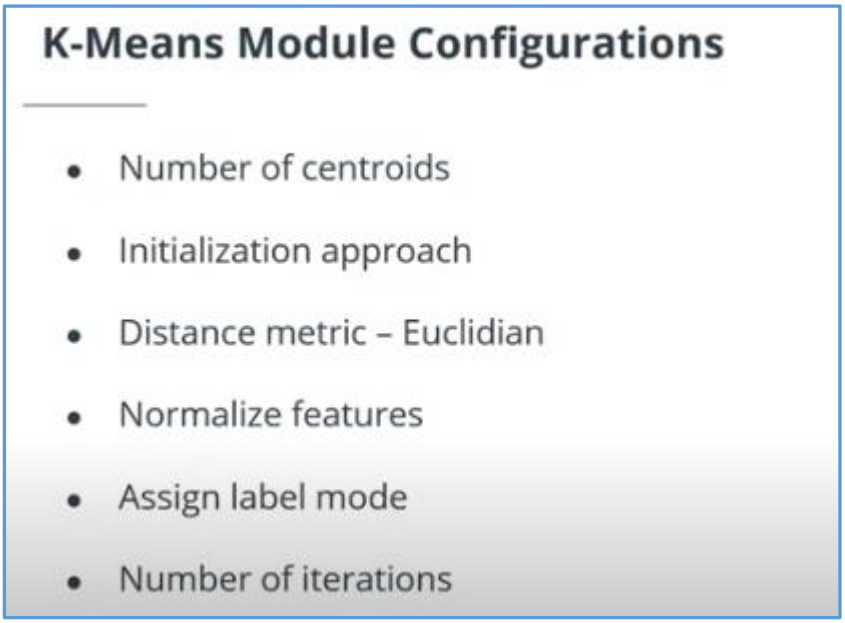
K Means Clustering – This is a centroid based unsupervised clustering algorithm.

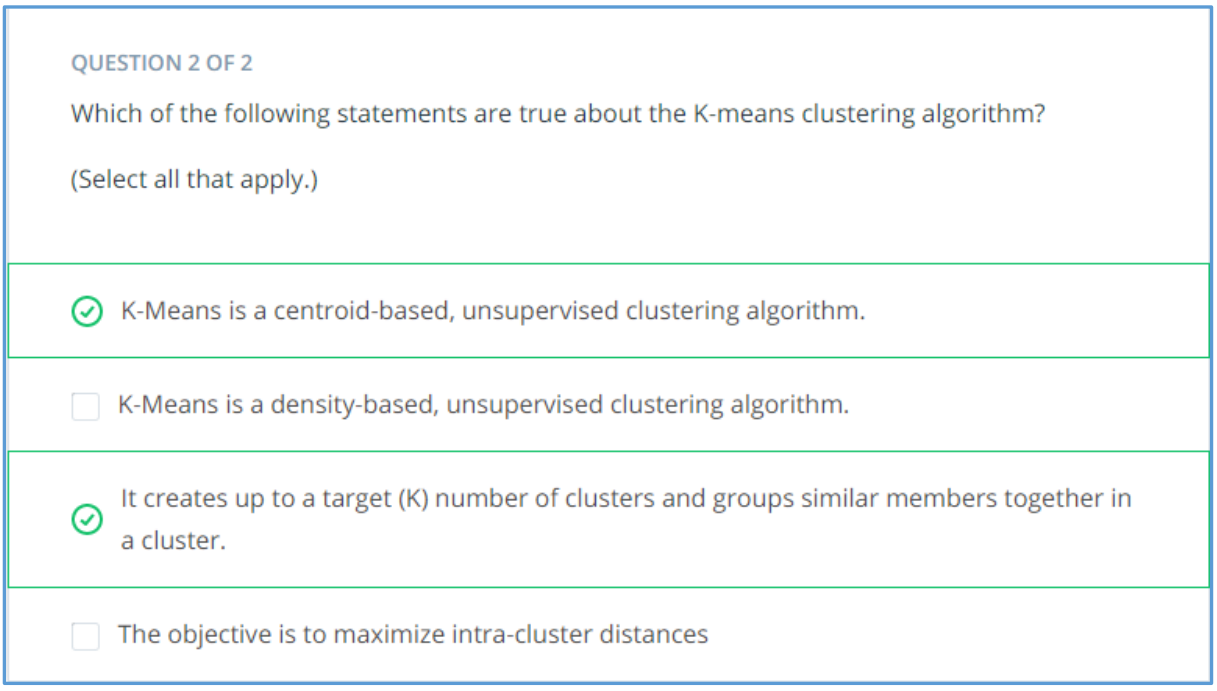
It creates k number of clusters and group similar members together in a cluster. Objective is to minimize inter-cluster distances.[Square error of the distance between the member of the cluster and its center]



- **Initialize Centroids** – Initialize centroid locations. Each centroid represent a cluster.
- **Cluster assignment** – Based on centroid location, members are assigned to a cluster. Assignment of membership is, based on Euclidean distance from the centroids.

- **Move Centroids – k** means will compute the new cluster centroids based on current cluster membership. Due to this, the central locations might change.
- **Check for convergence –** There are different type of convergence criteria's. How much did the centroid location change due to new cluster membership? If the total change of centroid location is less than the tolerance limit, it will assume convergence & stop. Other criteria is, based on fixed number of iterations.
- If convergence criteria is not, met then it will re-iterate Step# 2. Once convergence criteria is met it would stop iterating.





Chapter 25: Lab: Train a Simple Clustering Algorithm

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.

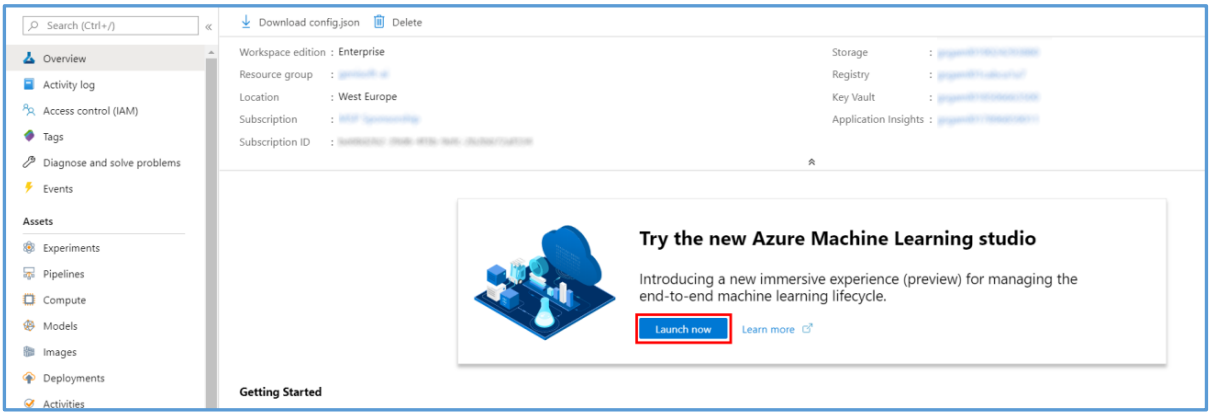
In this lab, we will be using the Weather Dataset that has weather data for 66 different airports in the USA from April to October 2013. We will cluster the dataset into 5 distinct clusters based on key weather metrics, such as visibility, temperature, dew point, wind speed etc. The goal is to group airports with similar weather conditions. We will do all of this from the Azure Machine Learning designer without writing a single line of code.

Exercise 1: Create New Training Pipeline

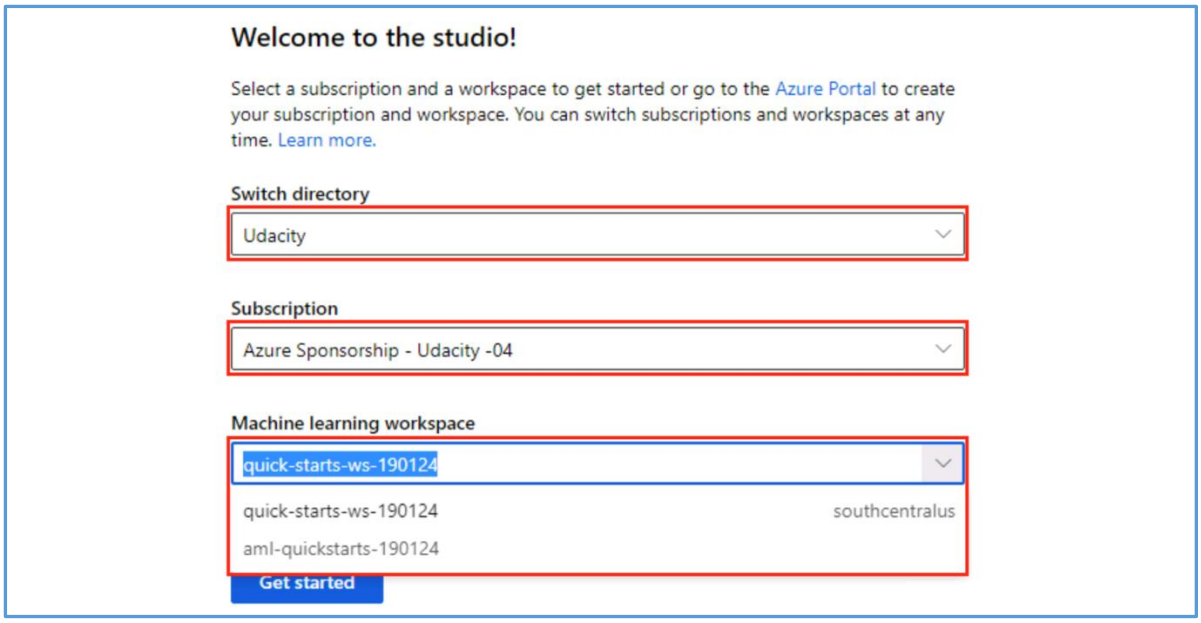
Task 1: Open Pipeline Authoring Editor

1. In Azure portal, open the available machine learning workspace.

2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

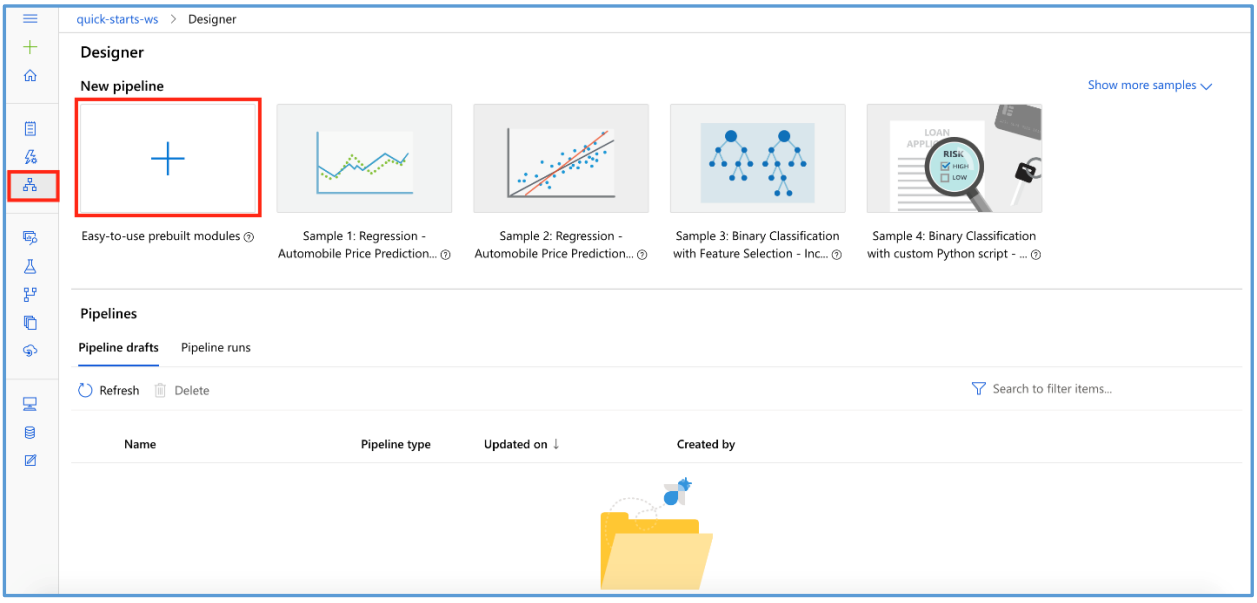


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



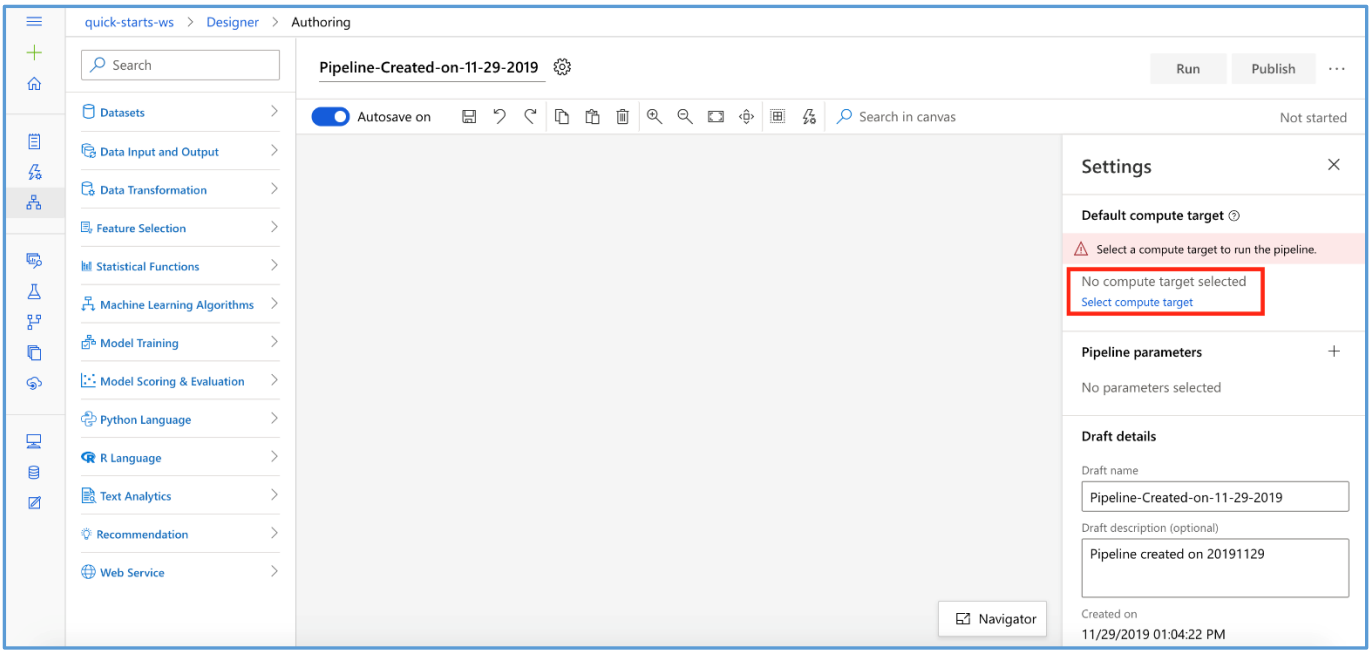
For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it does not matter which) and then click **Get started**.

4. From the studio, select **Designer, +**. This will open a visual pipeline authoring editor.



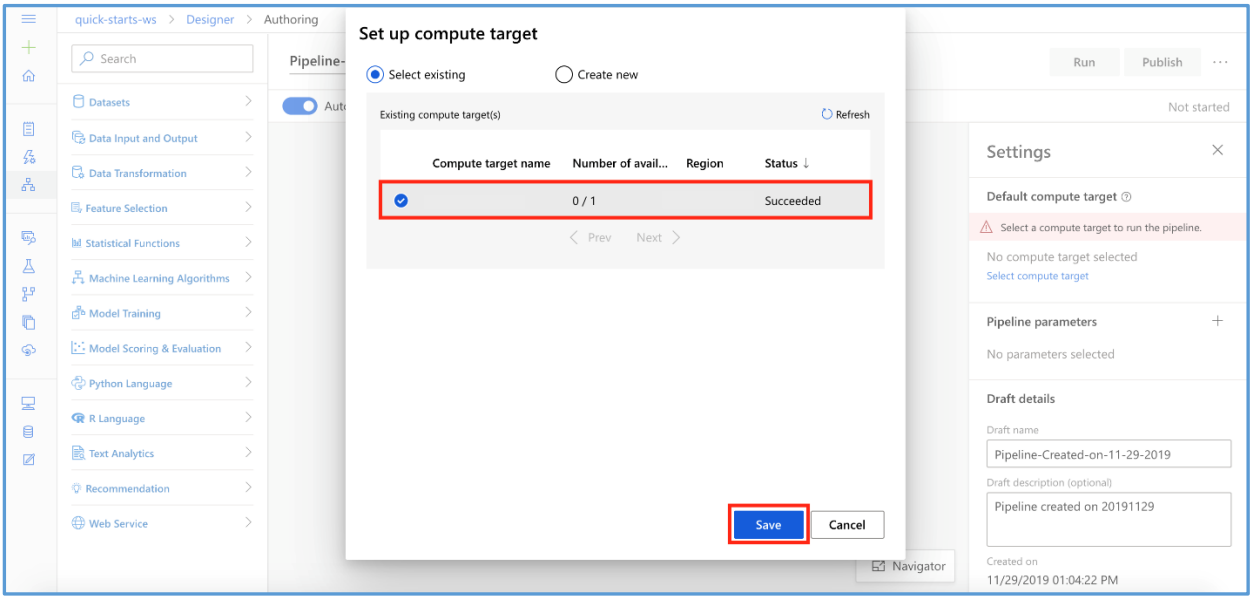
Task 2: Setup Compute Target

1. In the settings panel on the right, select **Select compute target**.



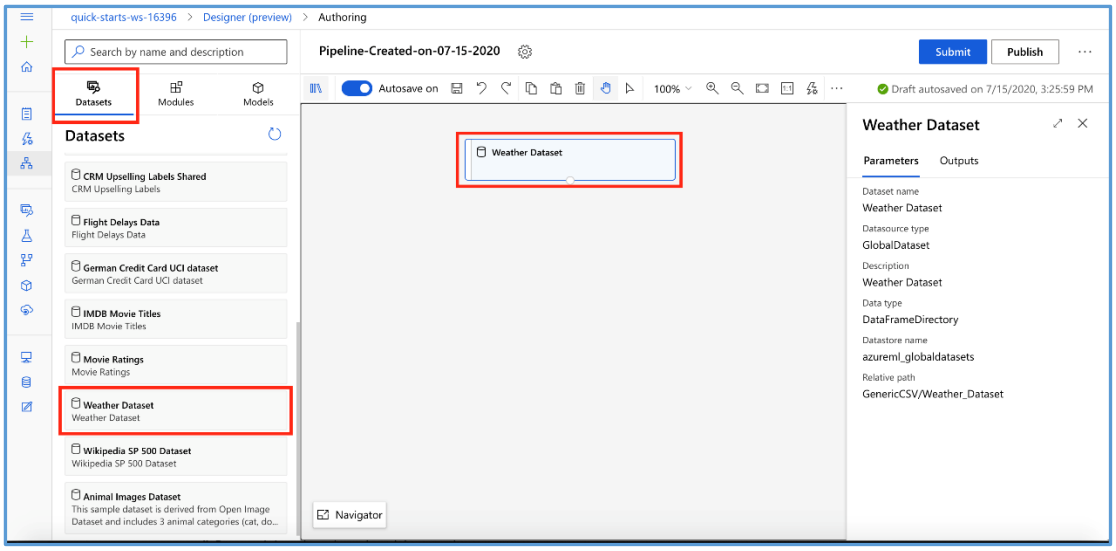
2. In the Set up compute target editor, select the available compute, and then select **Save**.

Note: If you are facing difficulties in accessing pop-up windows or buttons in the user interface, please refer to the Help section in the lab environment.



Task 3: Add Dataset

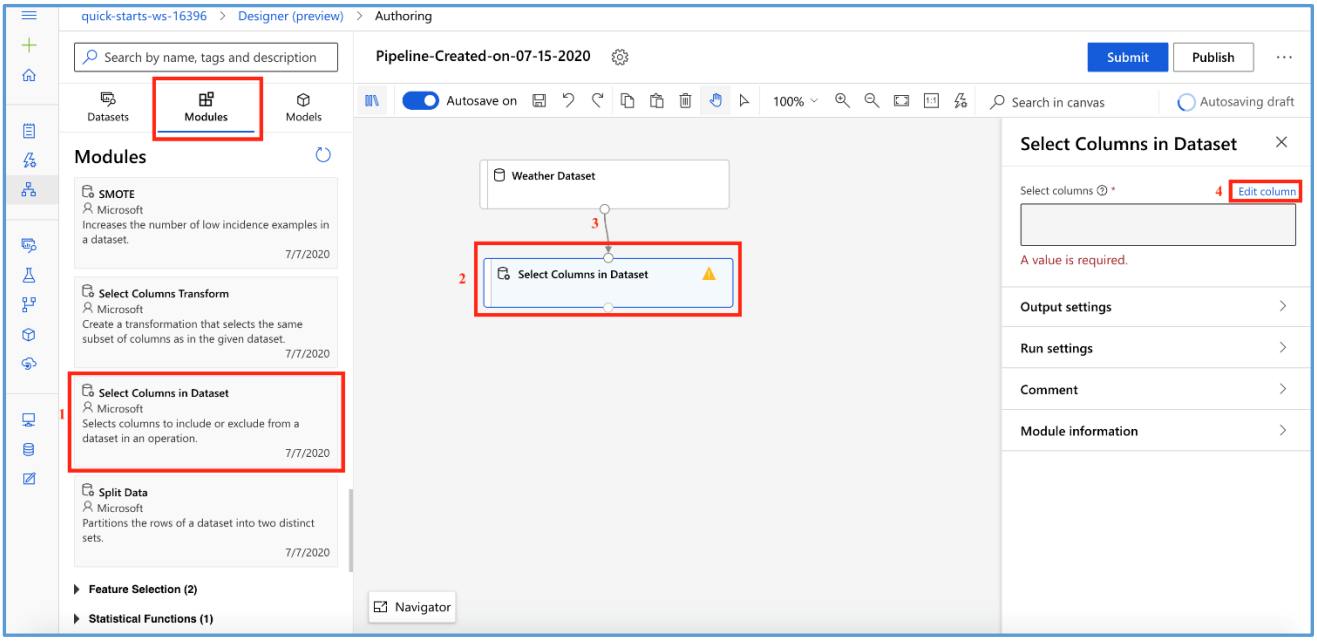
1. Select **Datasets** section in the left navigation. Next, select **Samples, Weather Dataset** and drag and drop the selected dataset on to the canvas.



Task 4: Select Columns in Dataset

1. Select **Modules, Data Transformation** section in the left navigation. Follow the steps outlined below:

    1. Select the **Select Columns in Dataset** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the Weather Dataset module to the Select Columns in Dataset module
    4. Select **Edit column** link to open the Select columns editor



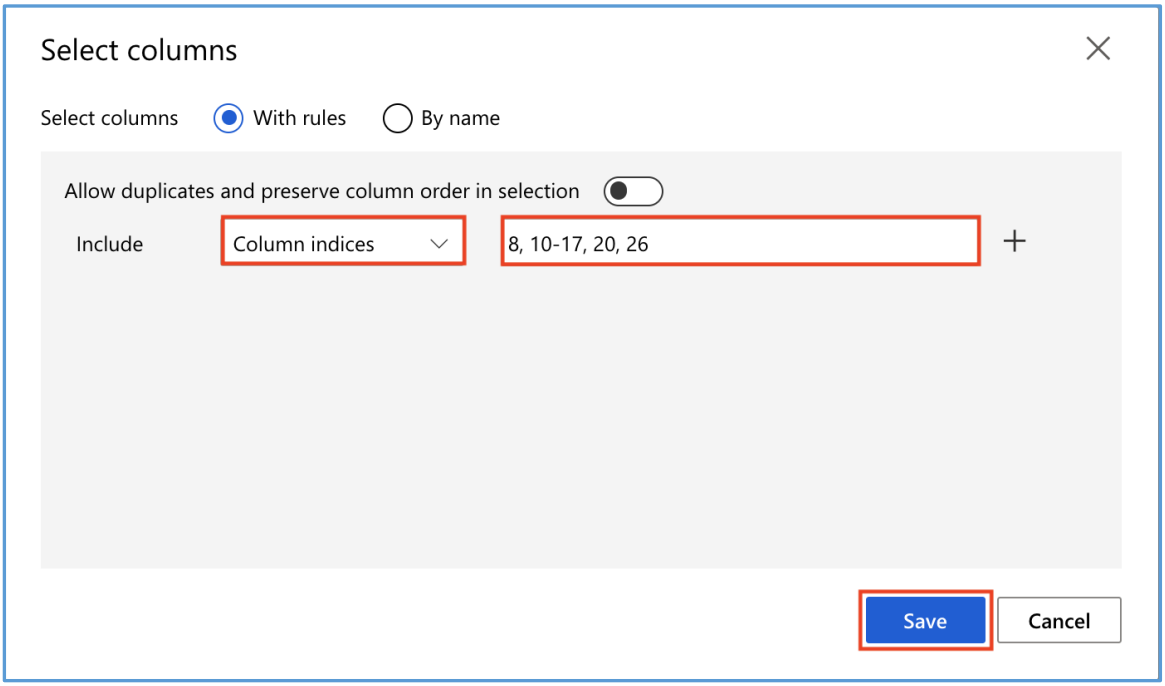
2. *Note that you can submit the pipeline at any point to peek at the outputs and activities. Running pipeline also generates metadata that is available for downstream activities such selecting column names from a list in selection dialogs.*

3. In the Select columns editor, follow the steps outlined below:

    1. Include: **Column indices**

    2. Provide column indices: **8, 10-17, 20, 26**

    3. Select **Save**



Task 5: Split Data

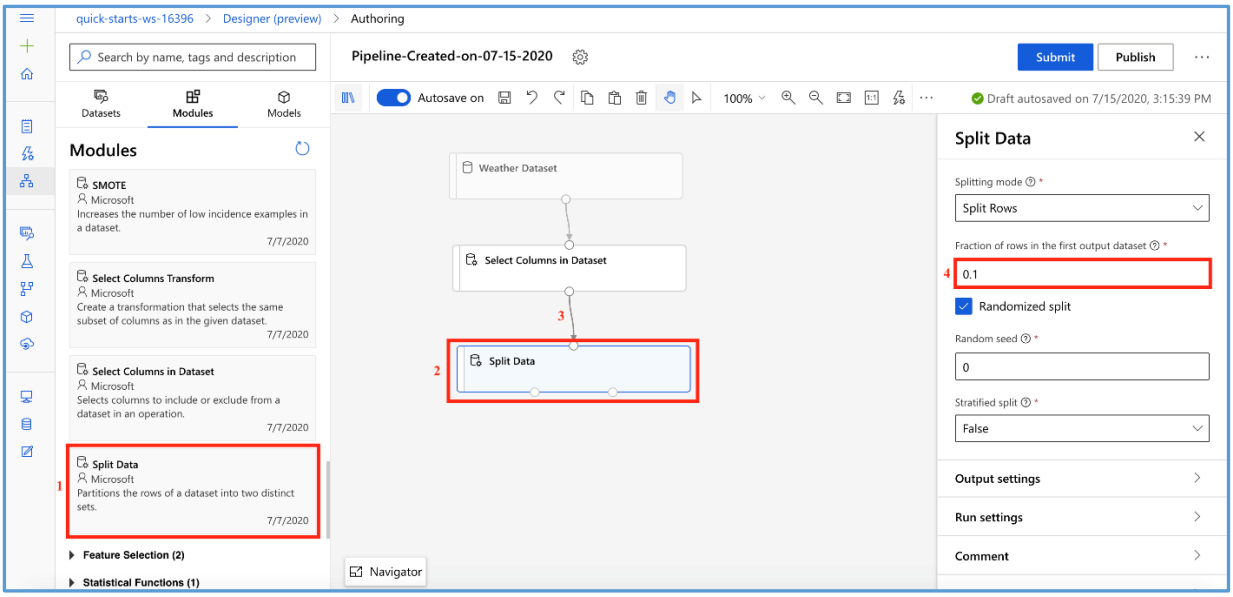
1. Select **Data Transformation** section in the left navigation. Follow the steps outlined below:

    1. Select the **Split Data** prebuilt module

    2. Drag and drop the selected module on to the canvas

    3. Connect the Select Columns in Dataset module to the Split Data module

    4. Fraction of rows in the first output dataset: **0.1**



Task 6: Normalize Data

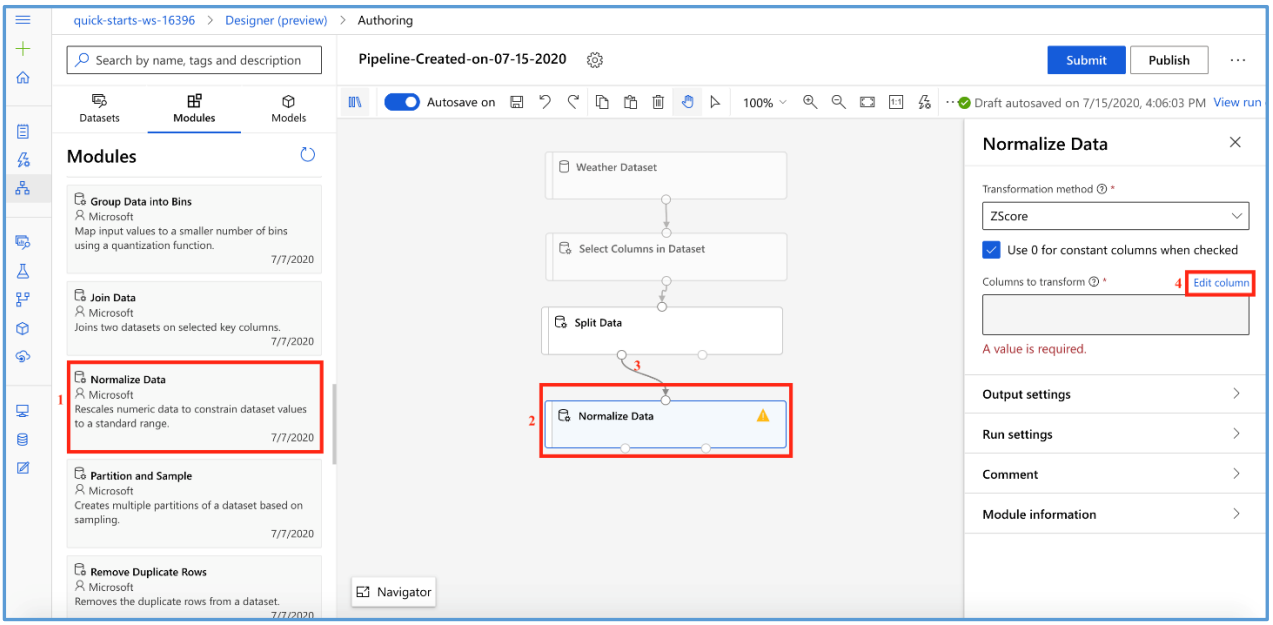
1. Select **Data Transformation** section in the left navigation. Follow the steps outlined below:

    1. Select the **Normalize Data** prebuilt module

    2. Drag and drop the selected module on to the canvas

    3. Connect the Split Data module to the Normalize Data module

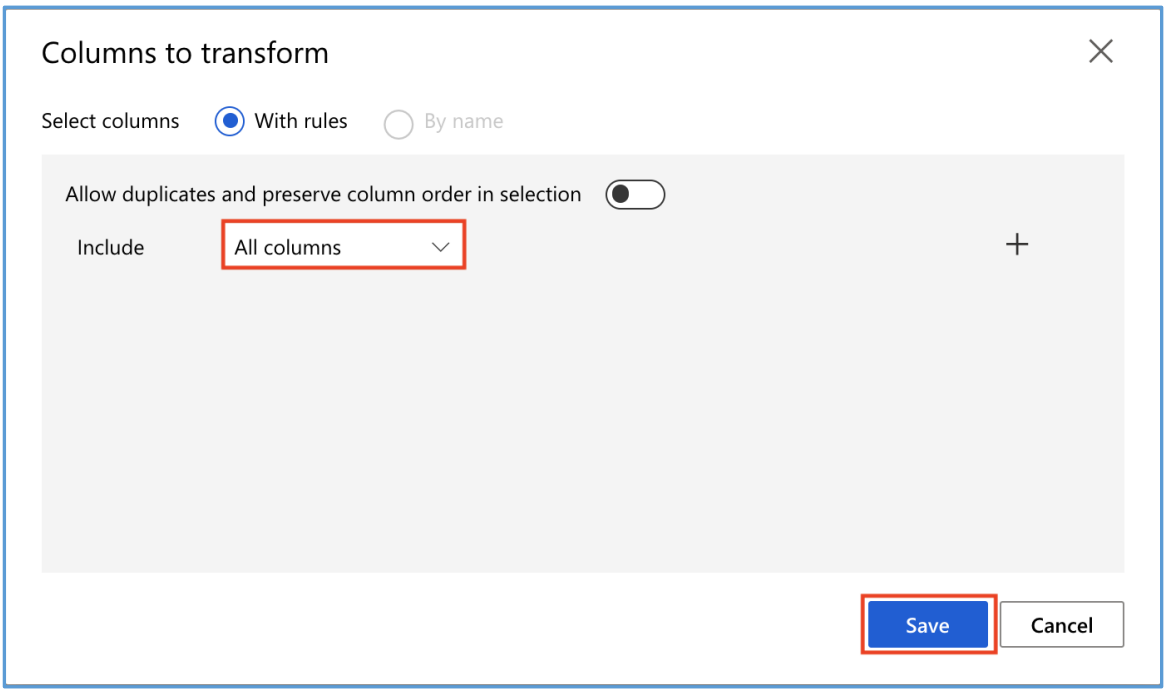
    4. Select **Edit column** link to open the Columns to transform editor



2. In the Columns to transform editor, follow the steps outlined below:

1. Include: **All columns**

2. Select **Save**



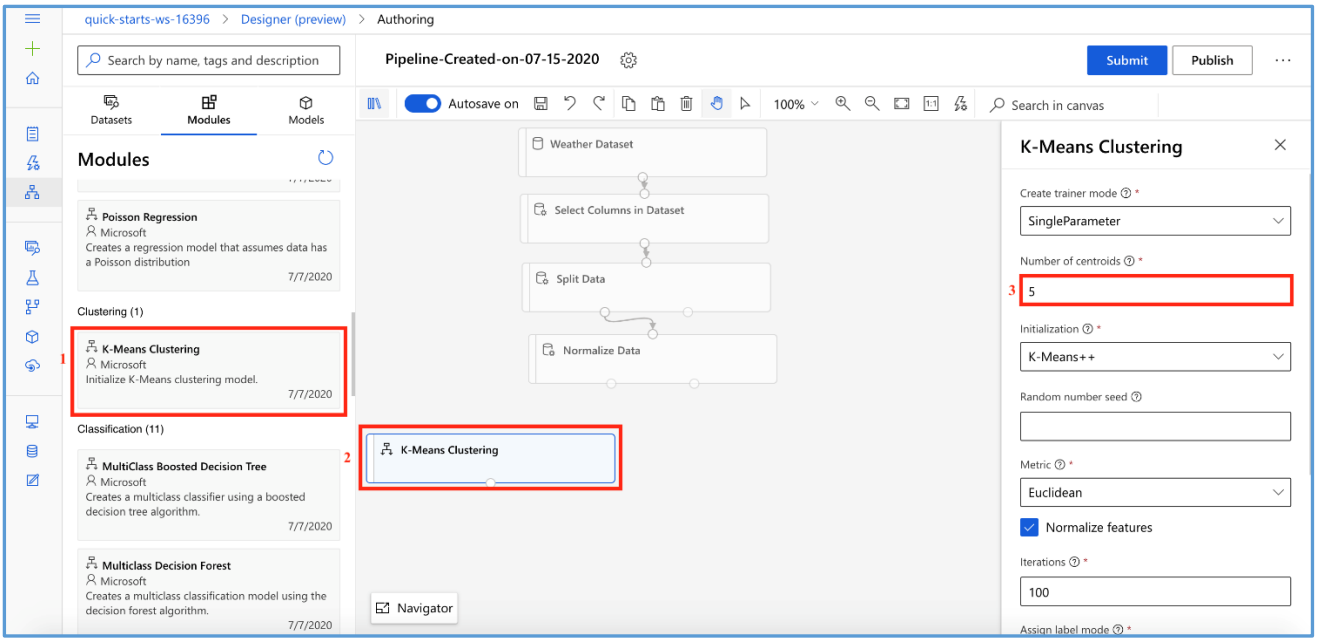
Task 7: Initialize K-Means Clustering Model

1. Select **Machine Learning Algorithms** section in the left navigation. Follow the steps outlined below:

    1. Select the **K-Means Clustering** prebuilt module

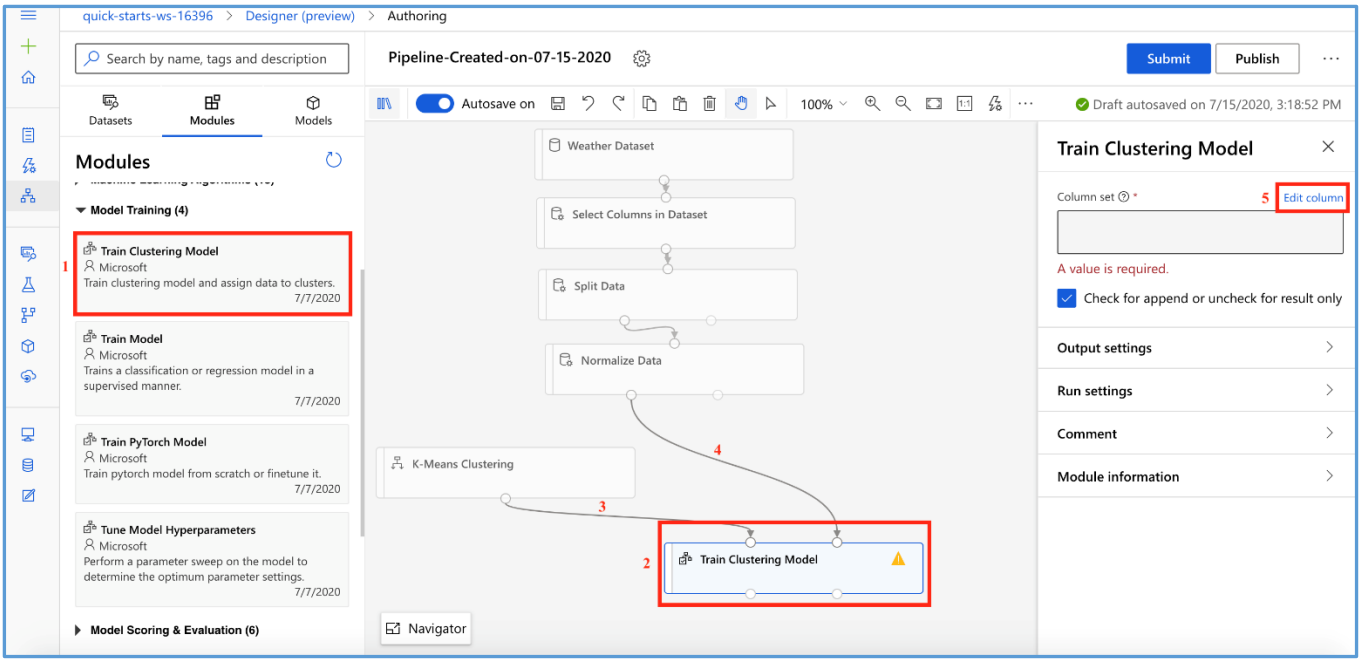
    2. Drag and drop the selected module on to the canvas

    3. Number of centroids: **5**



Task 8: Setup Train Clustering Model Module

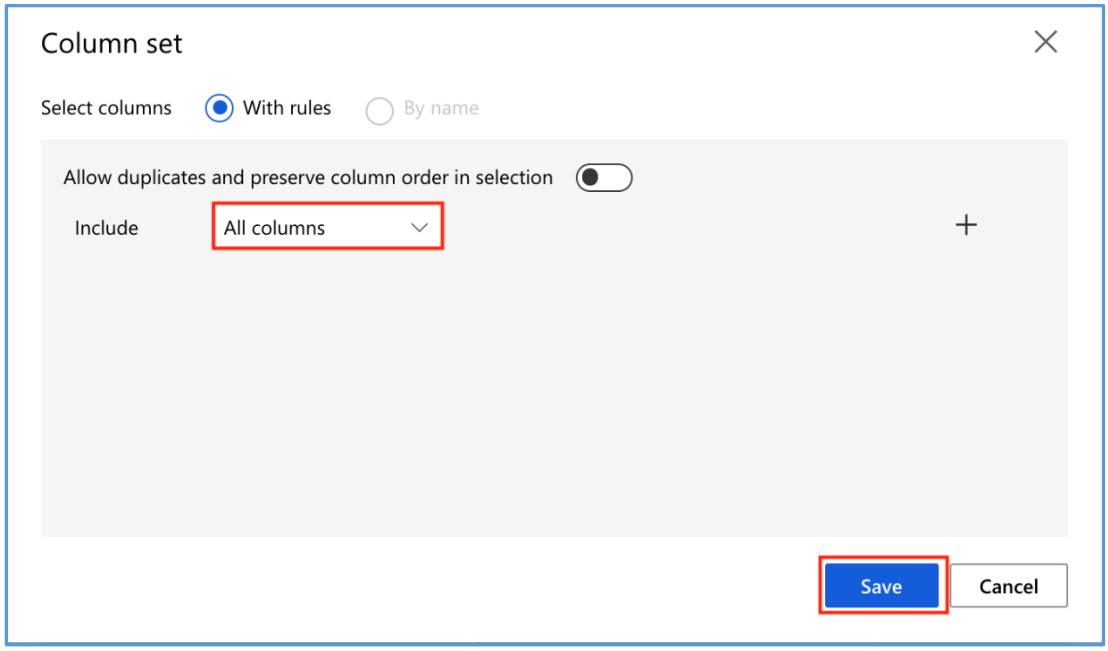
1. Select **Model Training** section in the left navigation. Follow the steps outlined below:
    1. Select the **Train Clustering Model** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the K-Means Clustering module to the first input of the Train Clustering Model module
    4. Connect the first output of the Normalize Data module to the second input of the Train Clustering Model module
    5. Select the **Edit column** link to open the Column set editor



2. In the Columns set editor, follow the steps outlined below:

    1. Include: **All columns**

    2. Select **Save**



Task 9: Setup Assign Data to Clusters Module

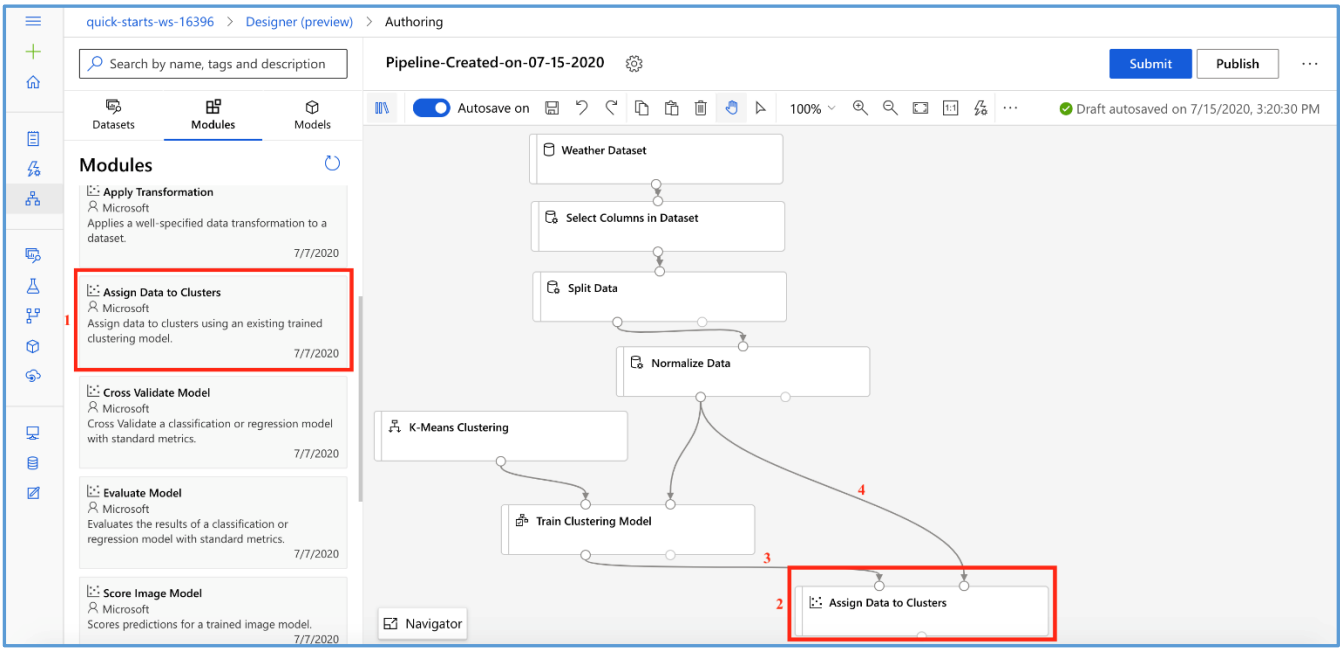
1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:

    1. Select the **Assign Data to Clusters** prebuilt module

2. Drag and drop the selected module on to the canvas

3. Connect the first output of the Train Clustering Model module to the first input of the Assign Data to Clusters module

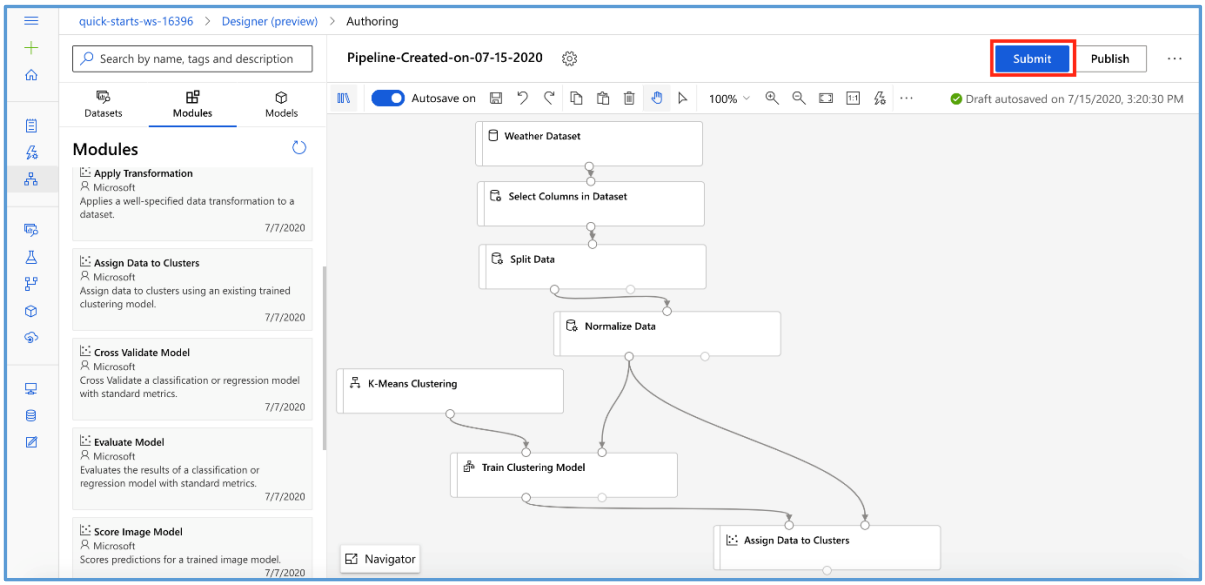
4. Connect the first output of the Normalize Data module to the second input of the Assign Data to Clusters module



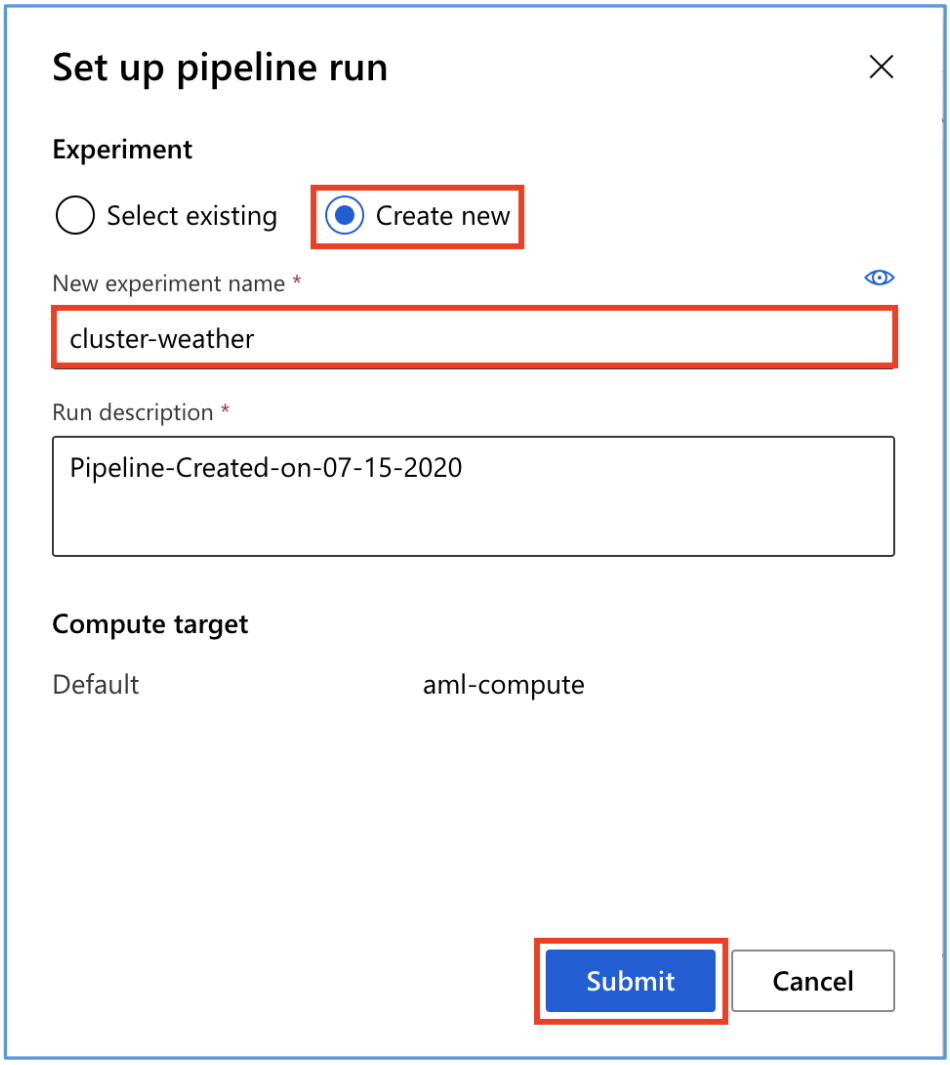
Exercise 2: Submit Training Pipeline

Task 1: Create Experiment and Submit Pipeline

1. Select **Submit** to open the Setup pipeline run editor.



2. In the Setup pipeline run editor, select **Experiment, Create new** and provide New experiment name: **cluster-weather**, and then select **Submit**.



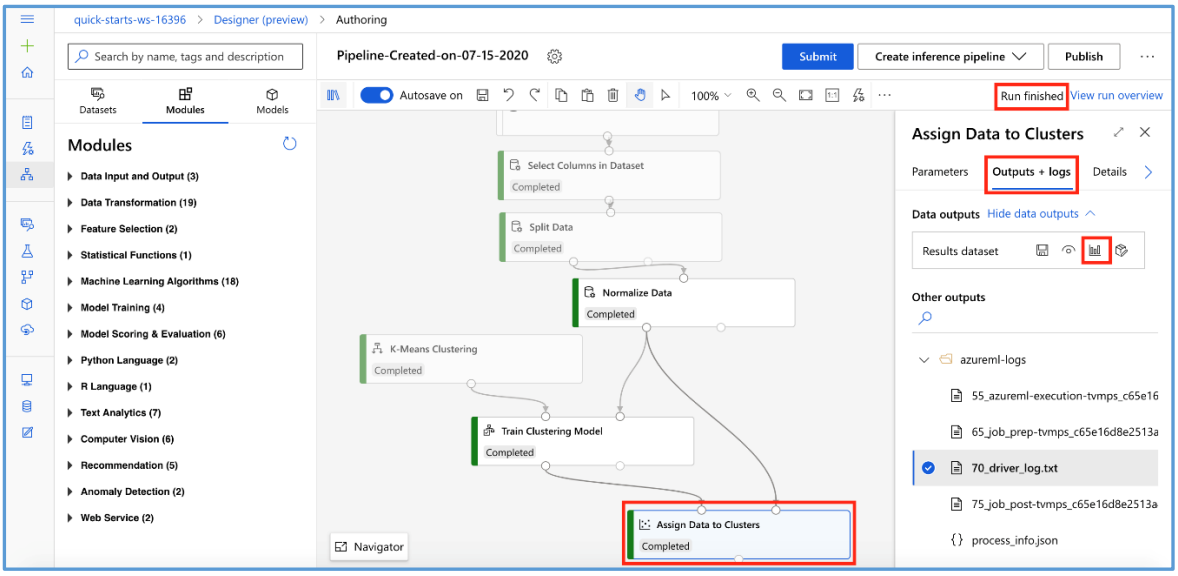
3. Wait for pipeline run to complete. It will take around **10 minutes** to complete the run.

4. While you wait for the model training to complete, you can learn more about the K-Means Clustering algorithm used in this lab by selecting K-Means Clustering.

Exercise 3: Visualize the Clustering Results

Task 1: Open the Visualization Dialog

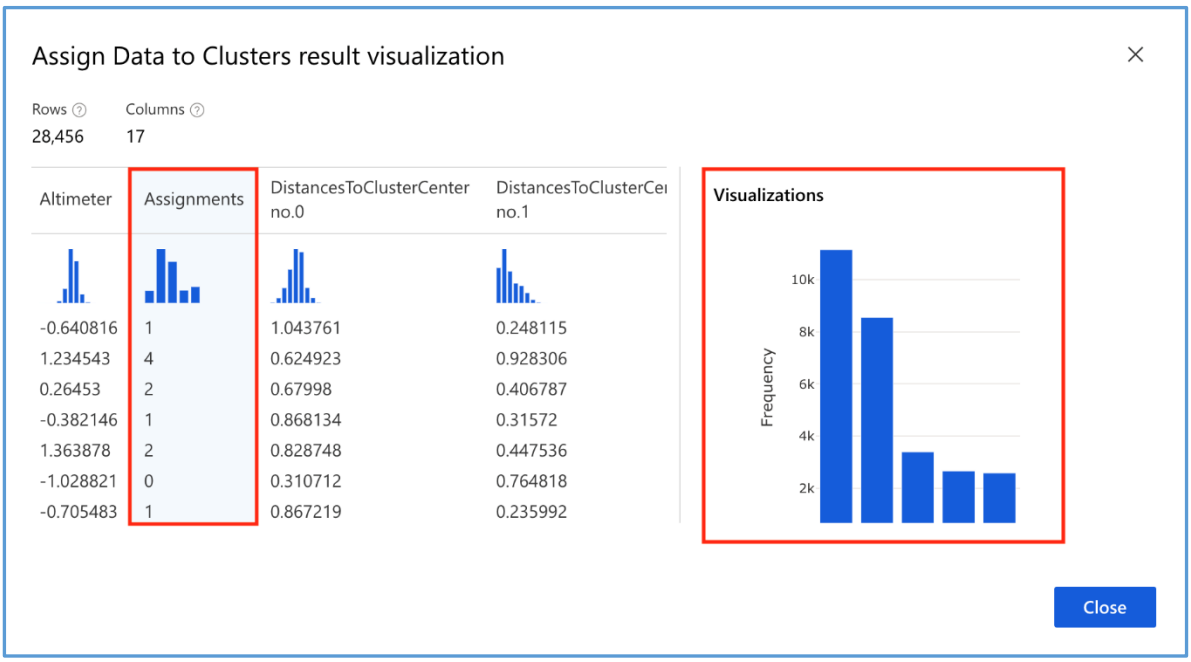
1. Select **Assign Data to Clusters, Outputs + logs, Visualize** to open the Assign Data to Clusters result visualization dialog.



Task 2: Evaluate Clustering Results

1. Scroll to the right and select **Assignments** column.

2. In the right-hand-side pane, scroll down to the **Visualizations** section.



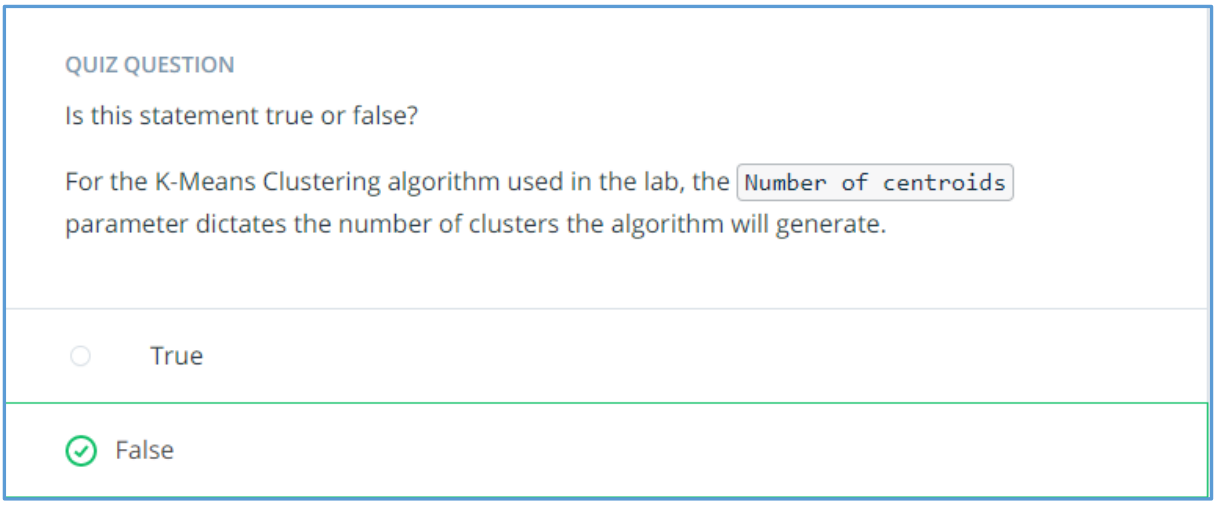
3. From the results, you can observe that each row (input) in the dataset is assigned to one of the 5 clusters: 0, 1, 2, 3, or 4. You can also see for each input, how far that input was from the various centroids. The cluster assignment is, made based on the shortest distance between the input and cluster centroids. From the bar graph, you can see the frequency distribution of all the inputs across the 5 clusters.

Next Steps

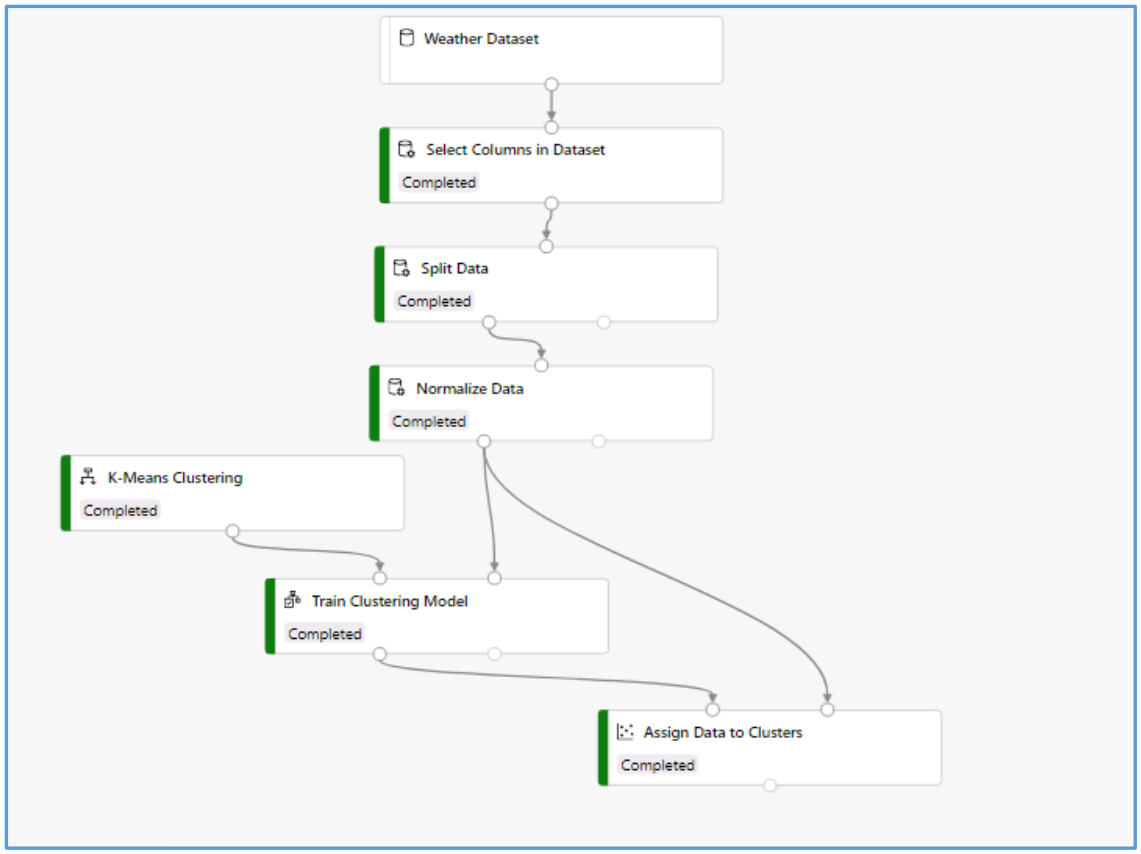
Congratulations! You have trained and evaluated your first clustering algorithm. You can continue to experiment in the environment but are free to close the lab environment tab and return to the Udacity portal to continue with the lesson.

Chapter 26: Walkthrough: Train a Simple Clustering Algorithm

- Weather dataset is being used which has weather data for 66 different airports from Apr-19 till Oct-19.
- We will cluster the datasets in 5 distinct clusters based on key weather metrics such as Visibility, Temperature, Dew point, Wind speed etc.
- Group airports with similar weather conditions.
- Create a new pipeline.
- Setup compute target.



The model is not, guaranteed to produce exactly this number of clusters. The algorithm starts with this number of data centroids and iterates to find the optimal configuration.



Chapter 27: Lesson Summary

This lesson covered two of Machine Learning's fundamental approaches: **supervised** and **unsupervised** learning.

First, we learned about **supervised learning**. Specifically, we learned:

- More about *classification* and *regression*, two of the most representative supervised learning tasks
- Some of the major *algorithms* involved in supervised learning, as well as how to evaluate and compare their performance
- How to use the Designer in Azure Machine Learning Studio to build pipelines that train and compare the performance of both binary and multi-class classifiers.
- How to use *automated machine learning* to automate the training and selection of classifiers and regressors, and how to use the Designer in Azure Machine Learning Studio to create automated Machine Learning experiments
-

Next, the lesson focused on **unsupervised learning**, including:

- Its most representative learning task, *clustering*
- How unsupervised learning can address challenges like lack of labelled data, the curse of dimensionality, overfitting, feature engineering, and outliers
- An introduction to *representation learning*
- How to train your first clustering model in Azure Machine Learning Studio