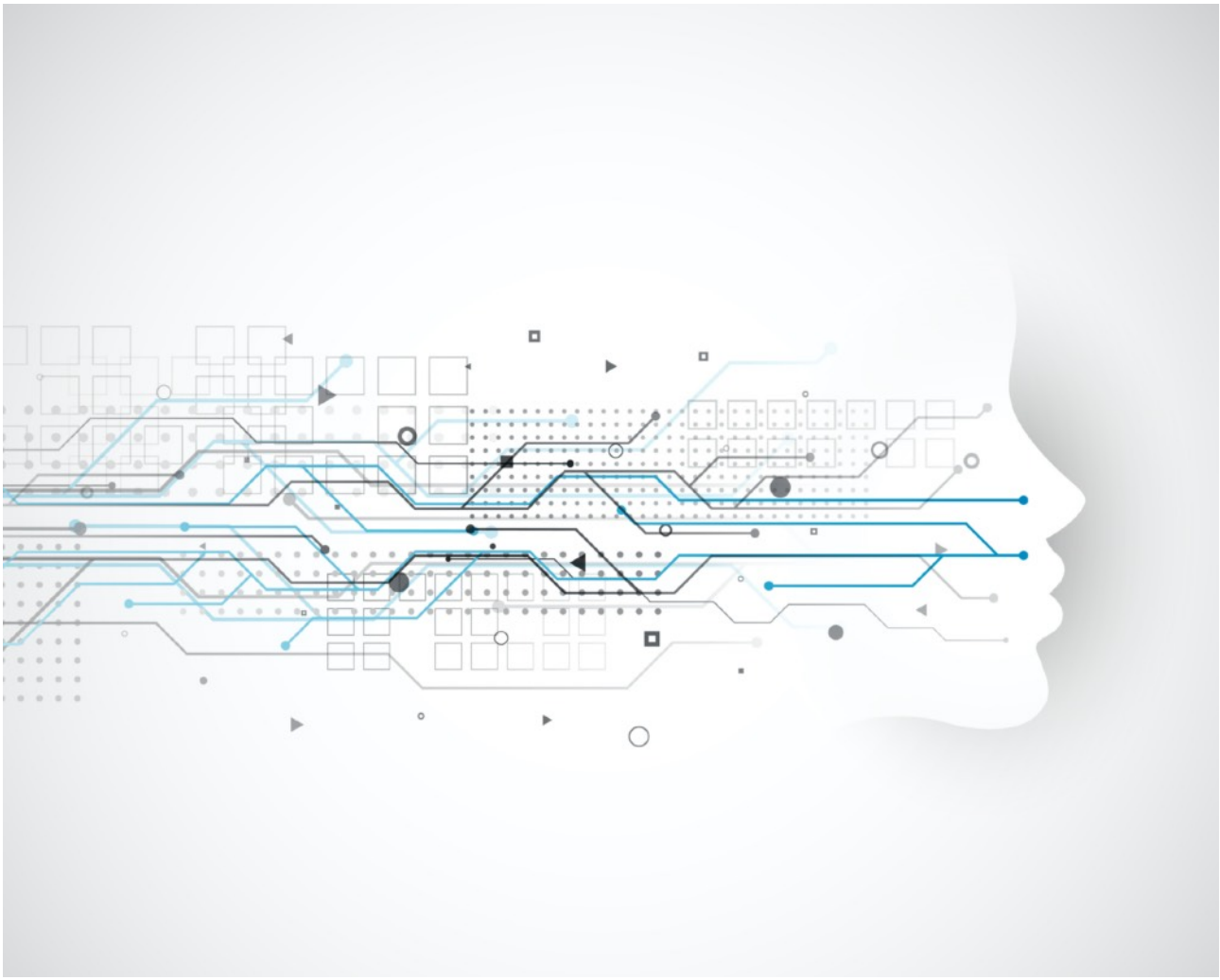
Microsoft Scholarship Foundation course Nanodegree Program

Labs walkthrough compilation



"Designed by starline / Freepik"

Lab 1 - Train a Linear Regression Model

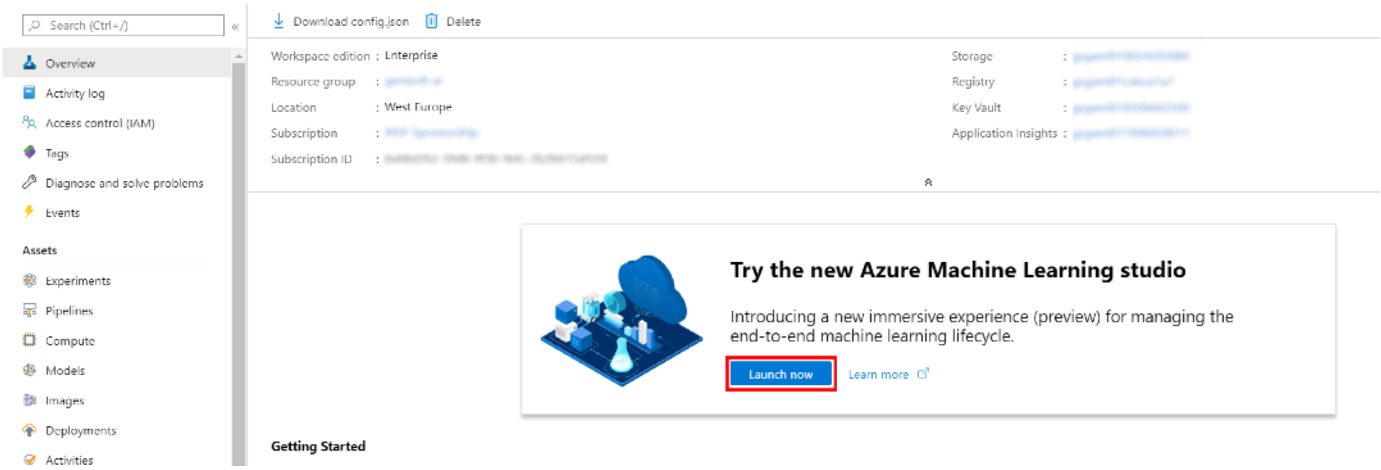
Overview

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 a subset of NYC Taxi & Limousine Commission - green taxi trip records available from Azure Open Datasets. The data is enriched with holiday and weather data. Based on the enriched dataset, we will learn to use the Azure Machine Learning Graphical Interface to process data, build, train, score, and evaluate a regression model to predict NYC taxi fares. To train the model, we will create Azure Machine Learning Compute resource. We will do all of this from the Azure Machine Learning designer without writing a single line of code.

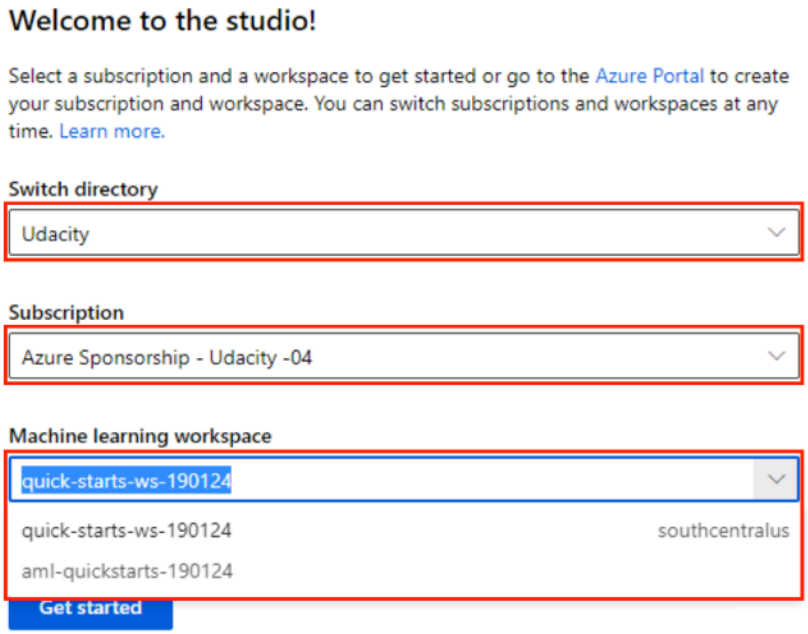
Exercise 1: Register Dataset with Azure Machine Learning studio

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.

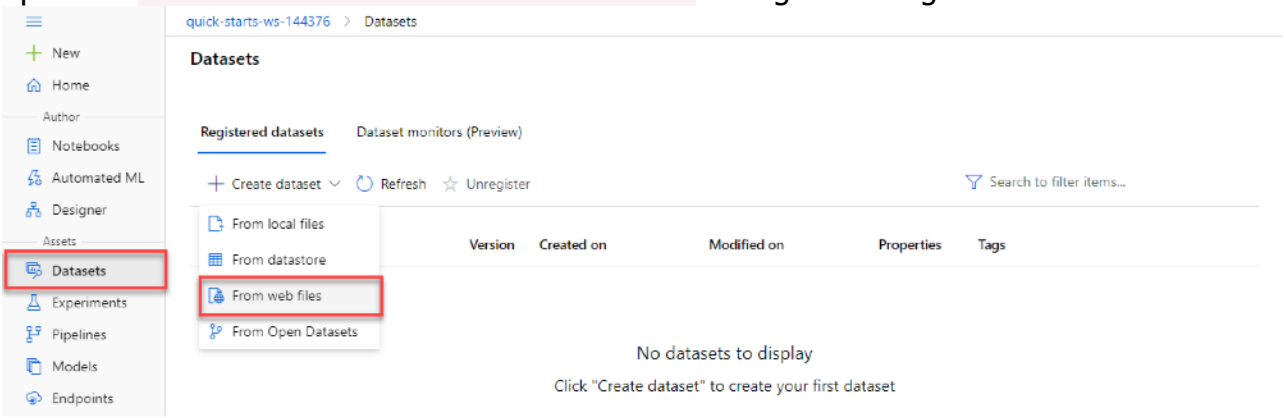


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



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**.

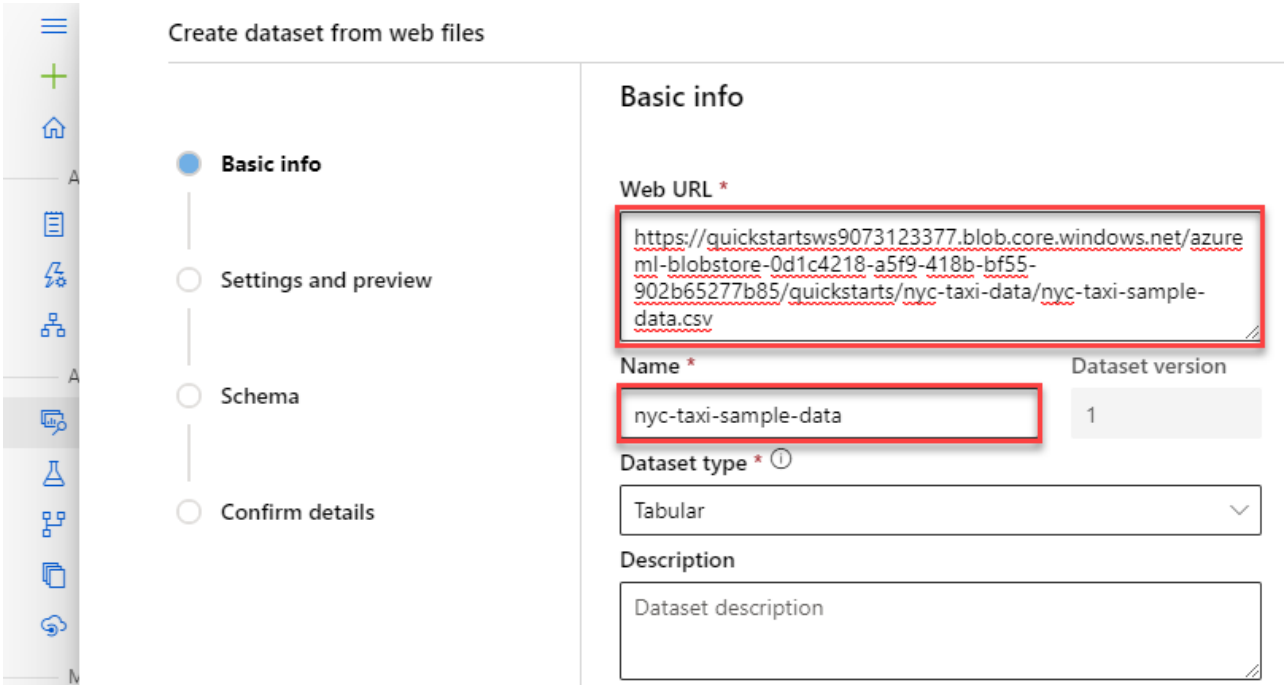
4. From the studio, select **Datasets, + Create dataset, From web files**. This will open the `Create dataset from web files` dialog on the right.



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

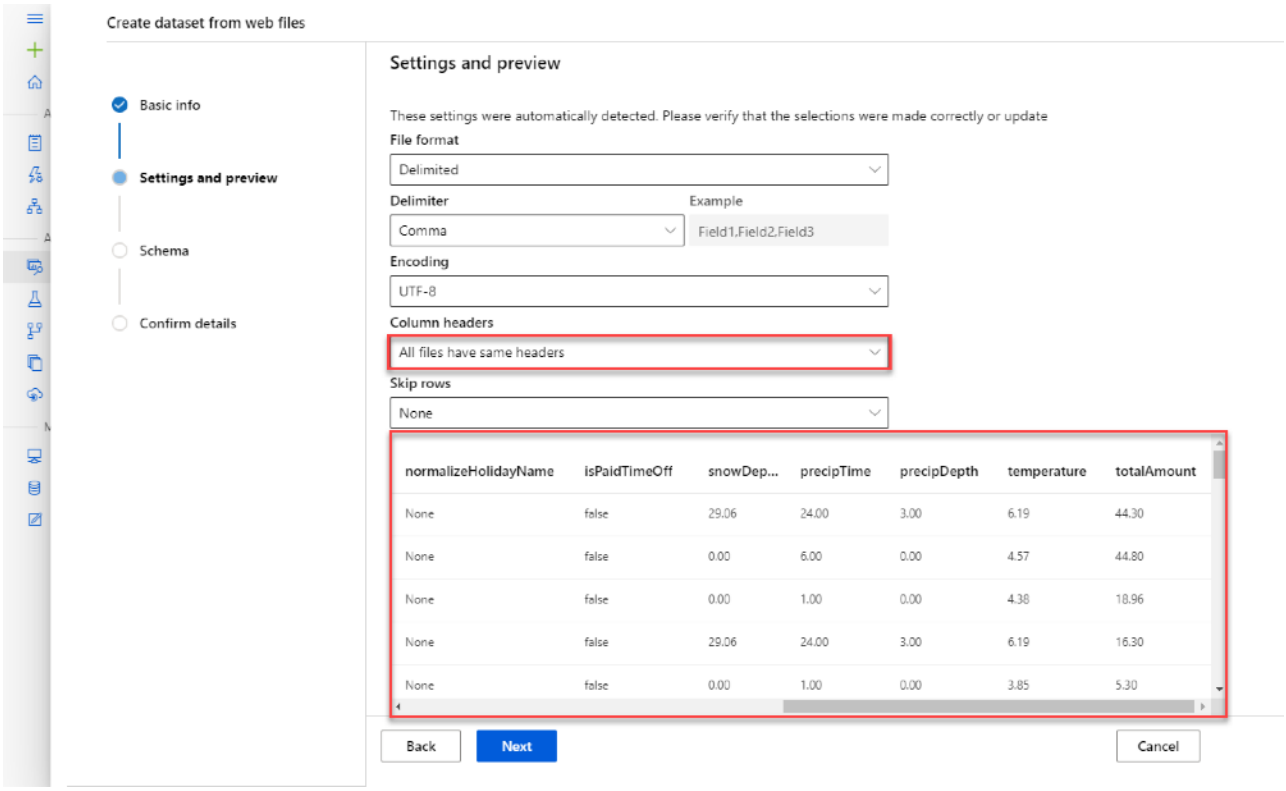
   `https://introtomlsampledata.blob.core.windows.net/data/nyc-taxi/nyc-taxi-sample-data.csv`

6. Provide `nyc-taxi-sample-data` as the Name, leave the remaining values at their defaults and select **Next**.



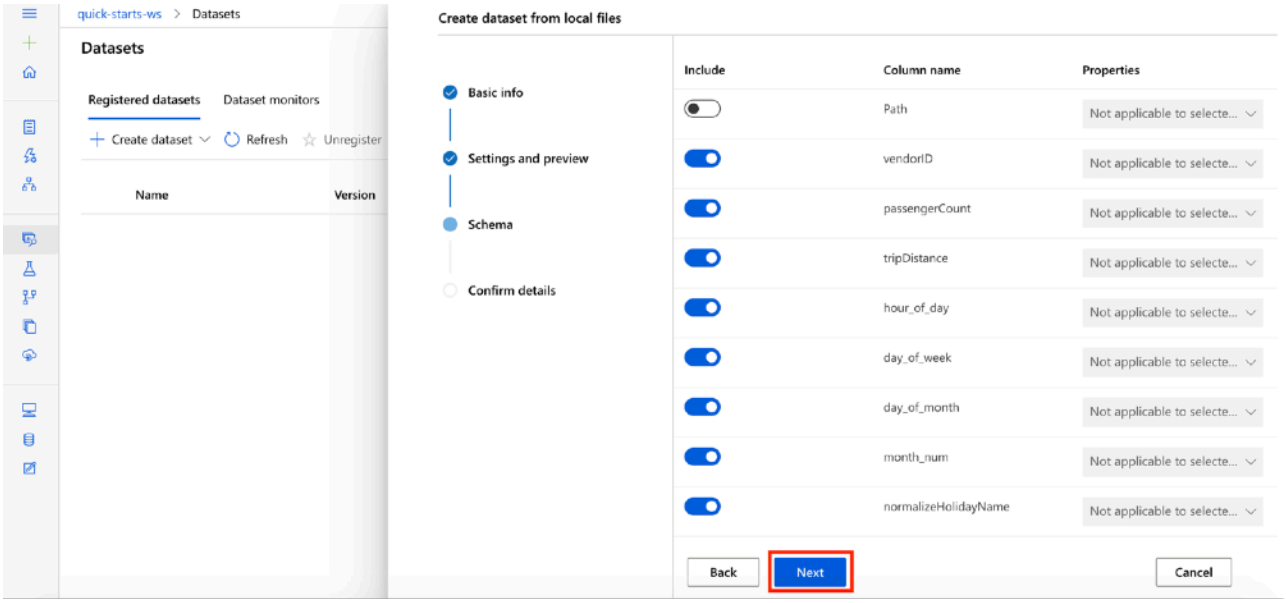
Task 2: Preview Dataset

1. On the Settings and preview panel, set the column headers drop down to `All files have same headers`.
2. Scroll the data preview to right to observe the target column: `totalAmount`. After you are done reviewing the data, select **Next**



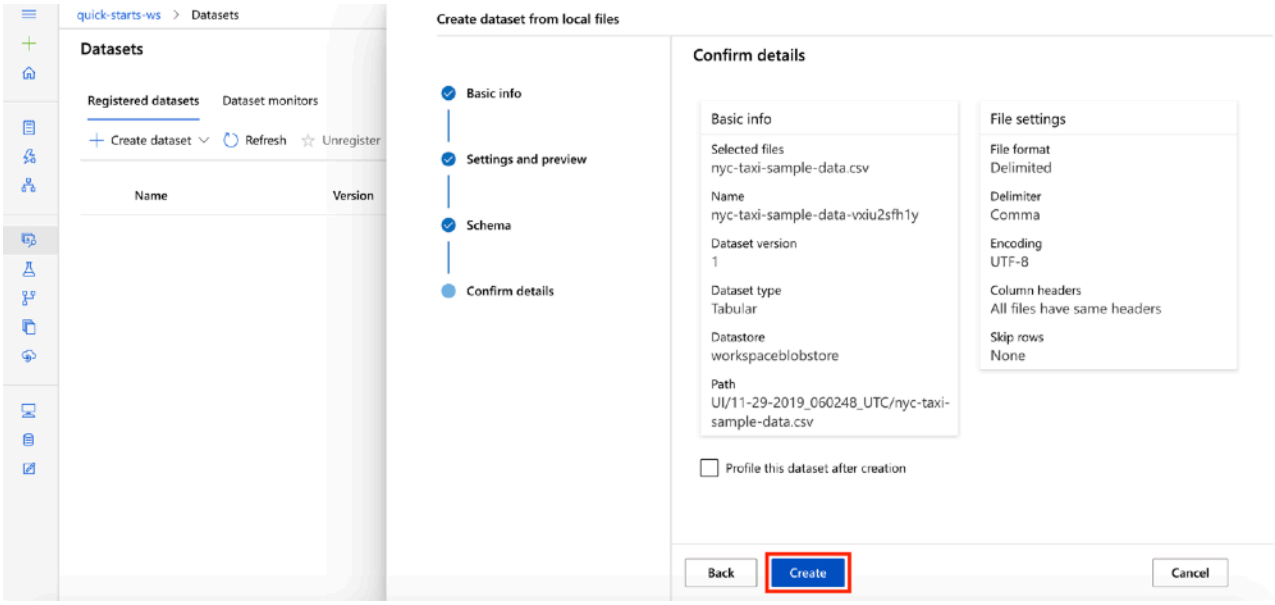
Task 3: Select Columns

1. Select columns from the dataset to include as part of your training data. Leave the default selections and select **Next**



Task 4: Create Dataset

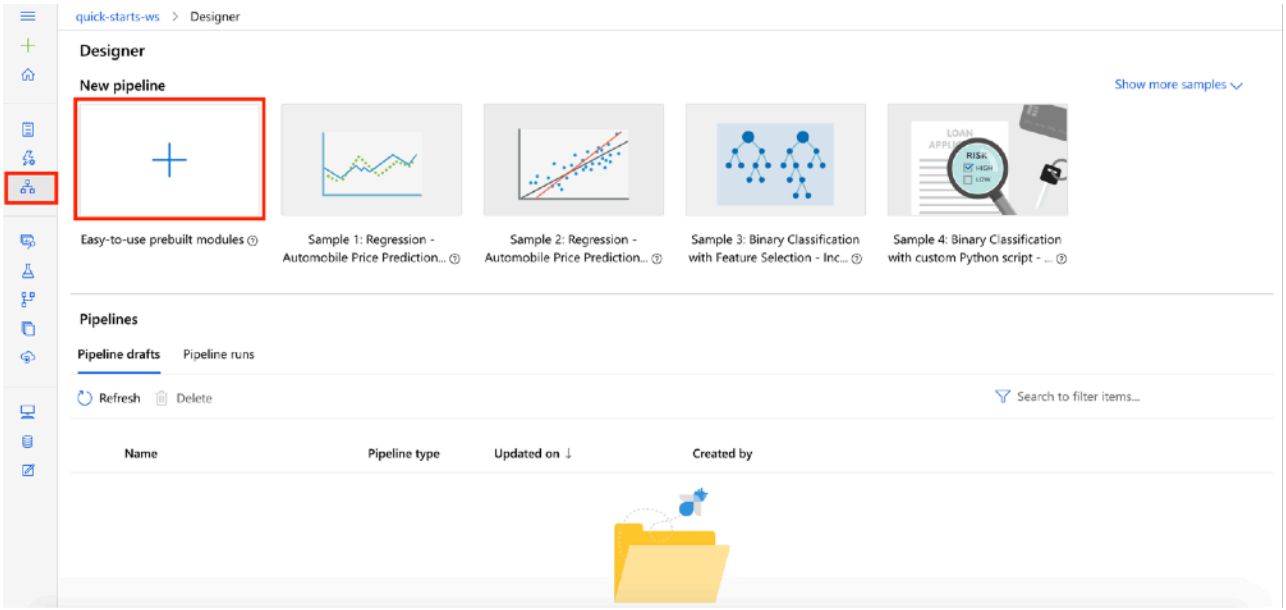
1. Confirm the dataset details and select **Create**



Exercise 2: Create New Training Pipeline

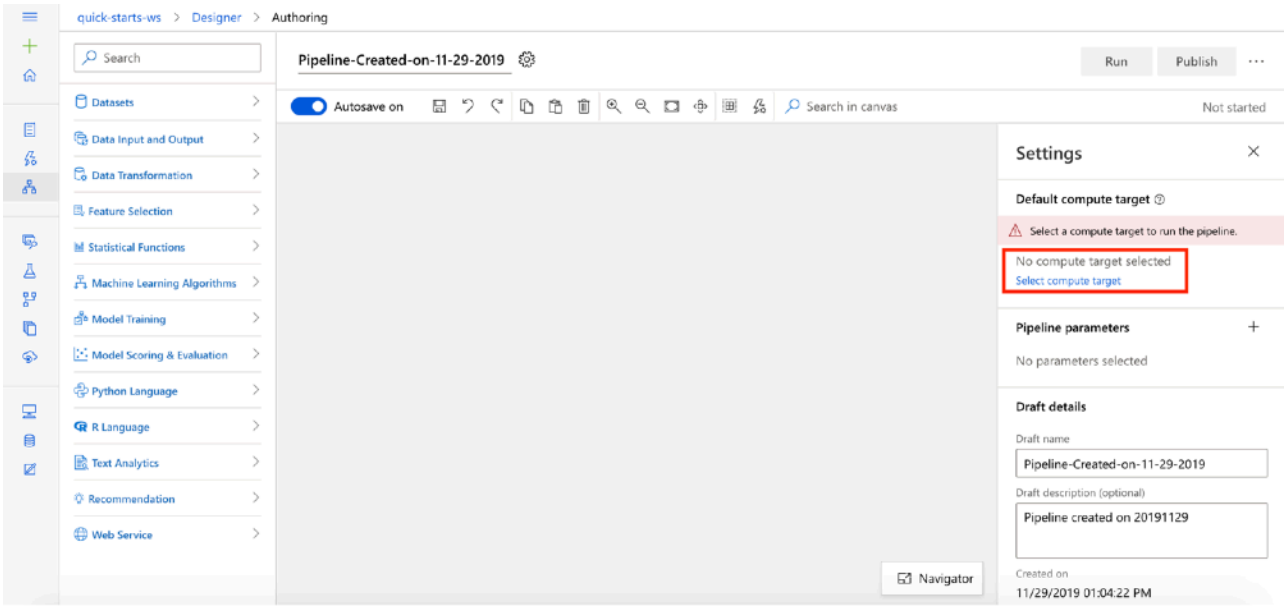
Task 1: Open Pipeline Authoring Editor

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



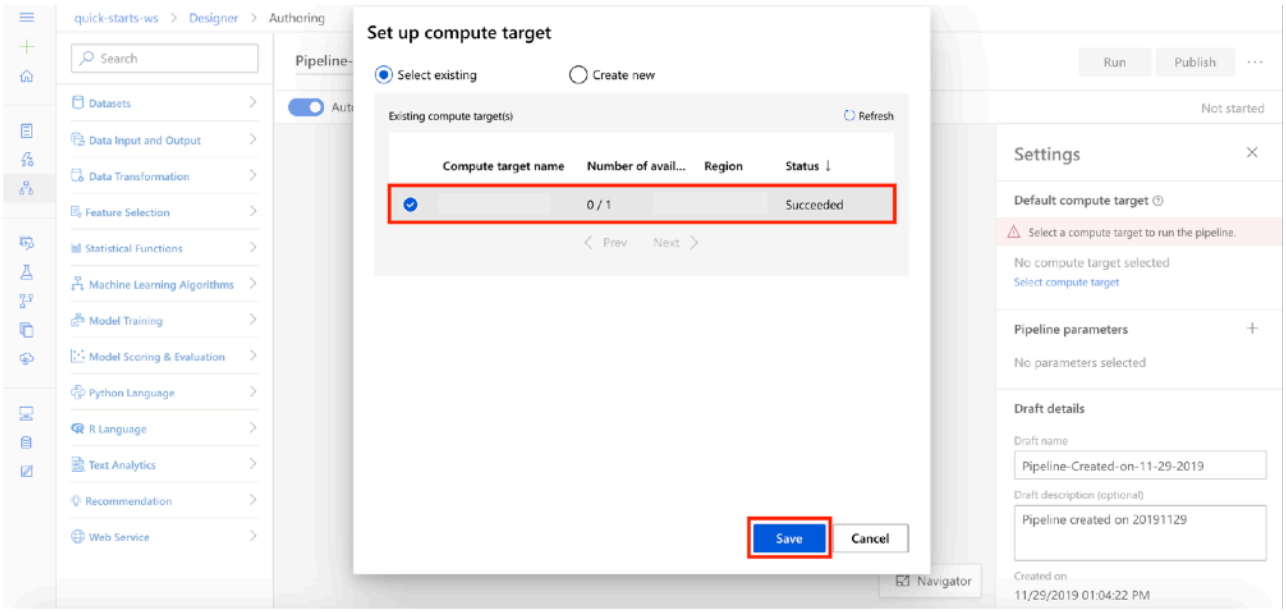
Task 2: Setup Compute Target

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



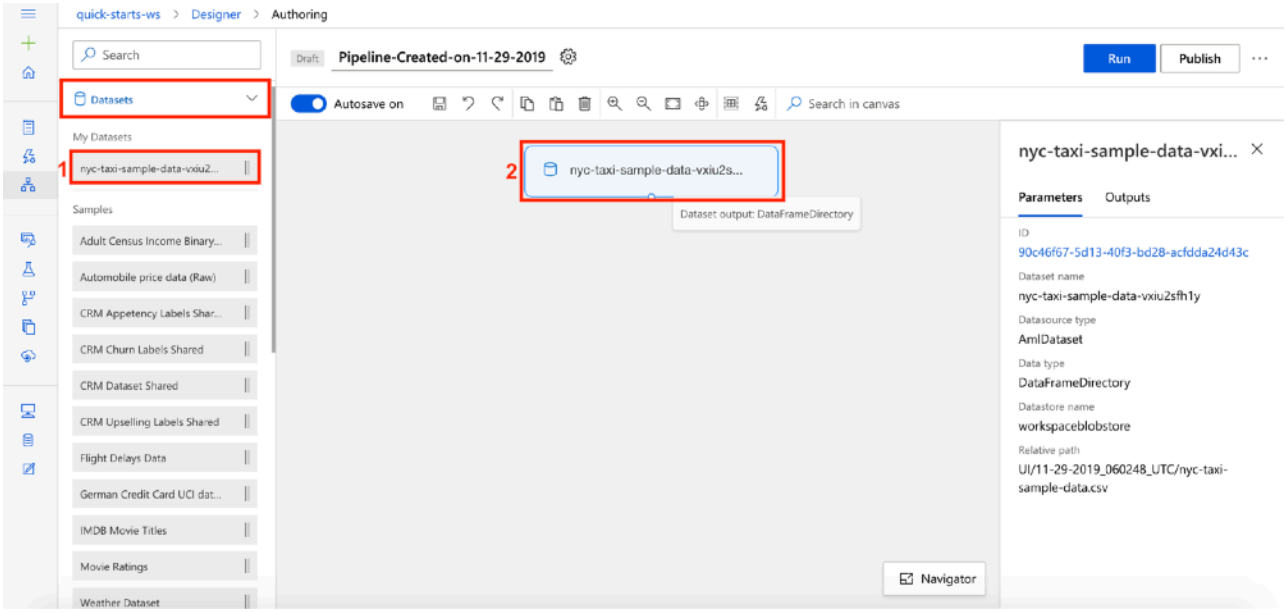
2. In the `Set up compute target` editor, select the available compute, and then select **Save**.

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.



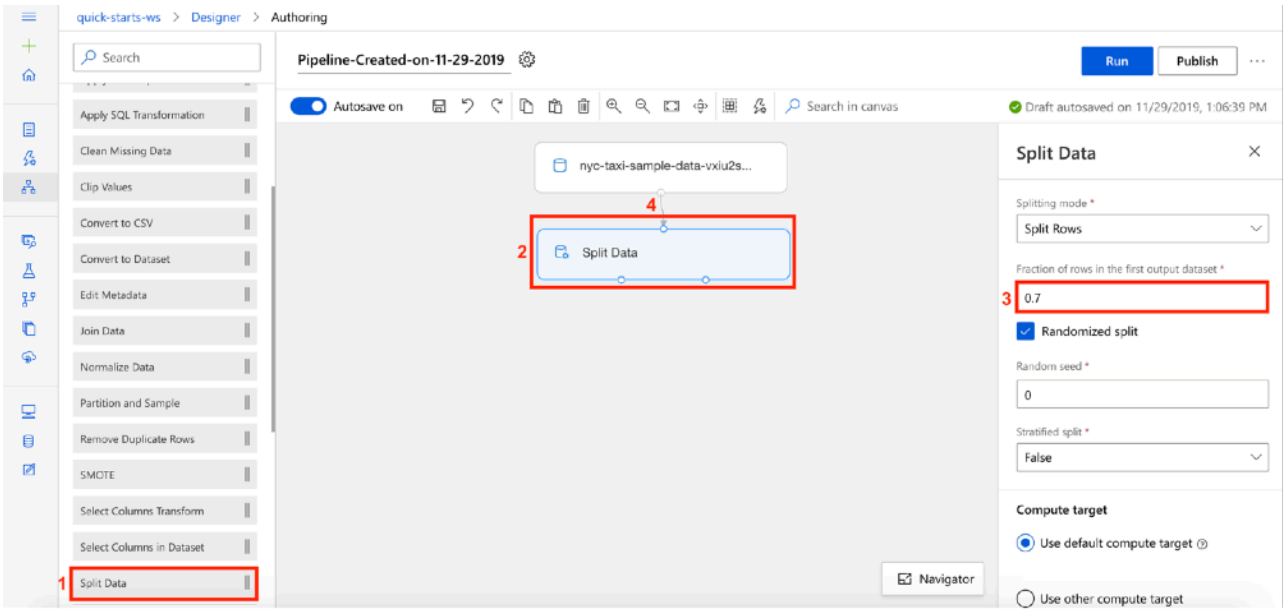
Task 3: Add Dataset

1. Select **Datasets** section in the left navigation. Next, select **My Datasets, nyc-taxi-sample-data** and drag and drop the selected dataset on to the canvas.



Task 4: Split Dataset

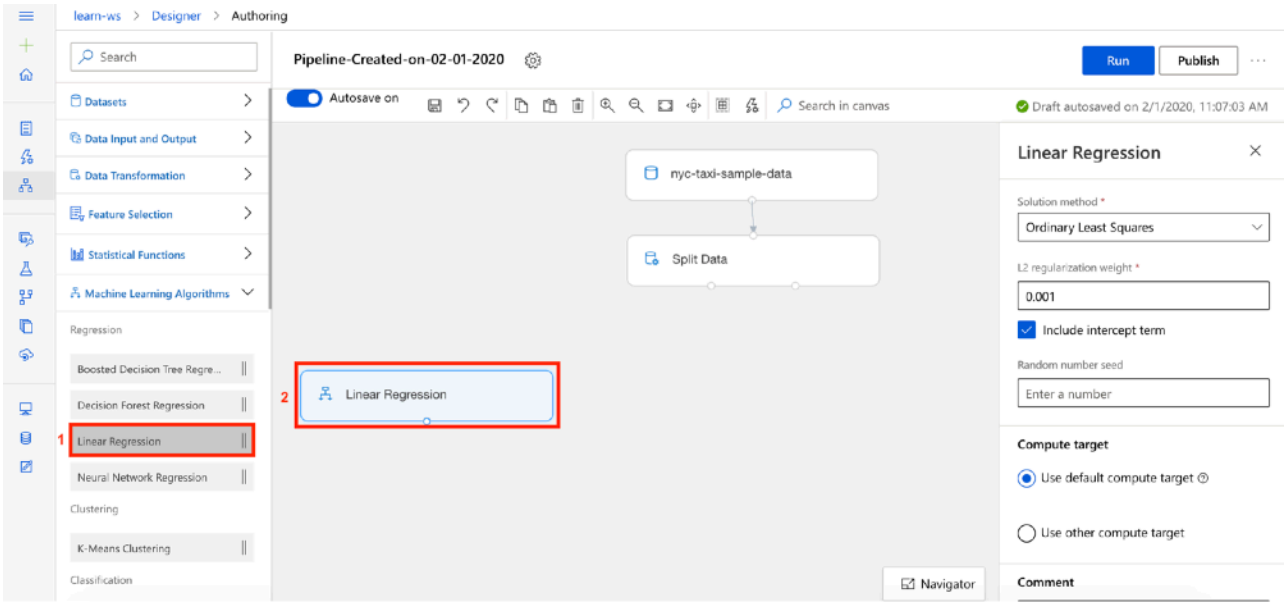
1. Select **Data Transformation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Split Data** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Fraction of rows in the first output dataset: **0.7**
    4. Connect the `Dataset` to the `Split Data` module



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.

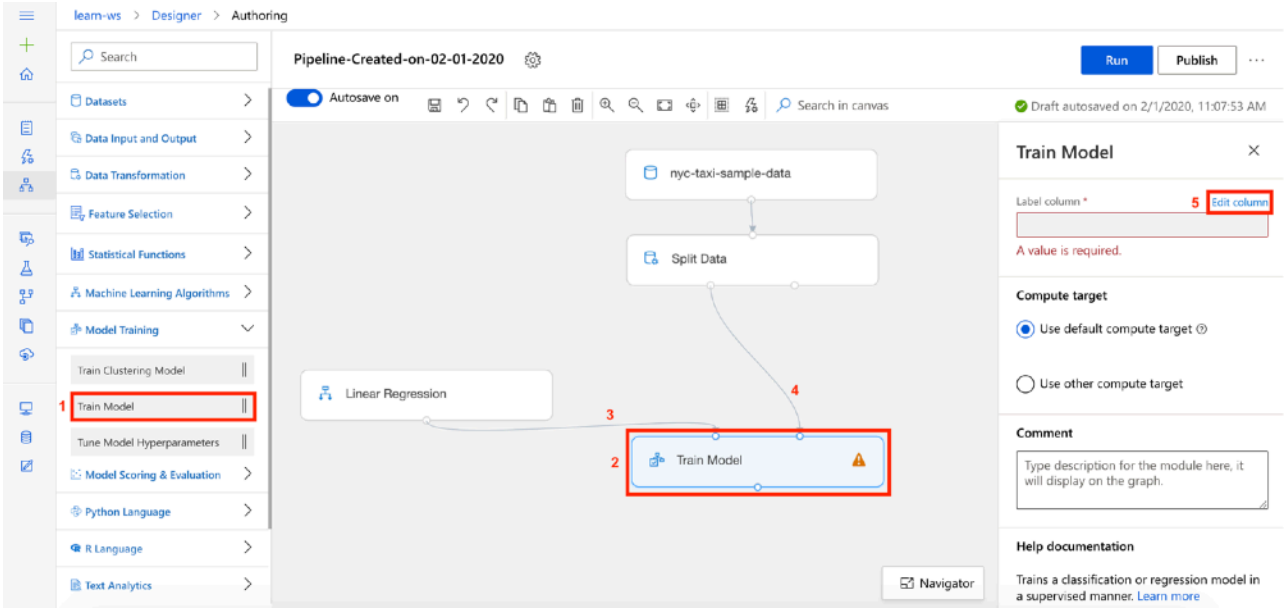
Task 5: Initialize Regression Model

1. Select **Machine Learning Algorithms** section in the left navigation. Follow the steps outlined below:
    1. Select the **Linear Regression** prebuilt module
    2. Drag and drop the selected module on to the canvas

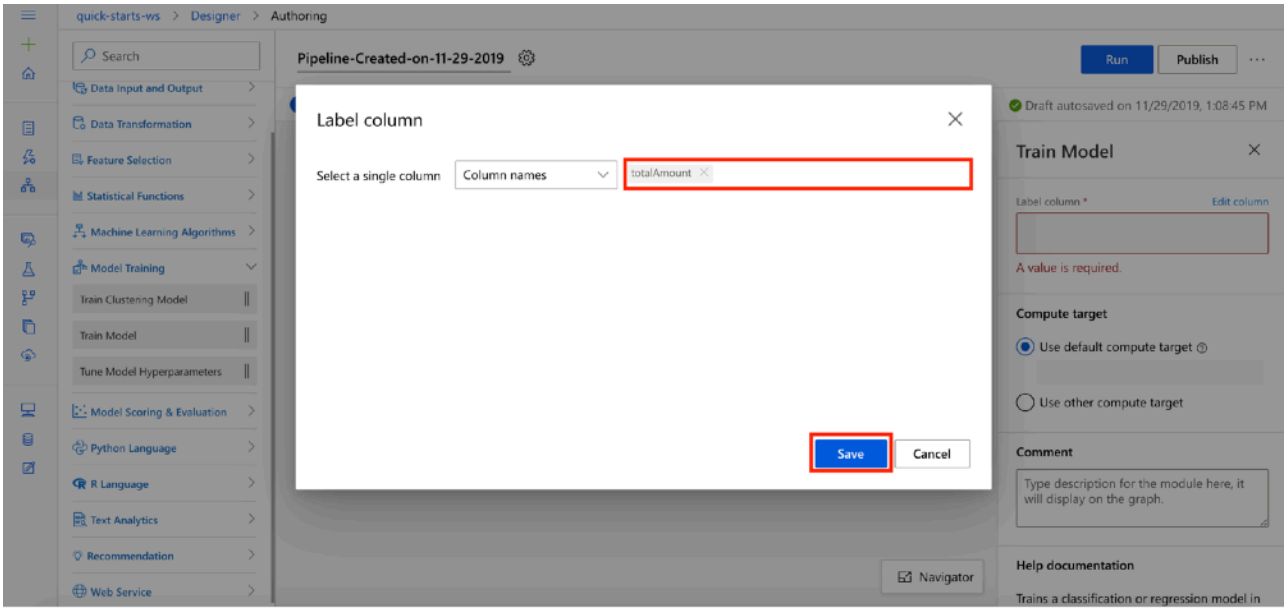


Task 6: Setup Train Model Module

1. Select **Model Training** section in the left navigation. Follow the steps outlined below:
    1. Select the **Train Model** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the `Linear 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
    5. Select the **Edit column** link to open the `Label column` editor

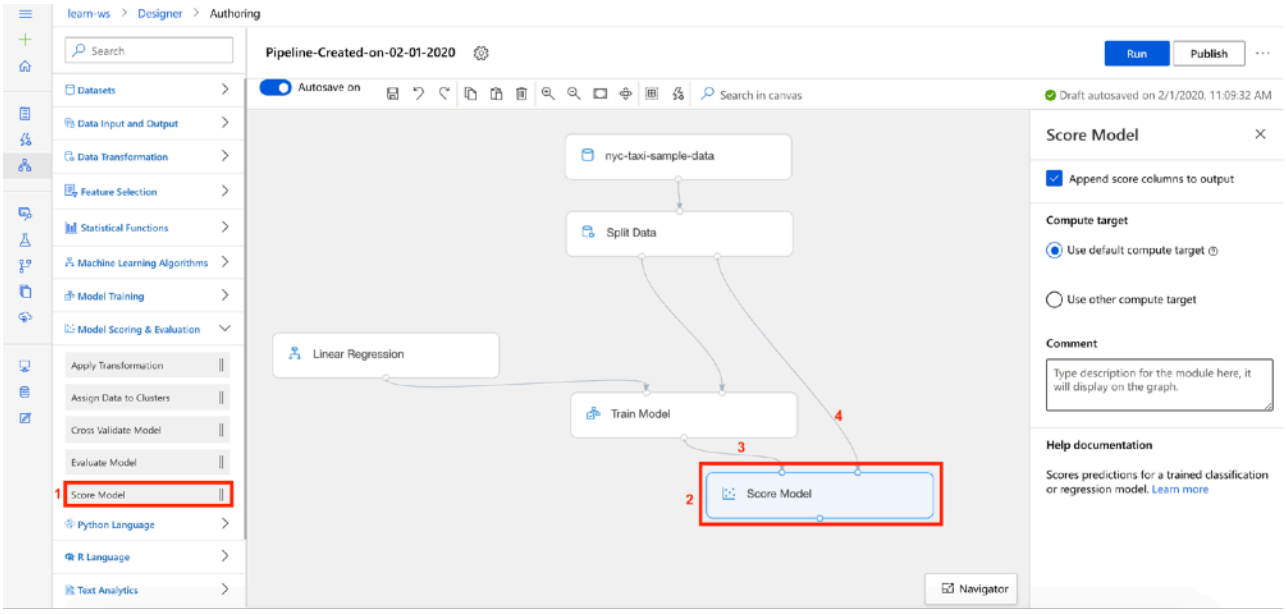


2. The `Label column` editor allows you to specify your `Label or Target column`. Type in the label column name **totalAmount** and then select **Save**.



Task 7: Setup Score Model Module

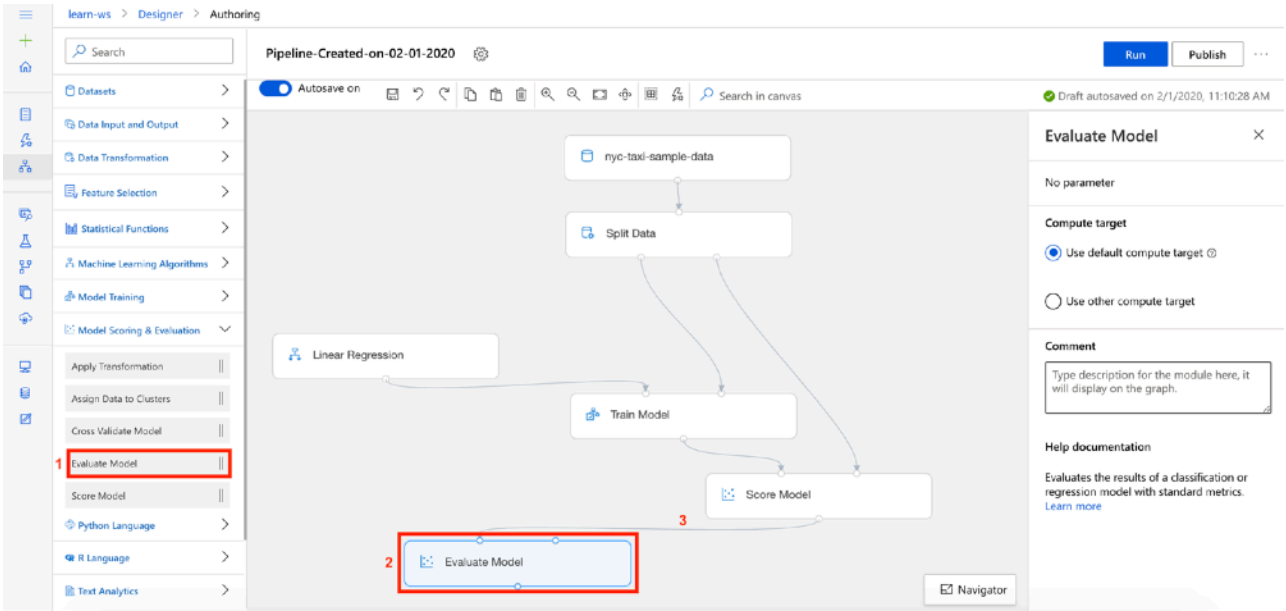
1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Score Model** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the `Train Model` module to the first input of the `Score Model` module
    4. Connect the second output of the `Split Data` module to the second input of the `Score Model` module



Note that *`Split Data`* module will feed data for both model training and model scoring. The first output (0.7 fraction) will connect with the *`Train Model`* module and the second output (0.3 fraction) will connect with the *`Score Model`* module.

Task 8: Setup Evaluate Model Module

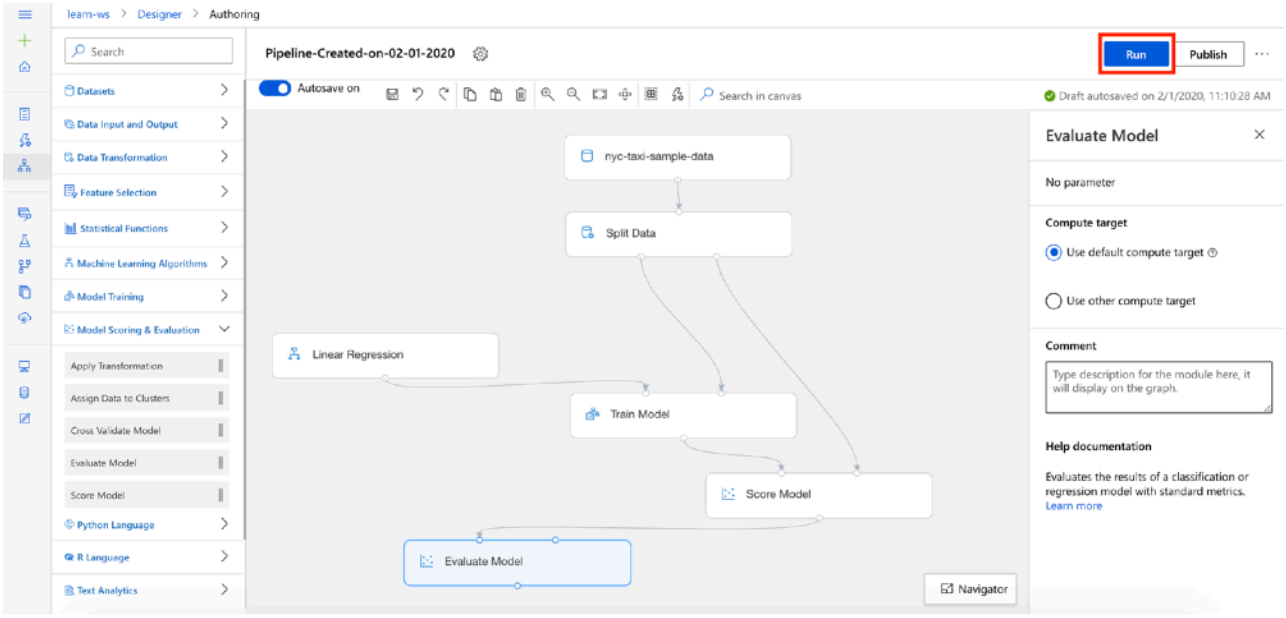
1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Evaluate Model** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the `Score Model` module to the first input of the `Evaluate Model` module



Exercise 3: Submit Training Pipeline

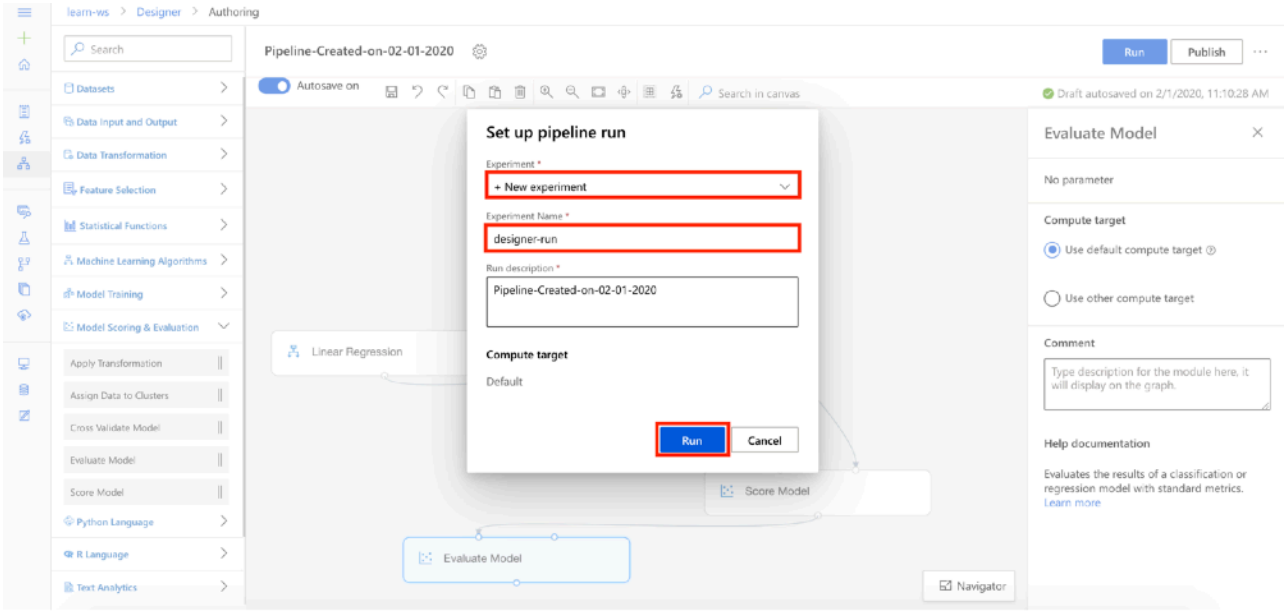
Task 1: Create Experiment and Submit Pipeline

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:` **designer-run**, and then select **Submit**.

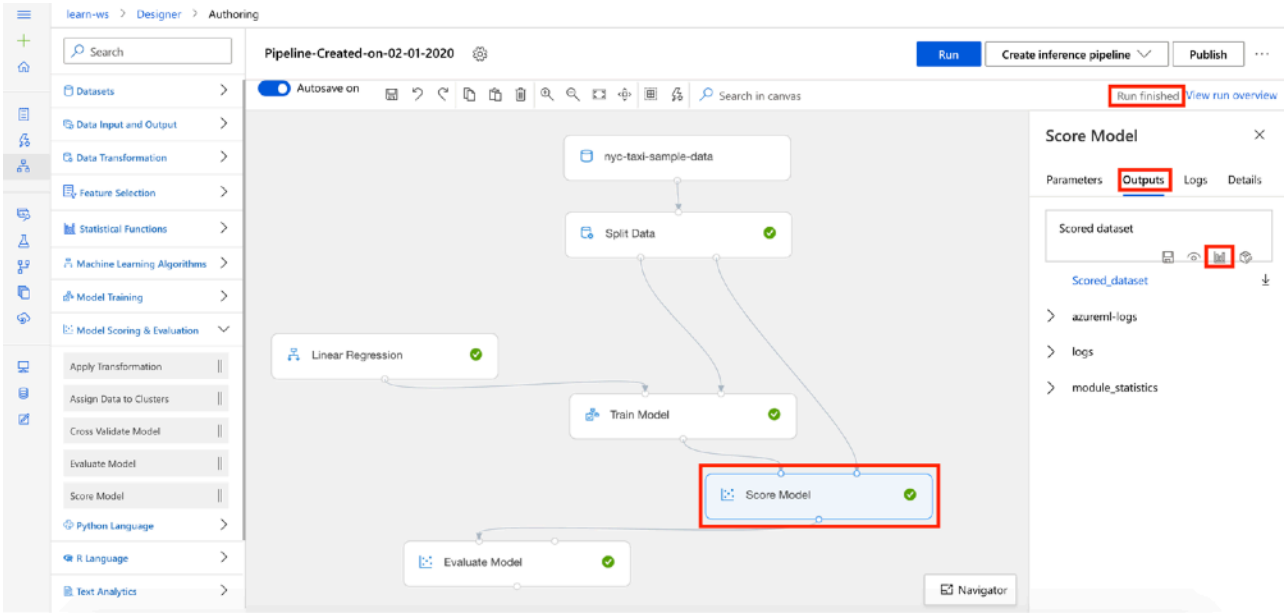


3. Wait for pipeline run to complete. It will take around **8 minutes** to complete the run.
4. While you wait for the model training to complete, you can learn more about the training algorithm used in this lab by selecting Linear Regression module.

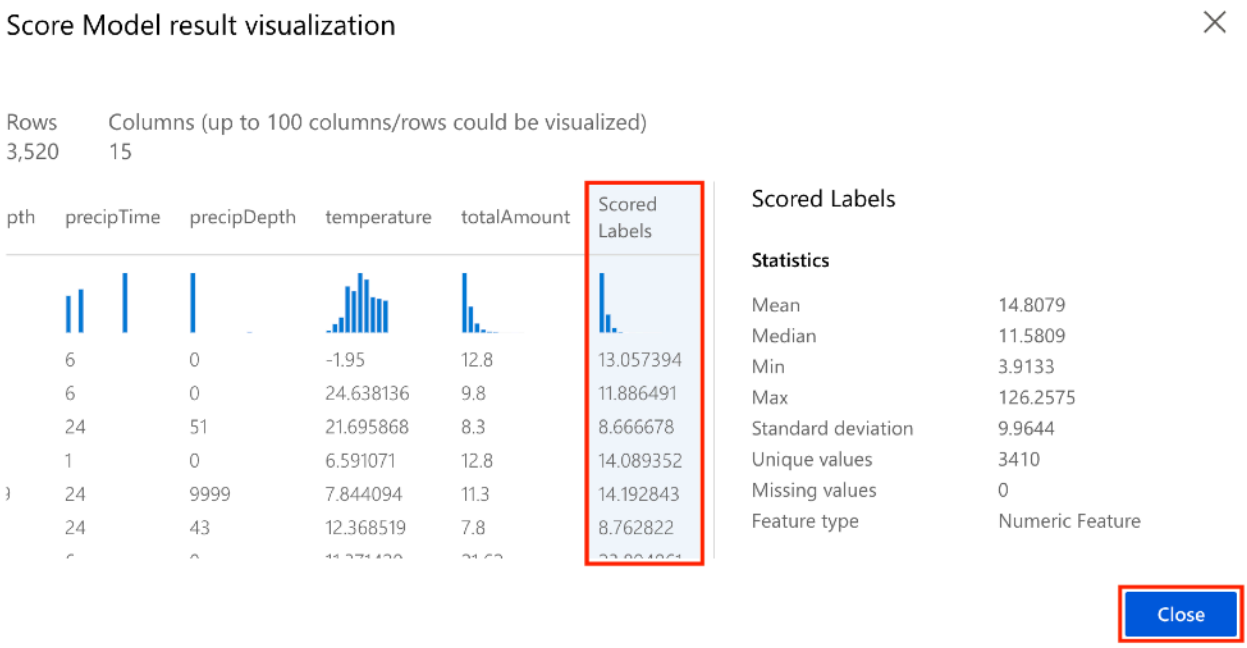
Exercise 4: Visualize Training Results

Task 1: Visualize the Model Predictions

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

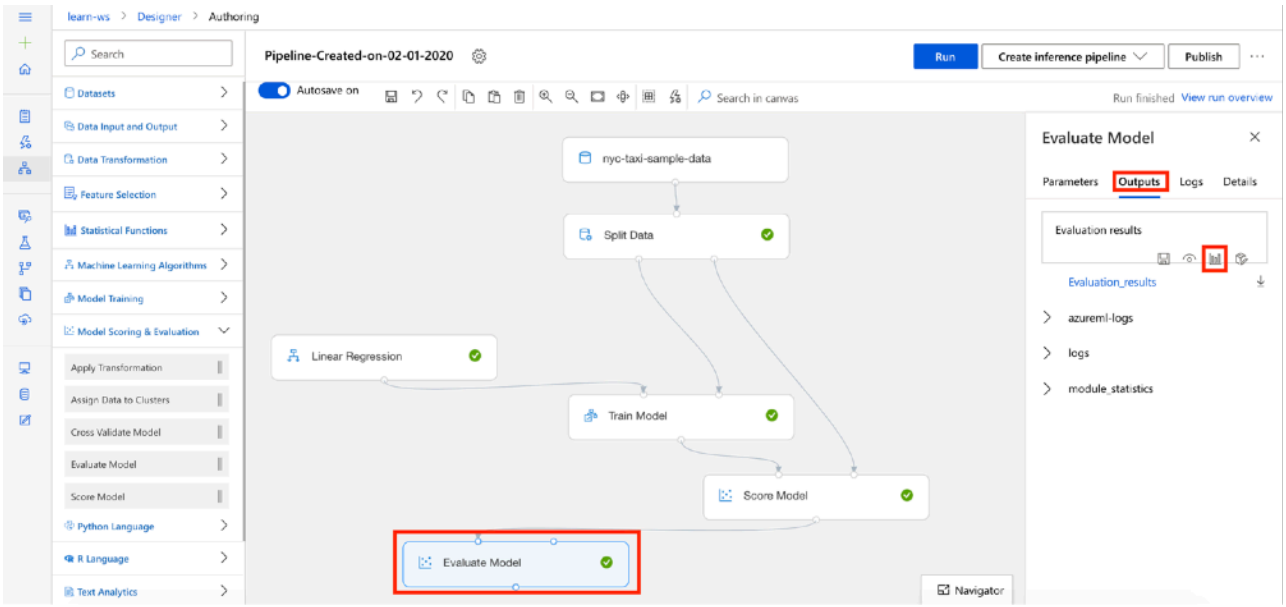


2. Observe the predicted values under the column **Scored Labels**. You can compare the predicted values (`Scored Labels`) with actual values (`totalAmount`).

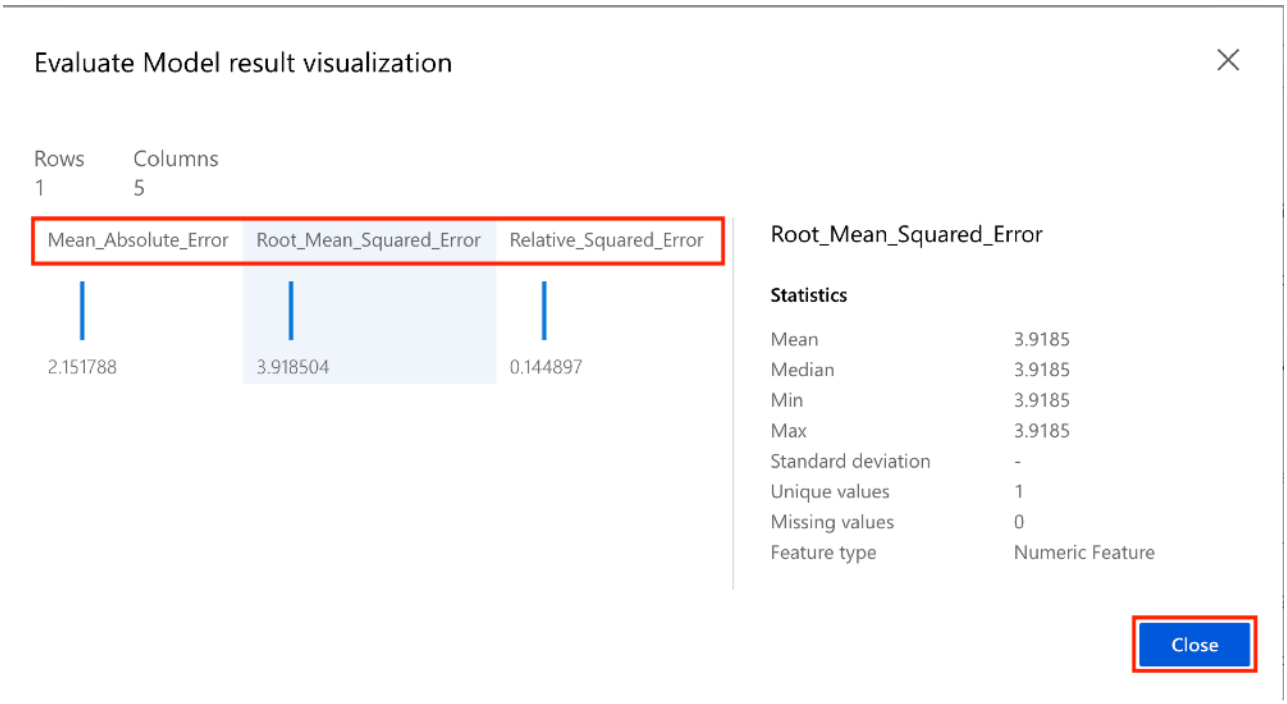


Task 2: Visualize the Evaluation Results

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



2. Evaluate the model performance by reviewing the various evaluation metrics, such as **Mean Absolute Error**, **Root Mean Squared Error**, etc.



Next Steps

Congratulations! You have trained and evaluated your first machine learning model. 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.

Lab 2 - Import, Transform, and Export Data

Overview

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

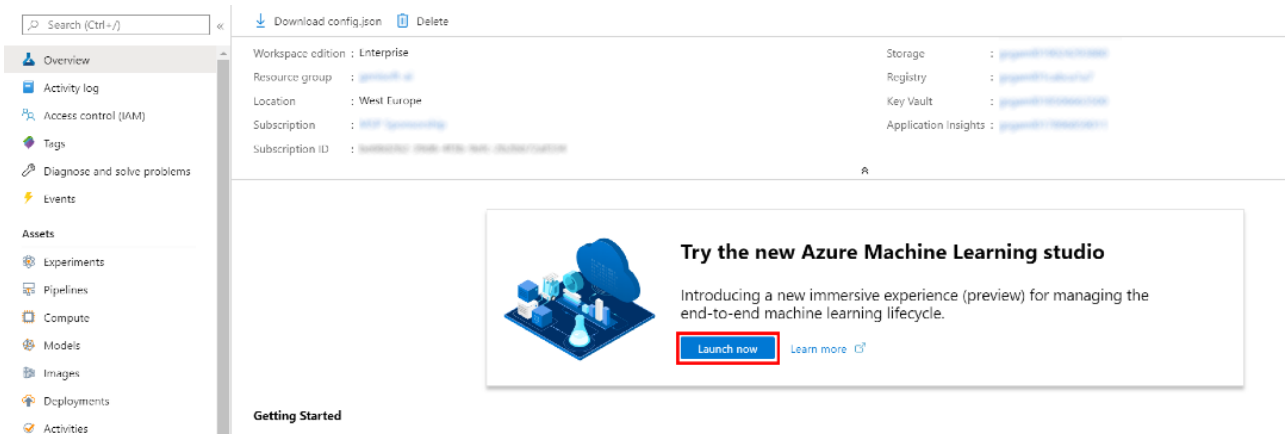
- Azure Machine Learning datasets
  Register datasets in Azure Machine Learning to enable advanced features that help you manage your data.
- Import Data module
  Use the Import Data module to directly access data from online datasources.

The first approach will be covered later in the next lab, which focuses on registering and versioning a dataset in Azure Machine Learning studio.
While the use of datasets is recommended to import data, you can also use the Import Data module from the designer. Data comes into the designer from either a `Datastore` or from `Tabular Datasets`. Datastores will be covered later in this course, but just for a quick definition, you can use Datastores to access your storage without having to hard code connection information in your scripts. As for the second option, the Tabular datasets, the following datasources are supported in the designer: Delimited files, JSON files, Parquet files or SQL queries.
The following exercise focuses on the Import Data module to load data into a machine learning pipeline from several datasets that will be merged and restructured. We will be using some sample data from the UCI dataset repository to demonstrate how you can perform basic data import transformation steps with the modules available in Azure Machine Learning designer.

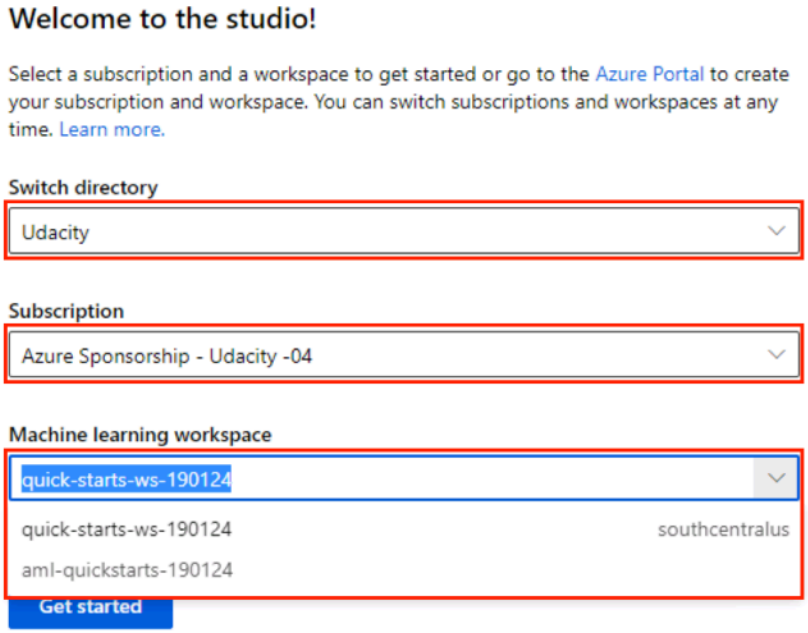
Exercise 1: Import, transform and export data using the Visual Pipeline Authoring Editor

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.

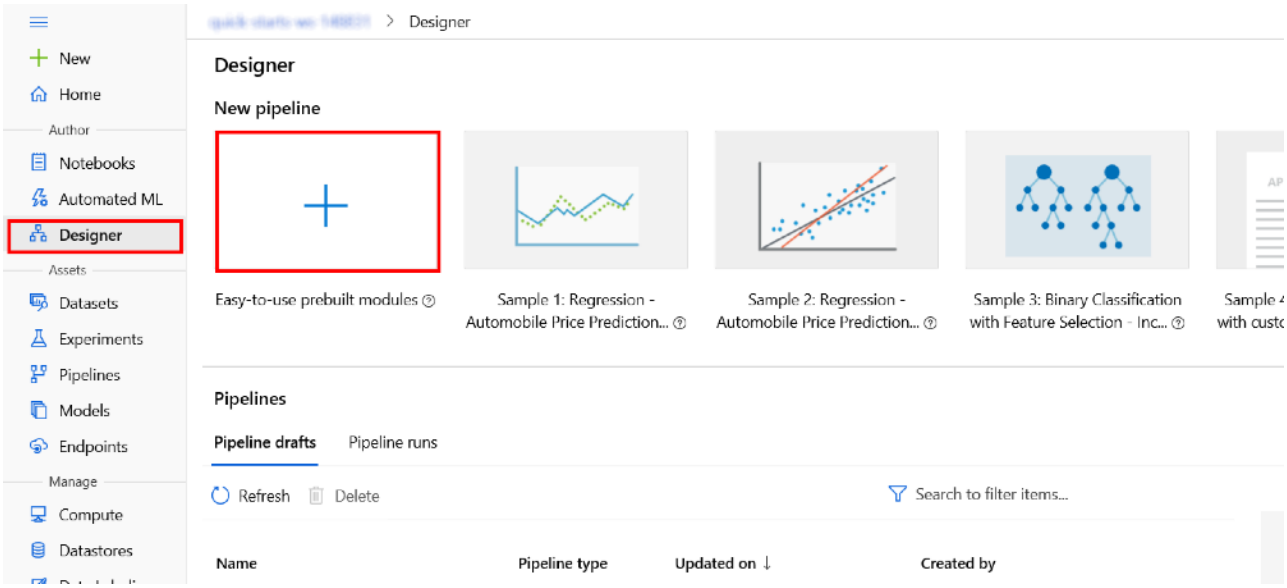


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



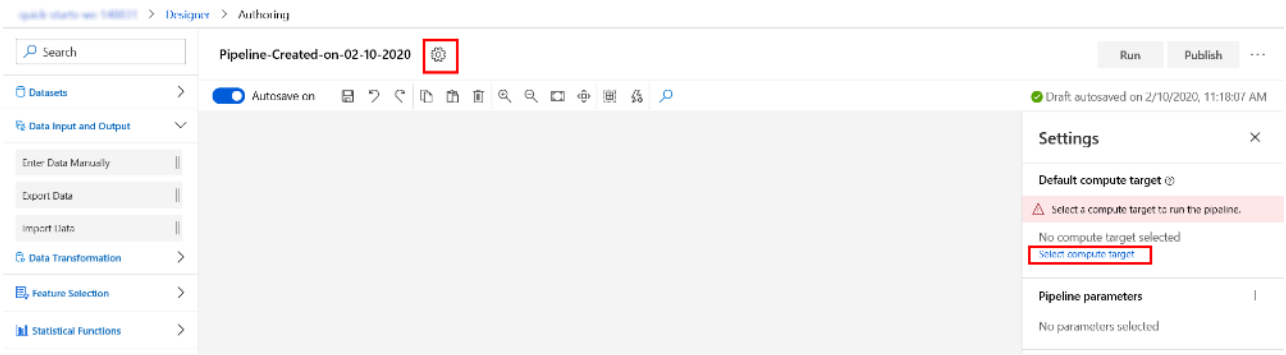
For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

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



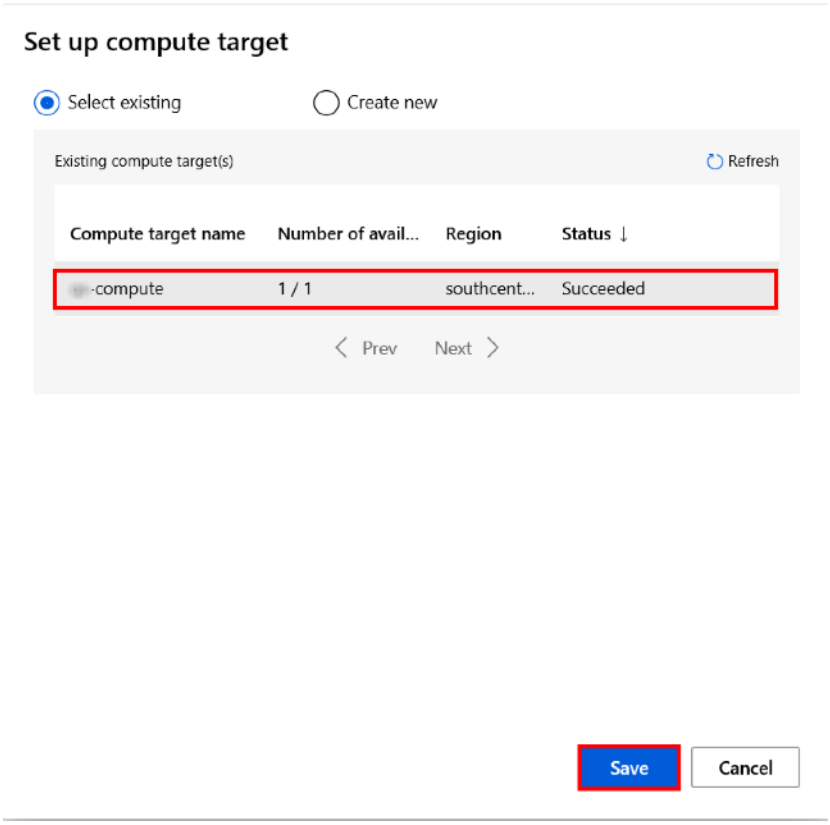
Task 2: Setup Compute Target

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



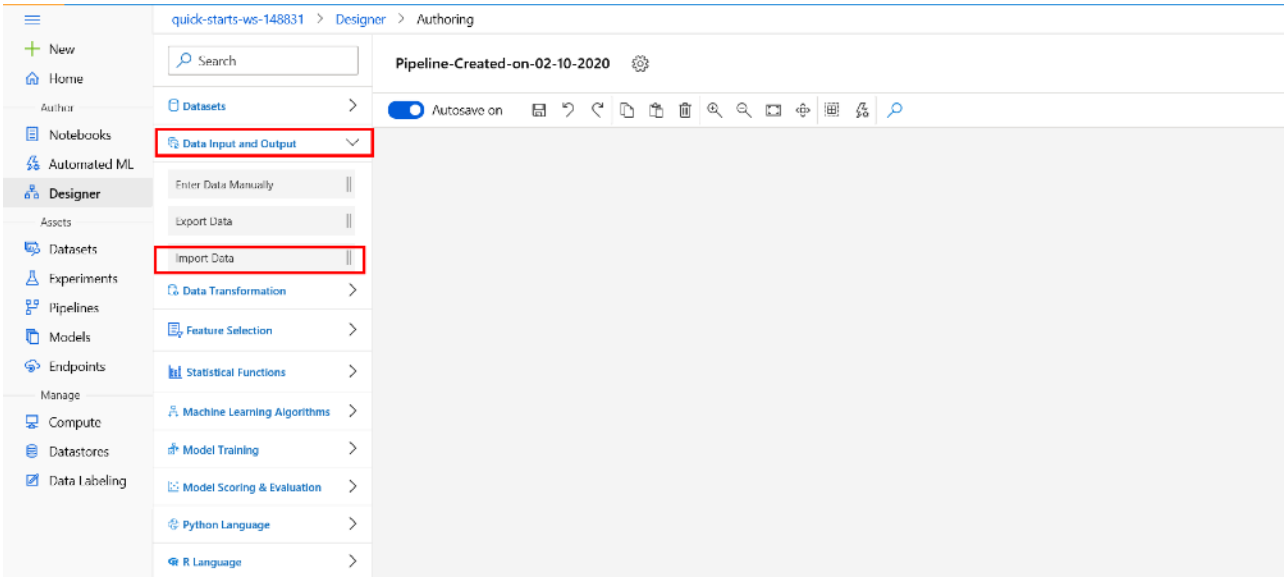
2. In the `Set up compute target` editor, select the existing compute target, and then select **Save**.

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.

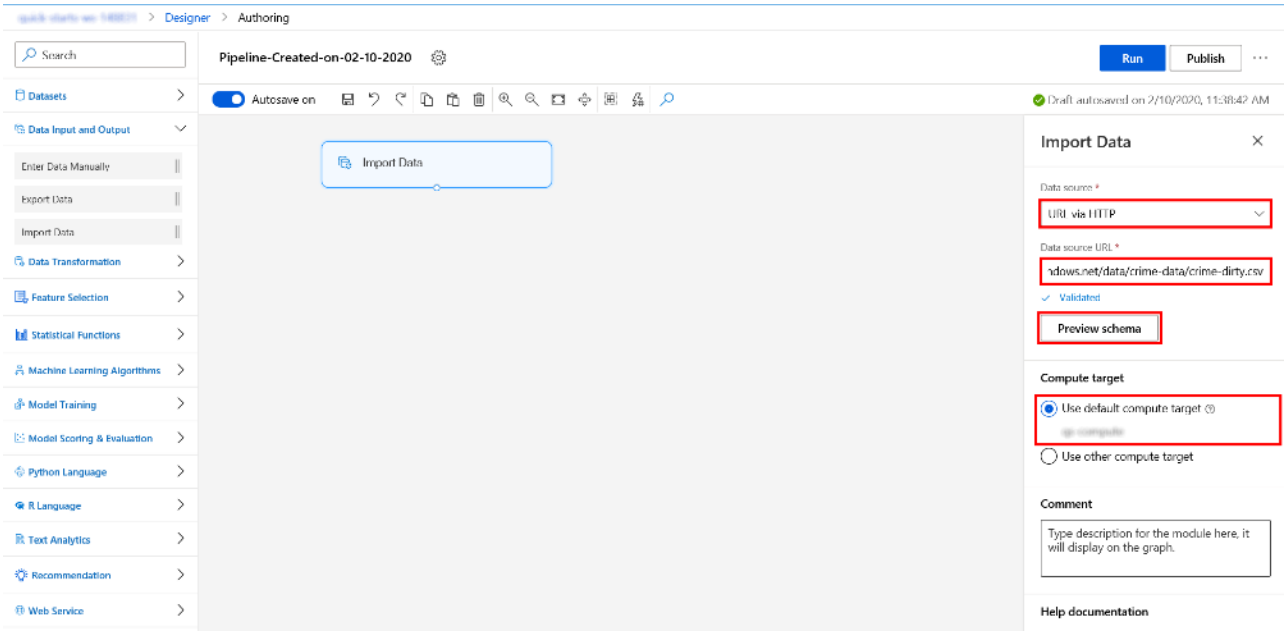


Task 3: Import data from Web URL

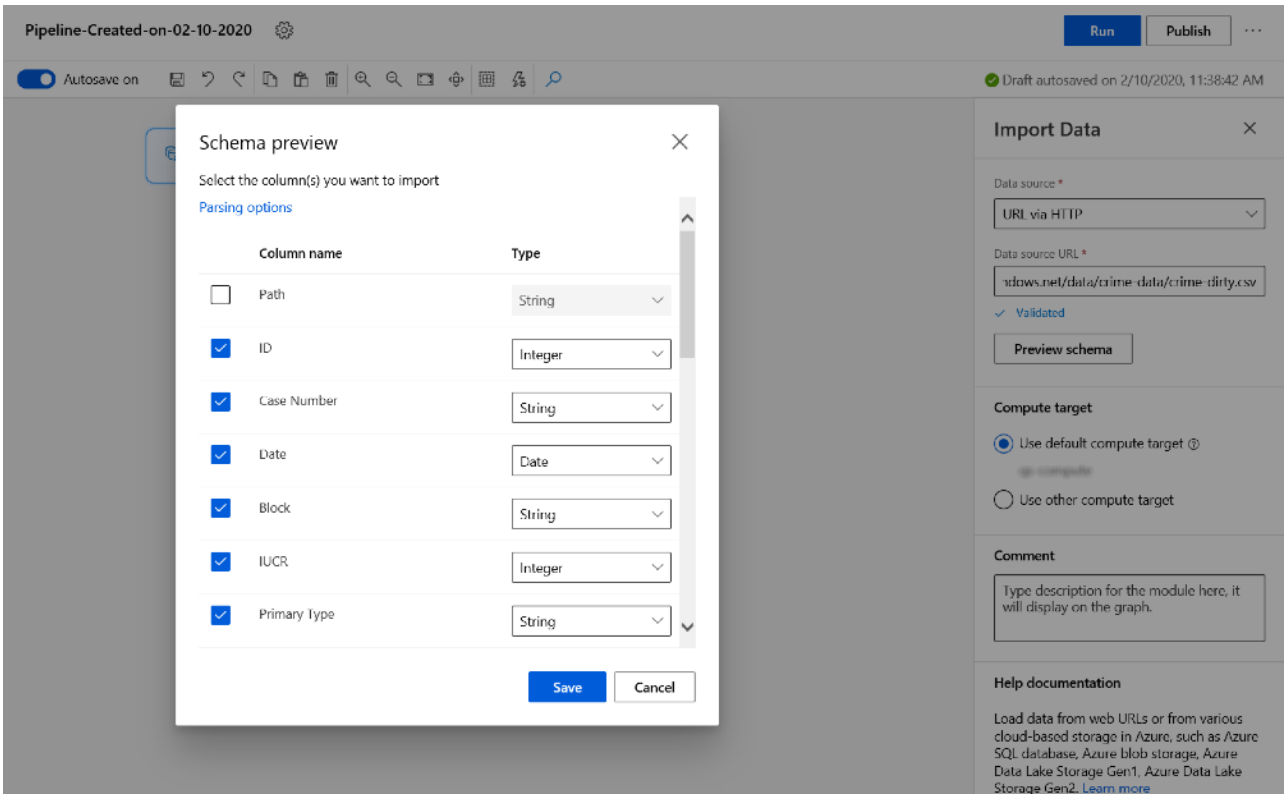
1. Select **Data Input and Output** section in the left navigation. Next, select **Import Data** and drag and drop the selected module on to the canvas.



2. In the `Import data` panel on the right, select the **URL via HTTP** option in the `Data Source` drop-down and provide the following `Data source URL` for the first CSV file you will import in your pipeline: `https://introtomlsampledata.blob.core.windows.net/data/crime-data/crime-dirty.csv`

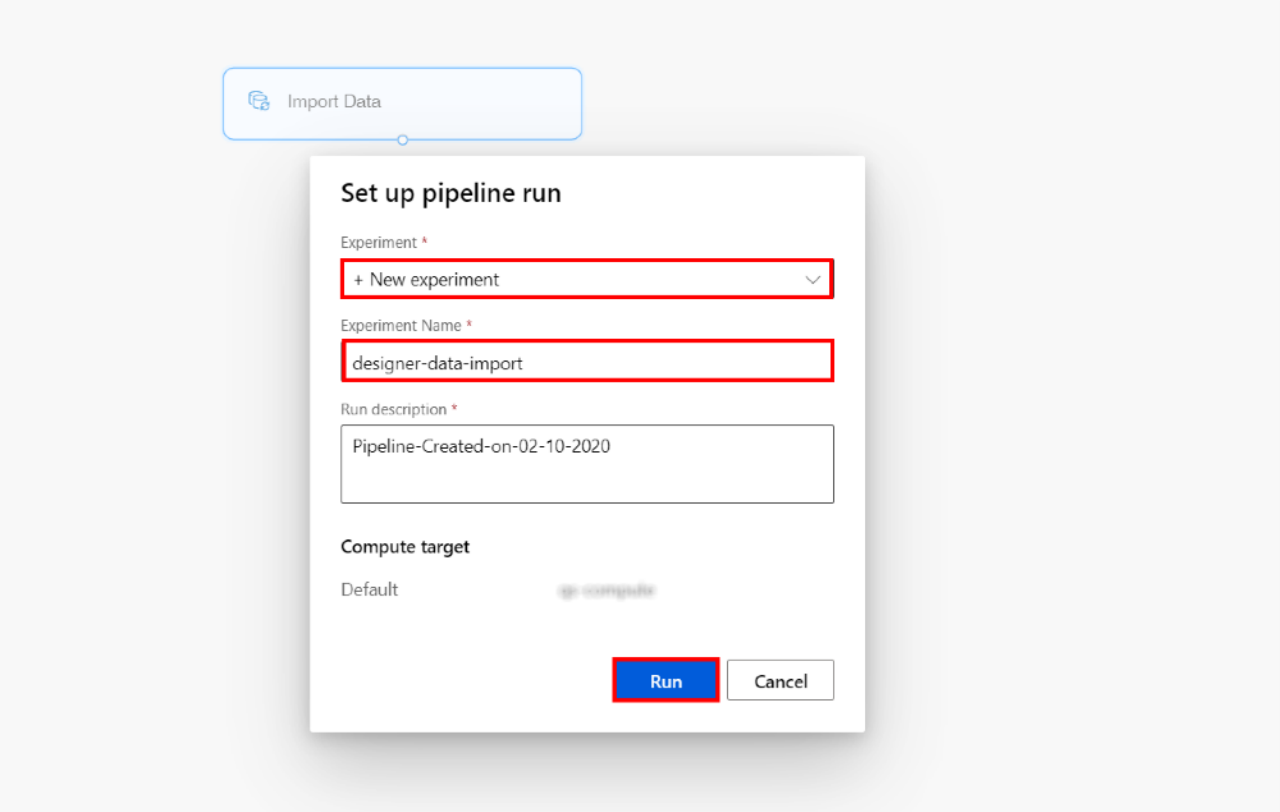


3. Select the **Preview schema** to filter the columns you want to include. You can also define advanced settings like Delimiter in `Parsing options`. Select **Save** to close the dialog.



Task 4: Create Experiment and Submit Pipeline

1. Back to the pipeline canvas, select **Submit** on the top right corner to open the `Setup pipeline run` editor.
2. In the `Setup pipeline run editor`, select **Experiment, Create new** and provide `New experiment name:` **designer-data-import**, and then select **Submit**.

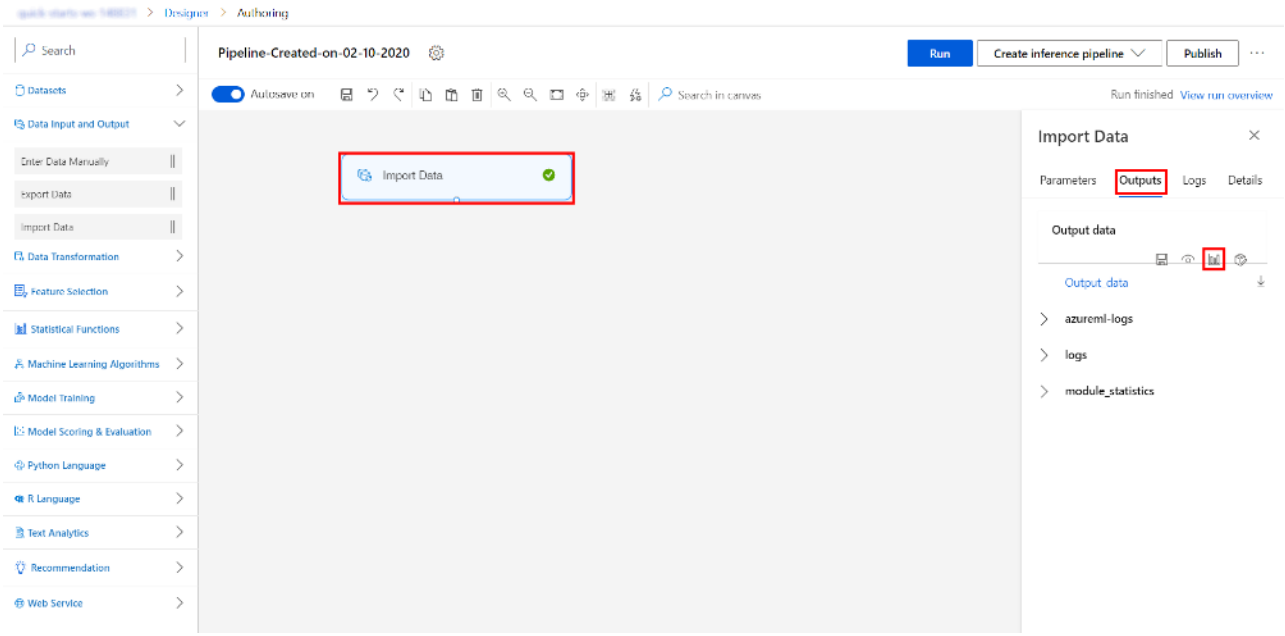


Please note that the button name in the UI is changed from **Run** to **Submit**.

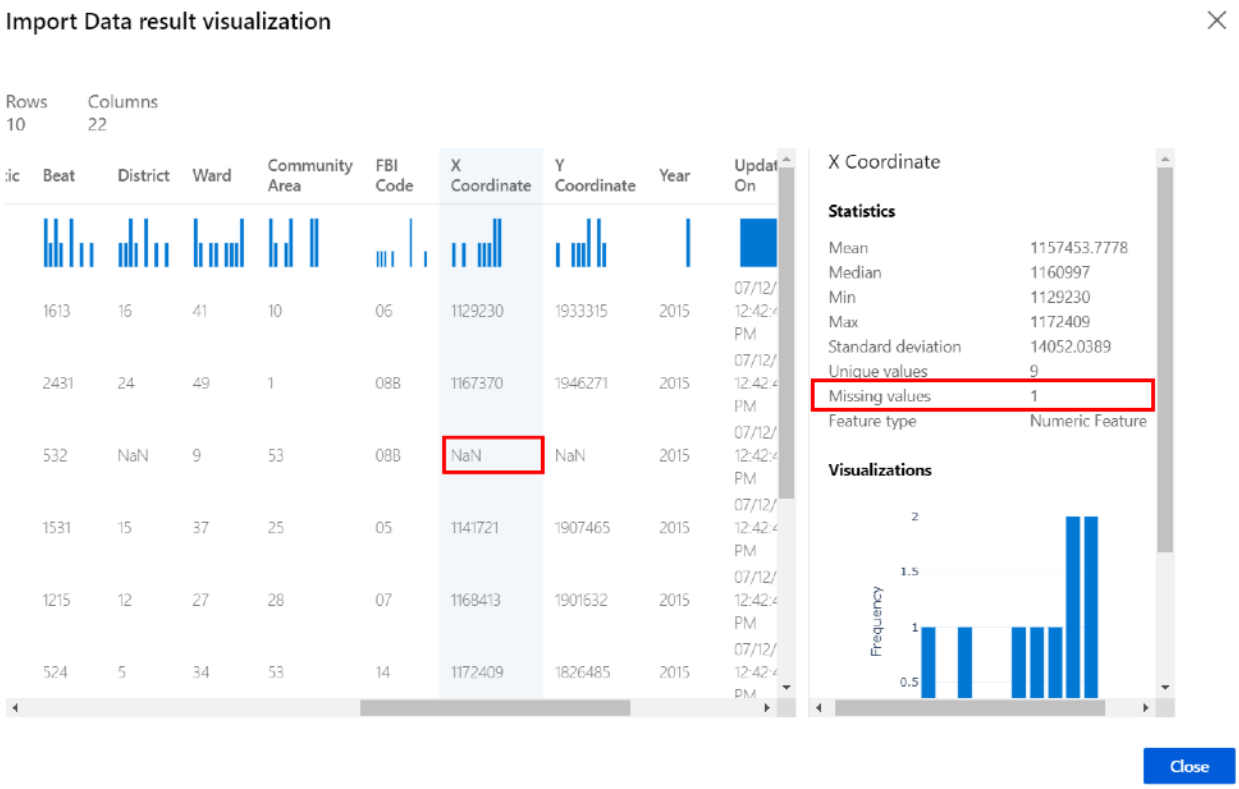
3. Wait for pipeline run to complete. It will take around **10 minutes** to complete the run.

Task 5: Visualize Import Data results

1. Select the `Import Data` module on the canvas and then select **Outputs** on the right pane. Click on the **Visualize** icon to open the `Import Data result visualization` dialog.



2. In the `Import Data result visualization` dialog take some moments to explore all the metadata that is now available to you, such as: number of rows, columns, preview of data and for each column you select you can observe: **Mean**, **Median**, **Min**, **Max** and also number of **Unique Values** and **Missing Values**. Data profiles help you glimpse into the column types and summary statistics of a dataset. Scroll right and select the **X Coordinate** column. Notice the `Nan` value on the third row in the preview table and check the `Missing values` number in the **Statistics** section.

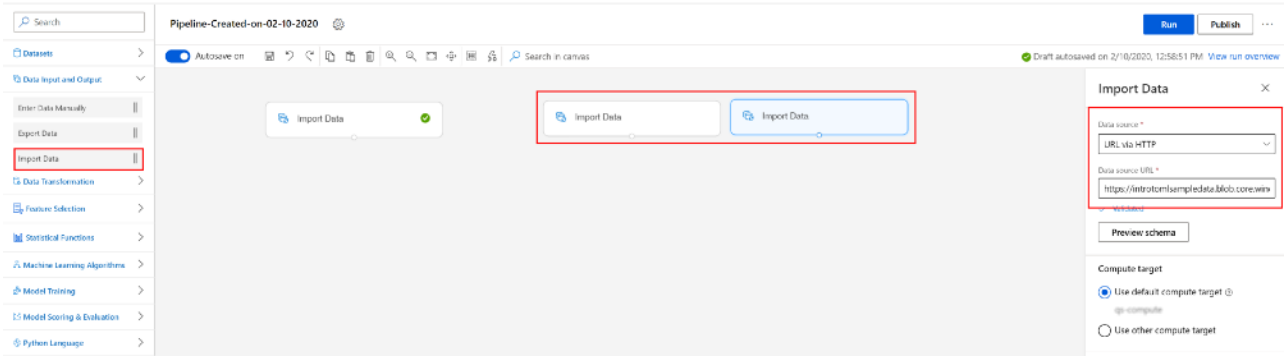


3. Select **Close** to return to the pipeline designer canvas where you can continue the data import phase.

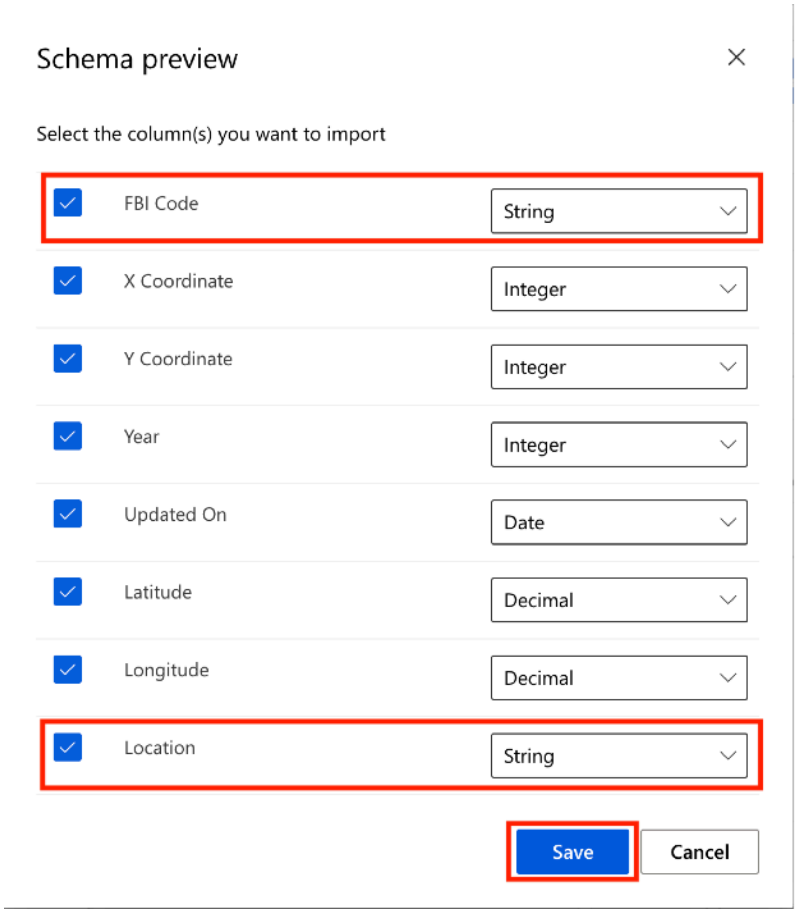
Exercise 2: Restructure the data split across multiple files

Task 1: Append rows from two additional data sources

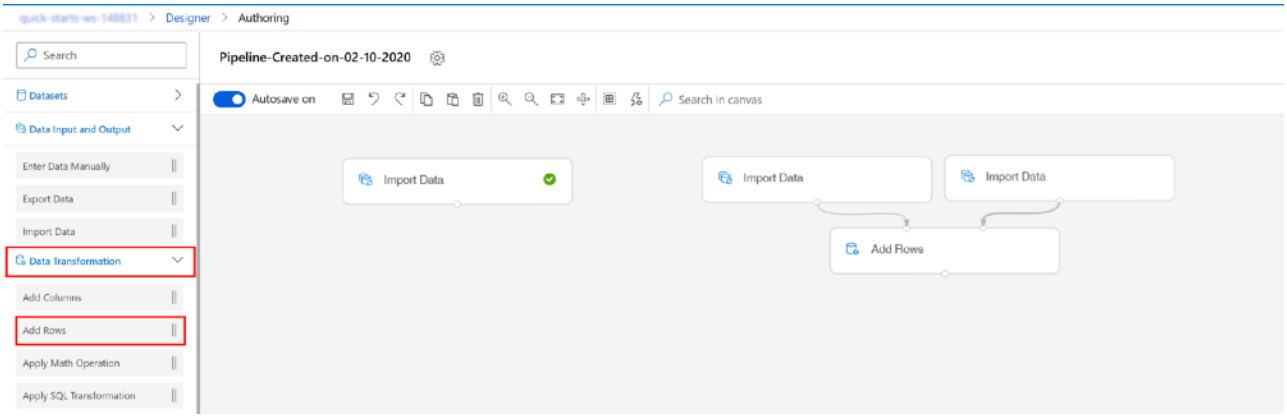
1. Select **Data Input and Output** section in the left navigation. Next, drag and drop two **Import Data** modules on to the canvas as demonstrated in the first exercise and fill in the Web URLs as follows:
   - for the first one, **Data source URL** : `https://introtomlsampledata.blob.core.windows.net/data/crime-data/crime-spring.csv`
   - for the second one, **Data source URL** : `https://introtomlsampledata.blob.core.windows.net/data/crime-data/crime-winter.csv`



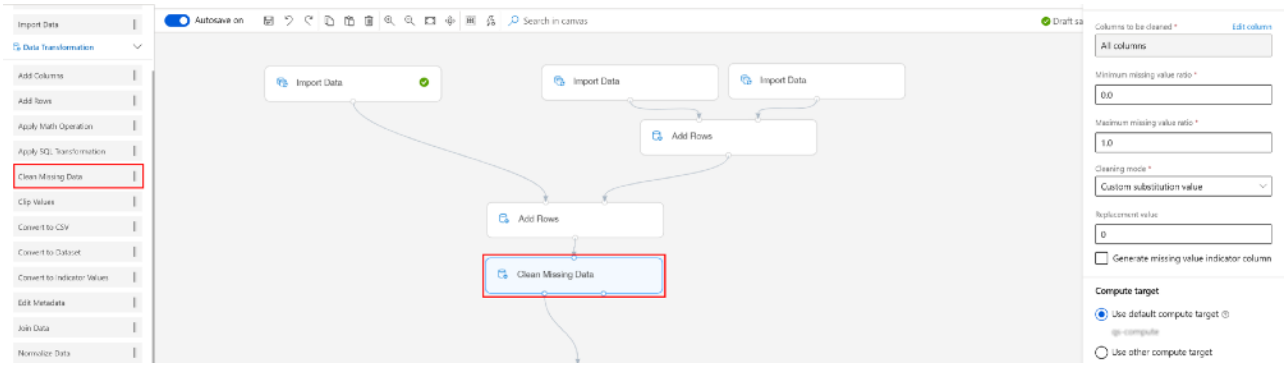
2. For each of the **three** `Import Data` modules, select **Preview schema** and ensure that the data type for `FBI Code` and `Location` is of type `String` and then select **Save**.



3. Select the **Data Transformation** section in the left navigation. Drag and drop the **Add rows** module and connect it to the above added Import data modules.

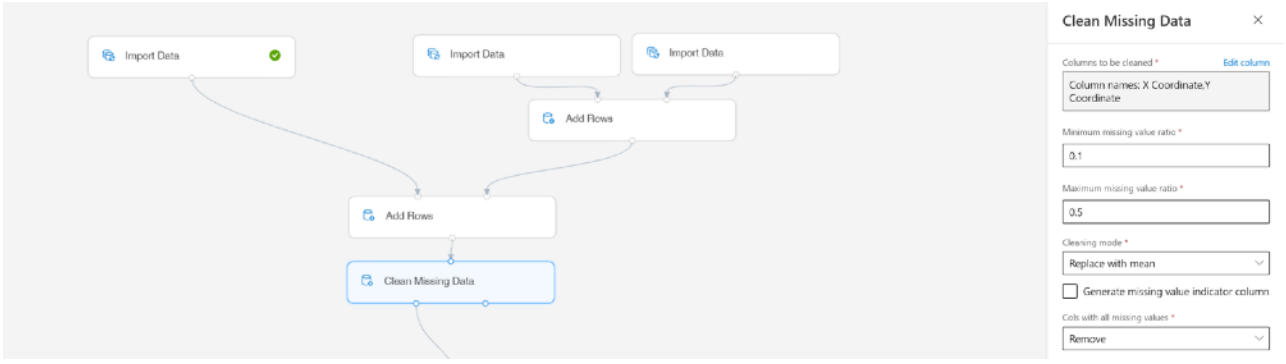


4. Repeat the same step and add a second **Add rows** module that connects the output from the first **Import data** module to the output of the first **Add rows** module.

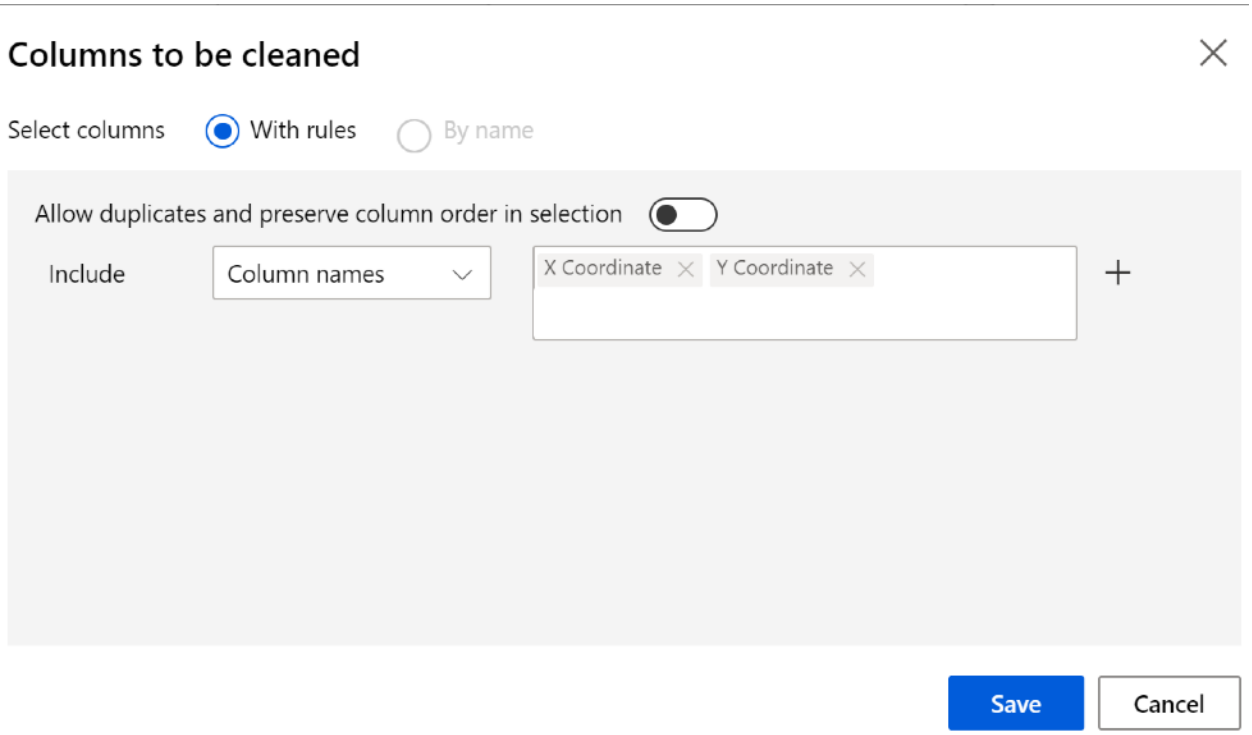


Task 2: Clean missing values

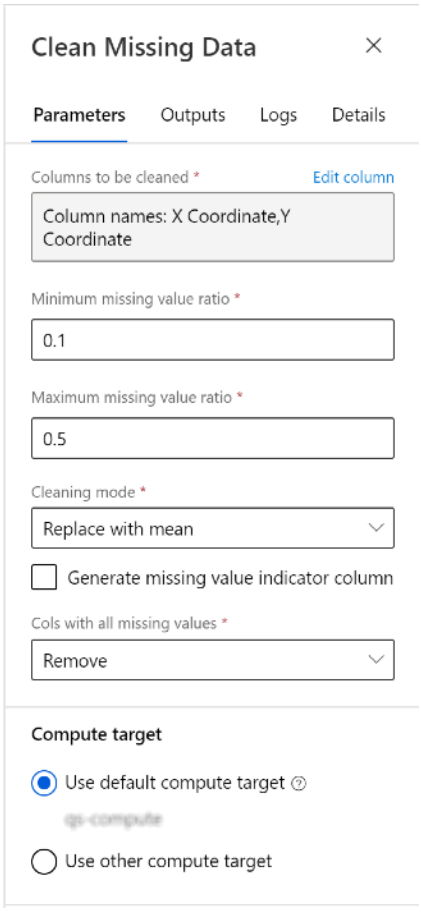
1. Drag the **Clean Missing Data** module from the **Data Transformation** section in the left navigation.



2. Select **Edit column** in the right pane to configure the list of columns to be cleaned. Select `Column names` from the available include options and type the name of the columns you intend to clean at this step: `X Coordinate` and `Y Coordinate`. Select **Save** to close the dialog.

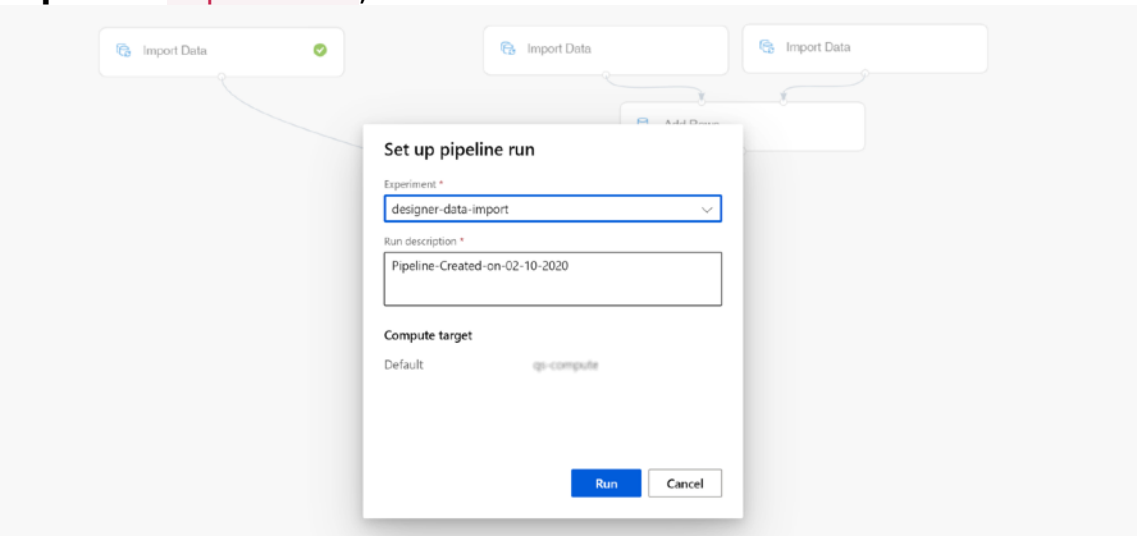


3. Set the **Minimum missing value ratio** to `0.1` and the **Maximum missing value ratio** to `0.5`. Select `Replace with mean` in the **Cleaning mode** field.



Task 3: Submit Pipeline

1. Select **Submit** to open the `Setup pipeline run` editor.
2. In the `Setup pipeline run` editor, select **Select existing, designer-data-import** for `Experiment`, and then select **Submit**.



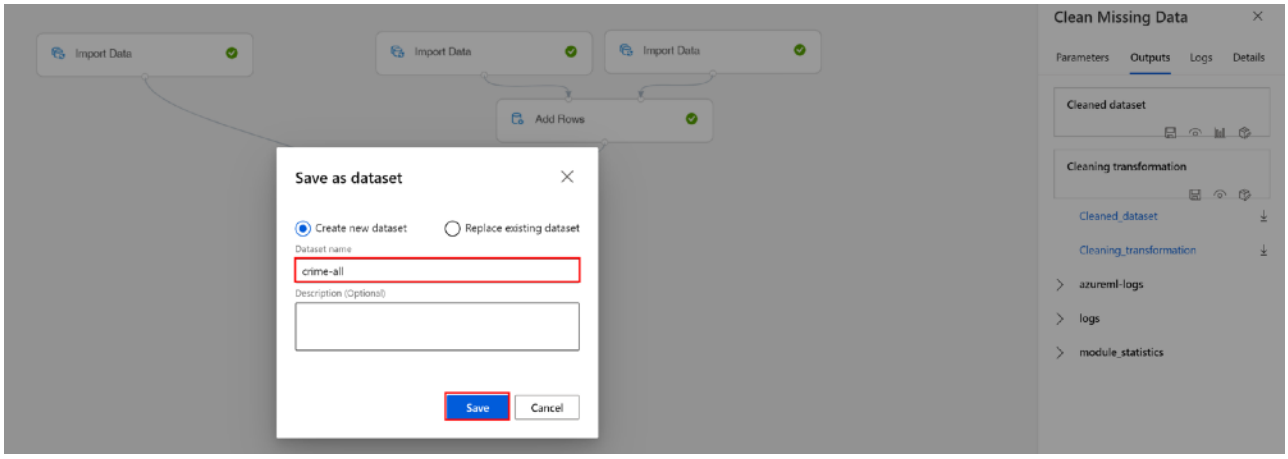
Please note that the button name in the UI is changed from **Run** to **Submit**.

3. Wait for pipeline run to complete. It will take around **8 minutes** to complete the run.

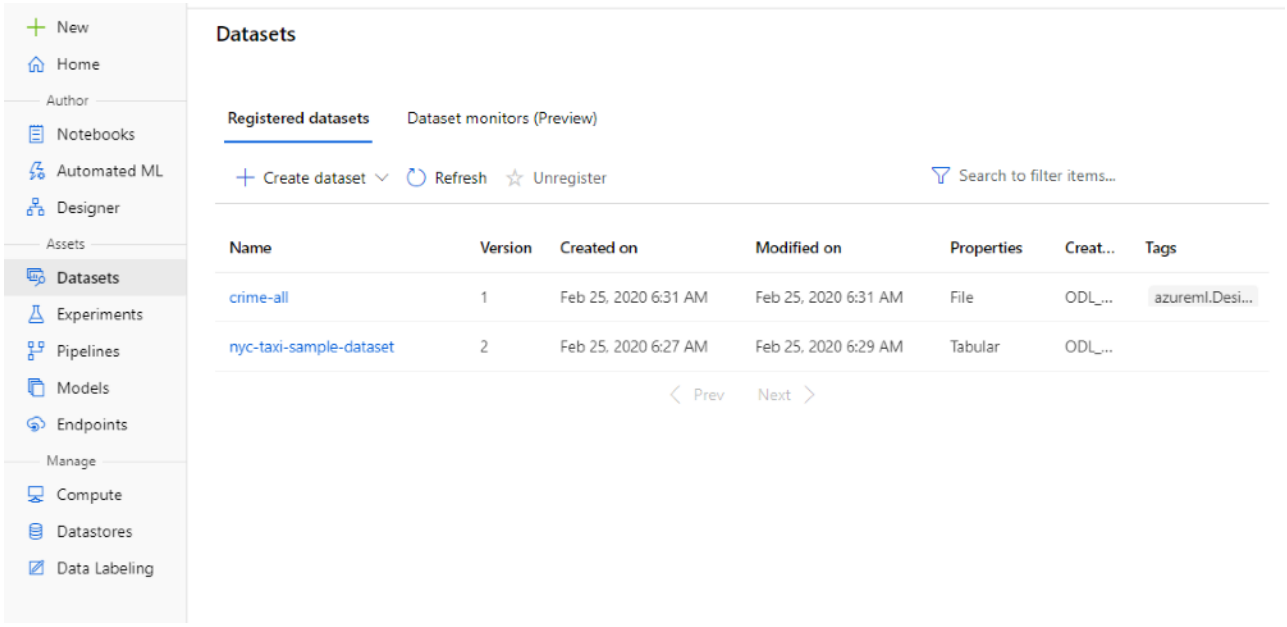
Task 4: Save the clean dataset

1. Select the `Clean missing data` module you created on the canvas and then select **Outputs + logs** on the right pane. Click on the **Save** icon under the **Cleaned dataset** section to open the `Save as dataset` dialog.

2. Check the option to create a new dataset and enter `crime-all` in the dataset name field. Select **Save** to close the dialog.



3. From the left navigation, select **Datasets**. This will open the `Registered datasets` page. See your registered dataset among the other datasets you used during this lesson.



Next Steps

Congratulations! You completed a few basic steps involved in the data explore and transform process, using the prebuilt modules you can find in the visual editor provided by Azure Machine Learning Studio. 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.

Lab 3 - Create and version a dataset

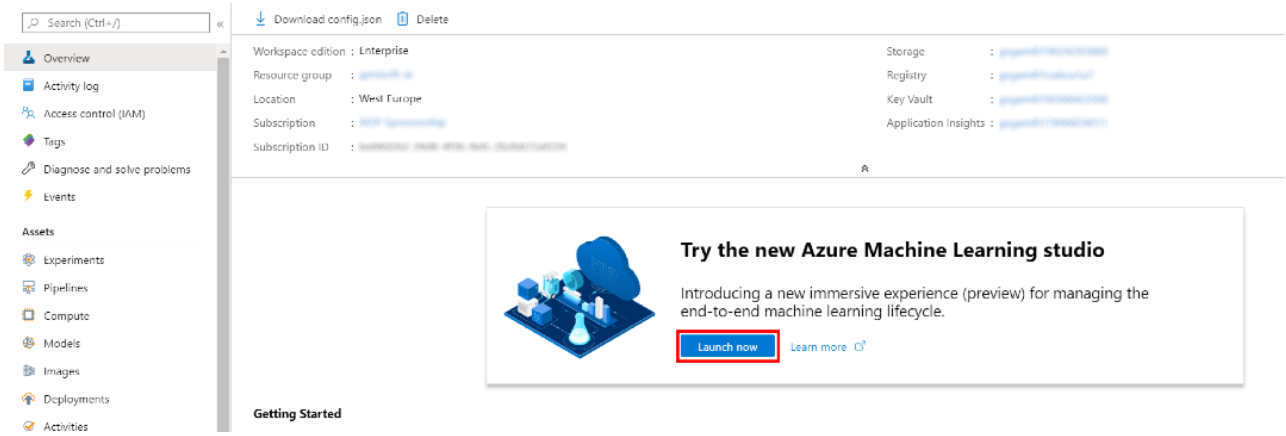
Overview

To access your data in your storage account, Azure Machine Learning offers datastores and datasets. Create an Azure Machine Learning datasets to interact with data in your datastores and package your data into a consumable object for machine learning tasks. Register the dataset to your workspace to share and reuse it across different experiments without data ingestion complexities.
Datasets can be created from local files, public urls, Azure Open Datasets, or specific file(s) in your datastores. To create a dataset from an in memory pandas dataframe, write the data to a local file, like a csv, and create your dataset from that file. Datasets aren't copies of your data, but are references that point to the data in your storage service, so no extra storage cost is incurred.
In this lab, we are using a subset of NYC Taxi & Limousine Commission - green taxi trip records available from Azure Open Datasets to show how you can register and version a Dataset using the AML designer interface. In the first exercises we use a modified version of the original CSV file, which includes collected records for five months (January till May). The second exercise demonstrates how we can create a new version of the initial dataset when new data is collected (in this case, we included records collected in June in the CSV file).

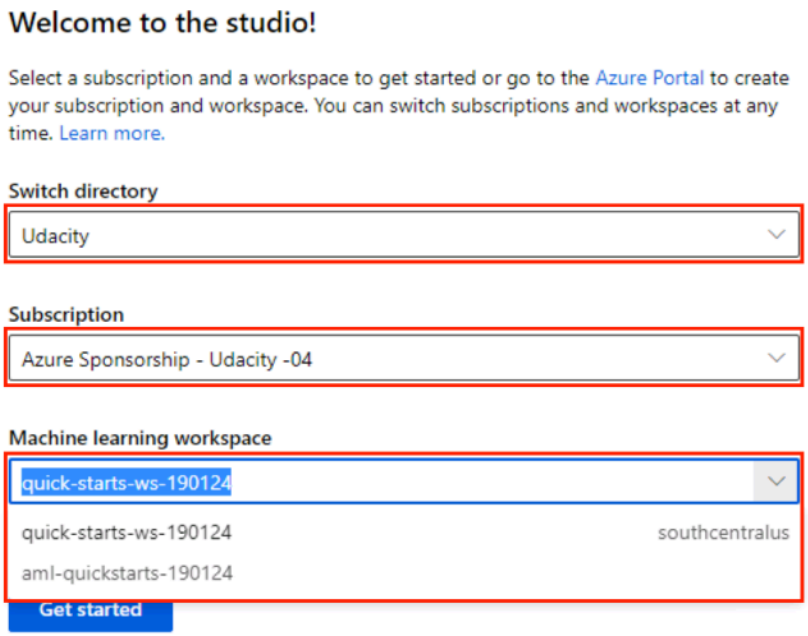
Exercise 1: Register Dataset with Azure Machine Learning studio

Task 1: Upload Dataset from web file

1. In Azure portal, open the available machine learning workspace.
2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

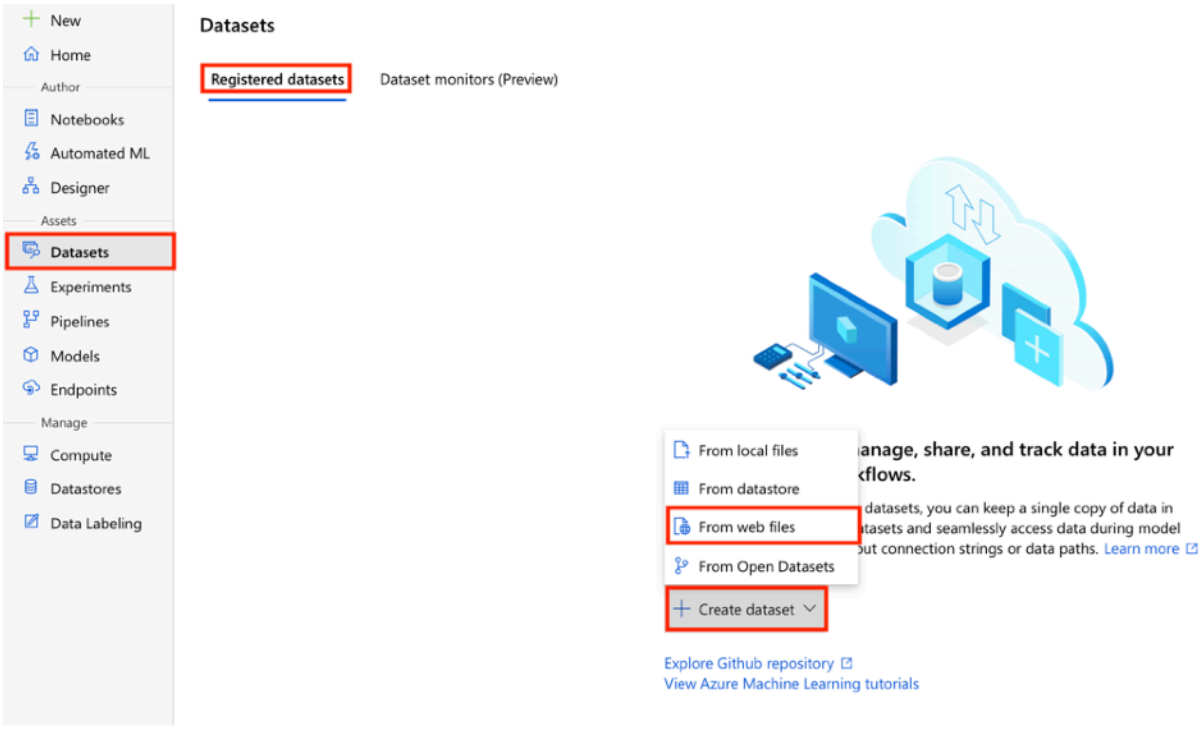


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:

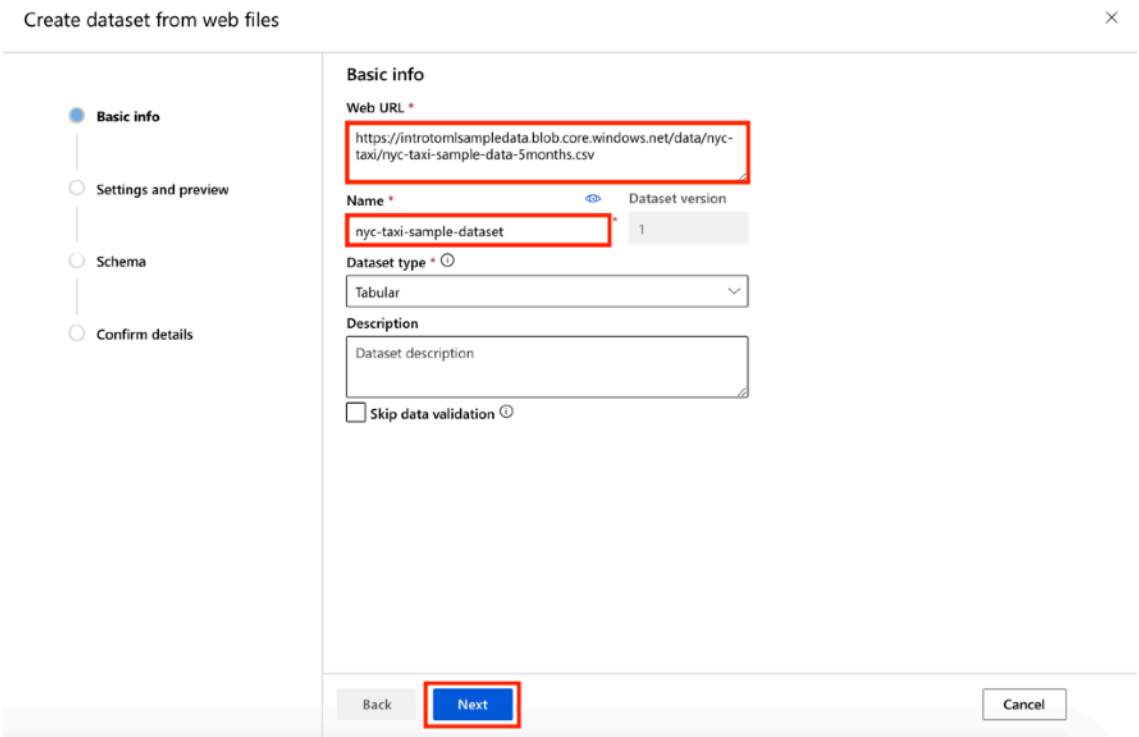


For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

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

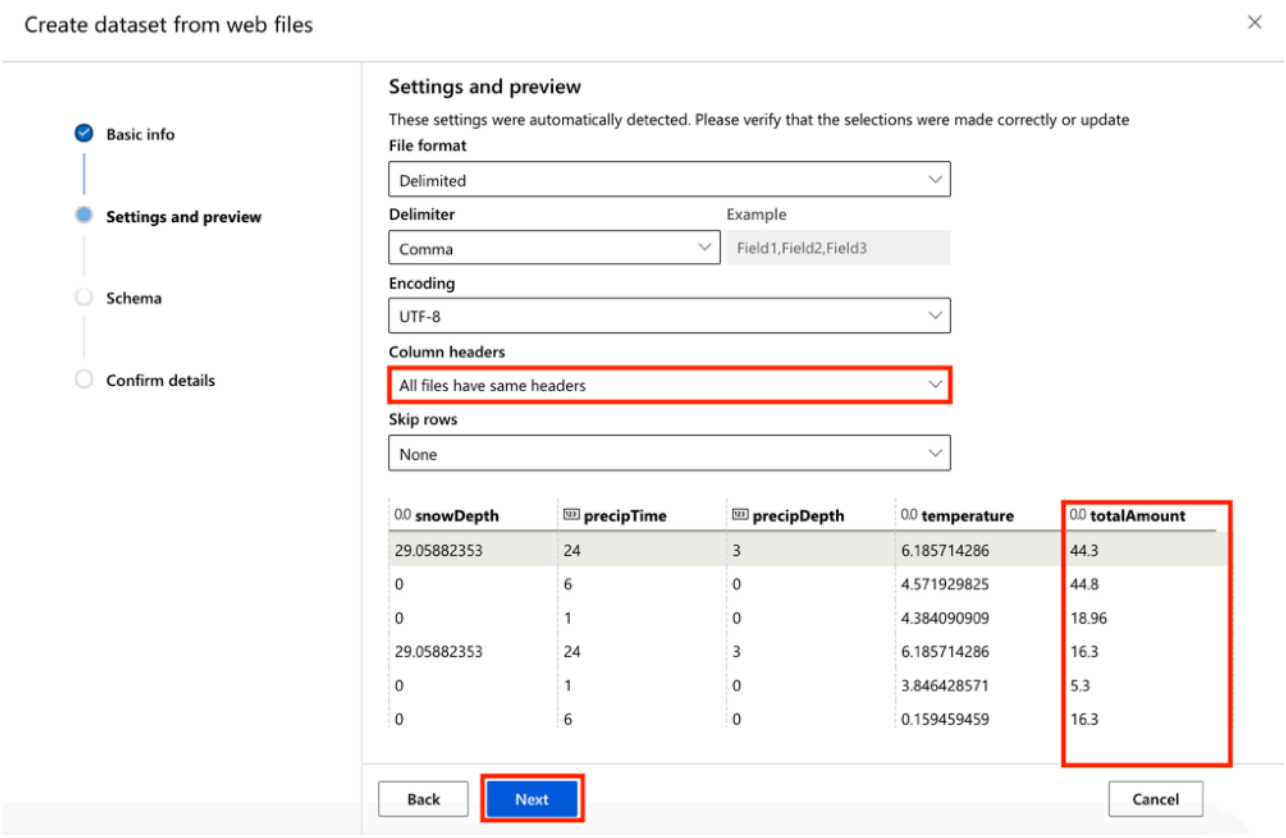


5. Provide the following information and then select **Next**:
    1. Web URL: `https://introtomlsampledata.blob.core.windows.net/data/nyc-taxi/nyc-taxi-sample-data-5months.csv`
    2. Name: `nyc-taxi-sample-dataset`



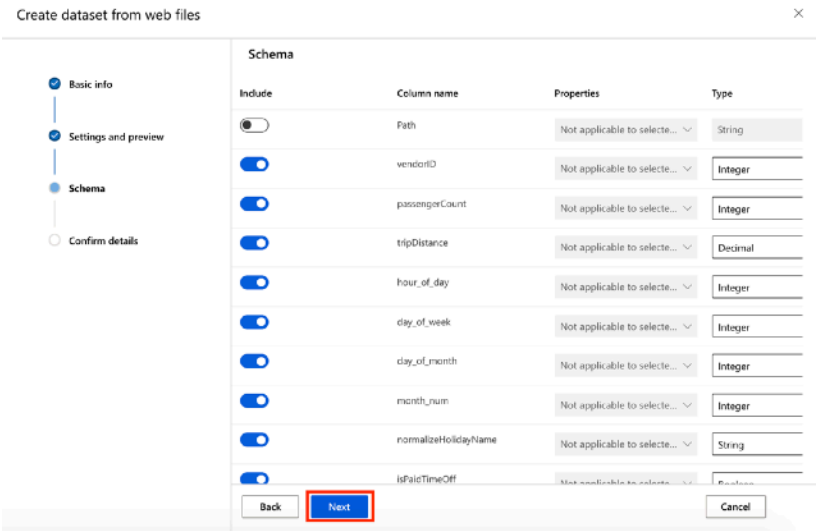
Task 2: Preview Dataset

1. On the Settings and preview panel, set the **Column headers** drop down to `All files have same headers`.
2. Scroll the data preview to right to observe the target column: `totalAmount`. After you are done reviewing the data, select **Next**



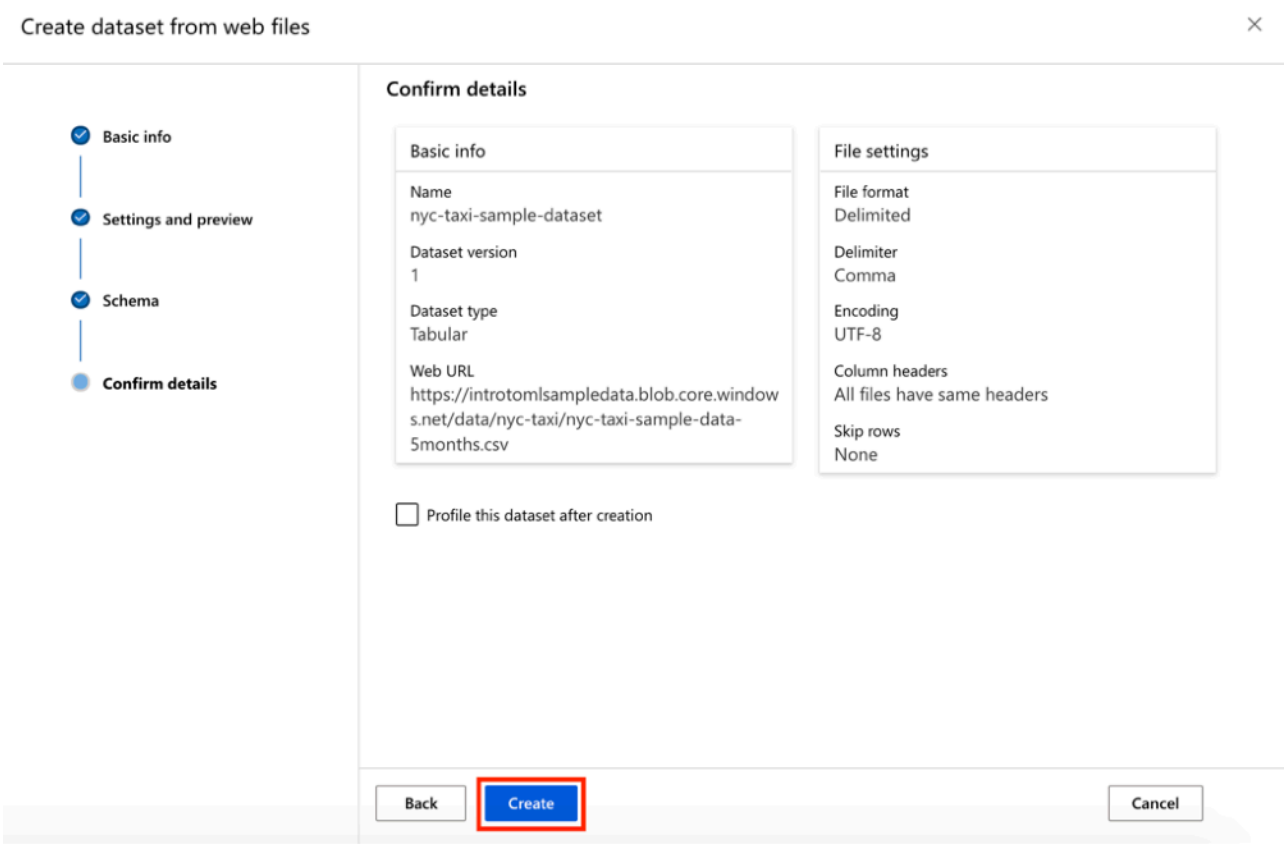
Task 3: Select Columns

1. Select columns from the dataset to include as part of your training data. Leave the default selections and select **Next**



Task 4: Create Dataset

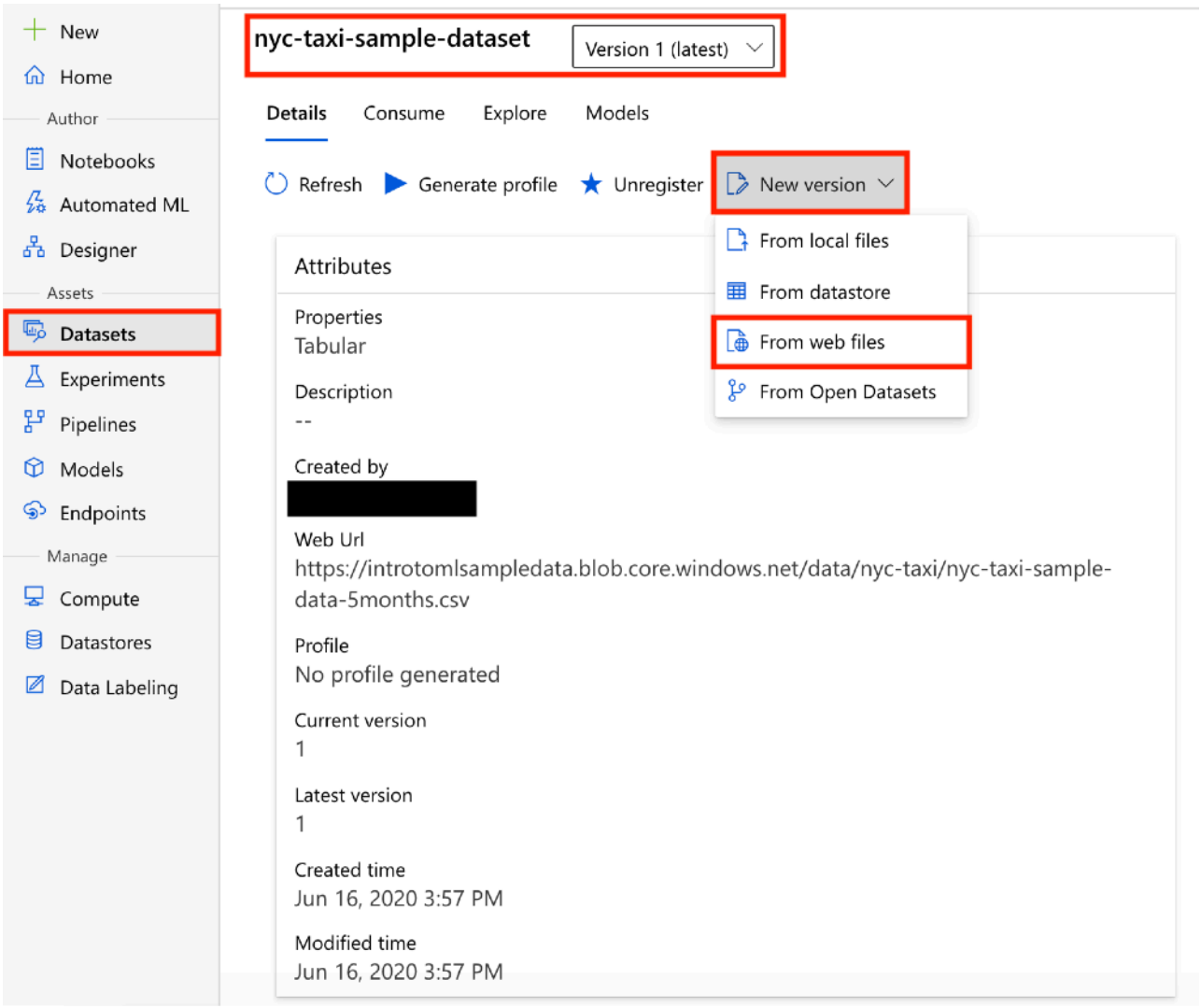
1. Confirm the dataset details and select **Create**



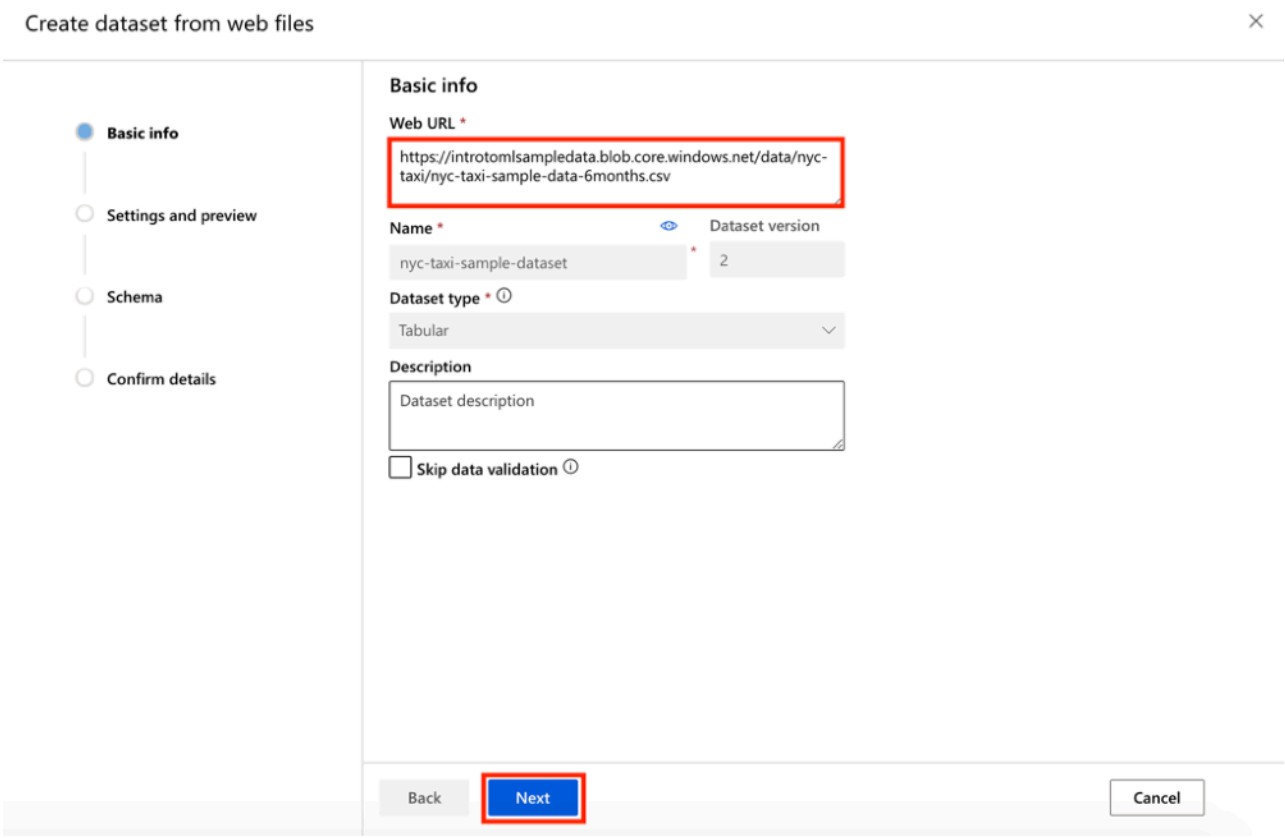
Exercise 2: Create a version of the existing Dataset

Task 1: Register new dataset version

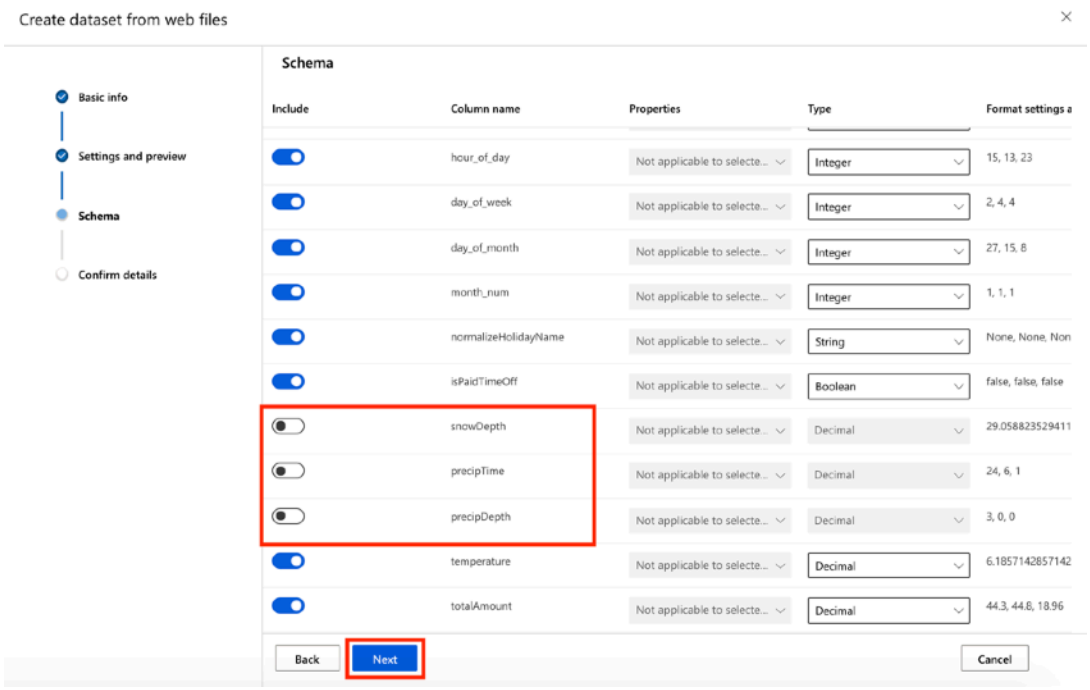
1. From the Azure Machine Learning studio, select **Datasets** and select the `nyc-taxi-sample-dataset` dataset created in the first exercise. This will open the `Dataset details` page.
2. Select **New version, From web files** to open the same `Create dataset from web files` dialog you already entered in the first exercise.



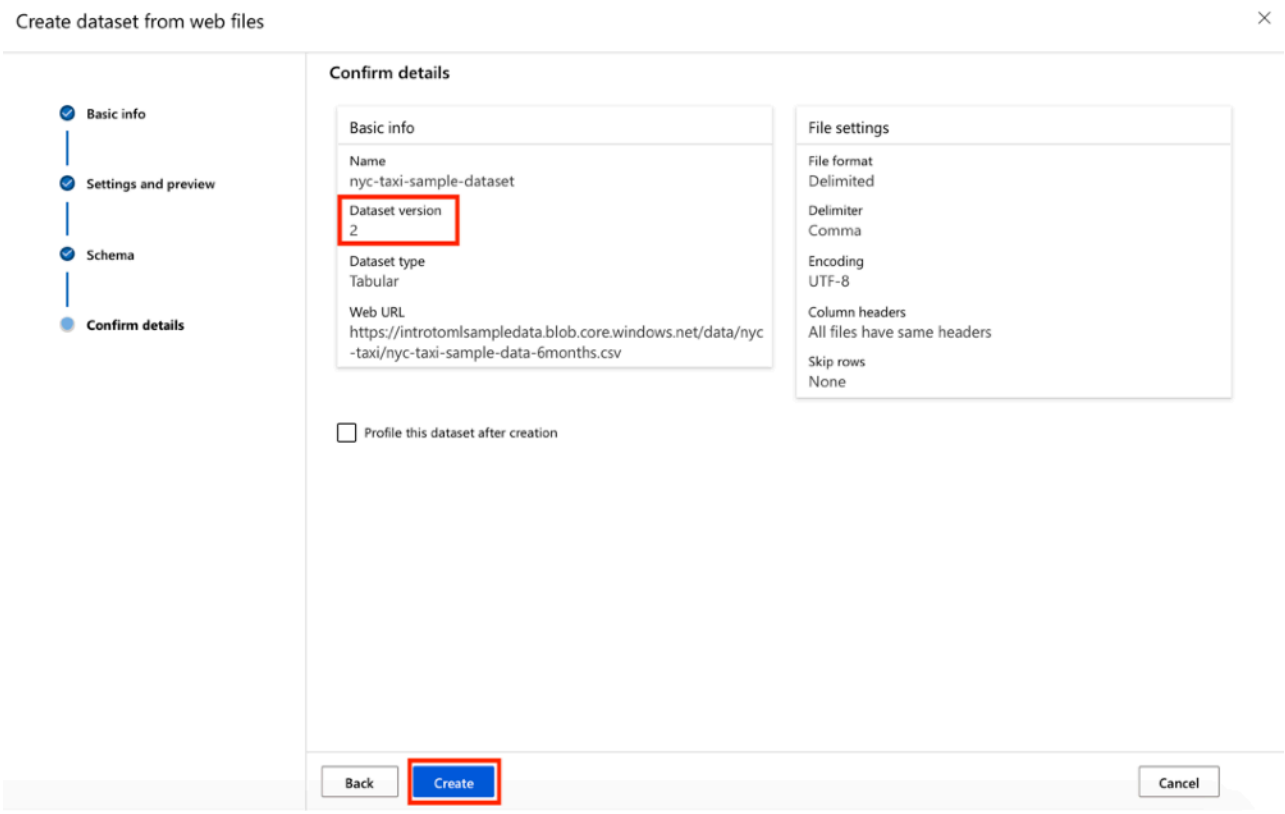
3. This time, the **Name** and **Dataset version** fields are already filled in for you. Provide the following information and select **Next** to move on to the next step:
    1. Web URL: https://introtomlsampledata.blob.core.windows.net/data/nyc-taxi/nyc-taxi-sample-data-6months.csv



4. Select `All files have the same headers` in the **Column headers** drop-down and move on to the schema selection step.
5. On the `Schema` page, let's suppose you decided to exclude some columns from your dataset. Exclude columns: **snowDepth**, **prcipTime**, **precipDepth**. Select **Next** to move on to the final step.

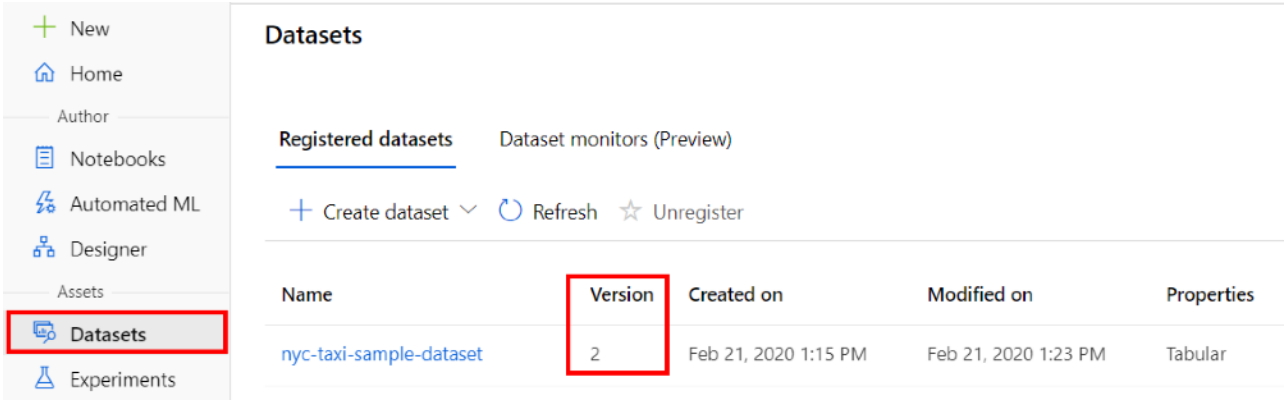


6. Notice the `Dataset version` value in the basic info section. Select **Create** to close the new version confirmation page.



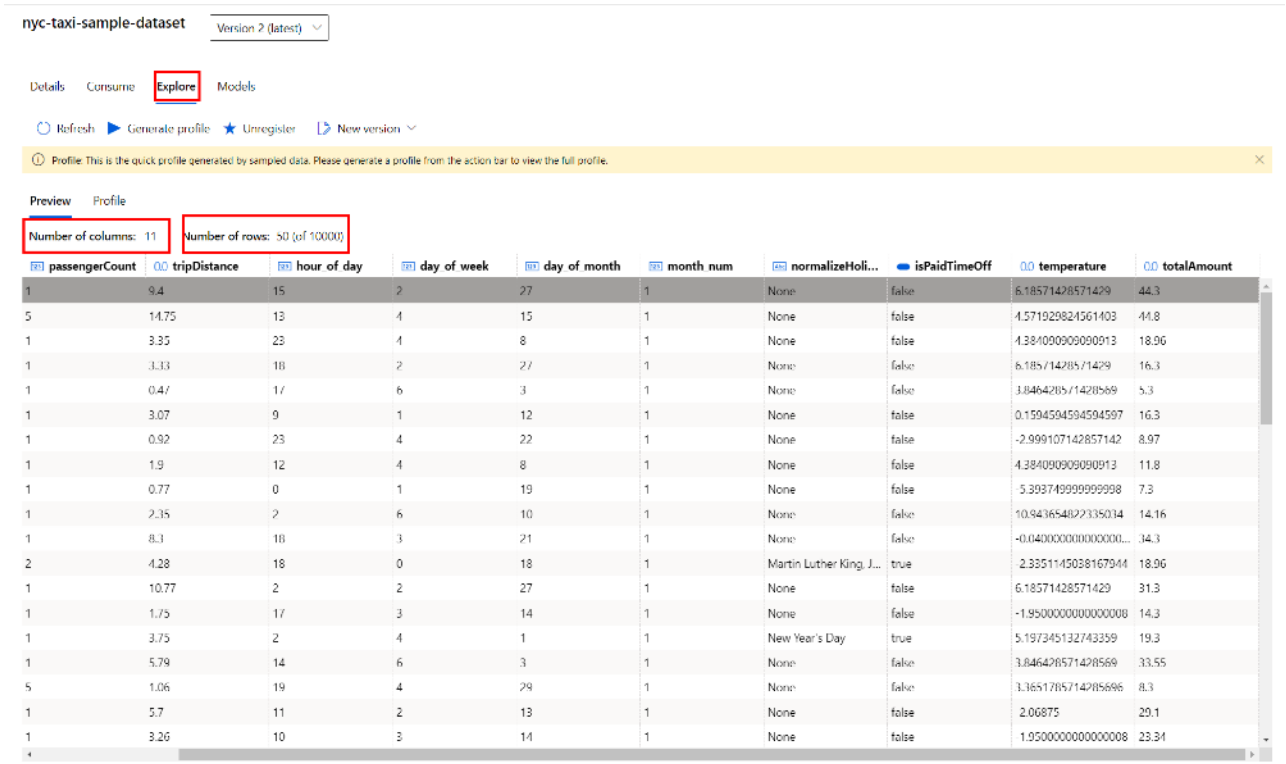
Task 2: Review both versions of the dataset

1. Back to the **Datasets** page, in the **Registered datasets** list, notice the version value for the `nyc-taxi-sample-dataset` dataset.



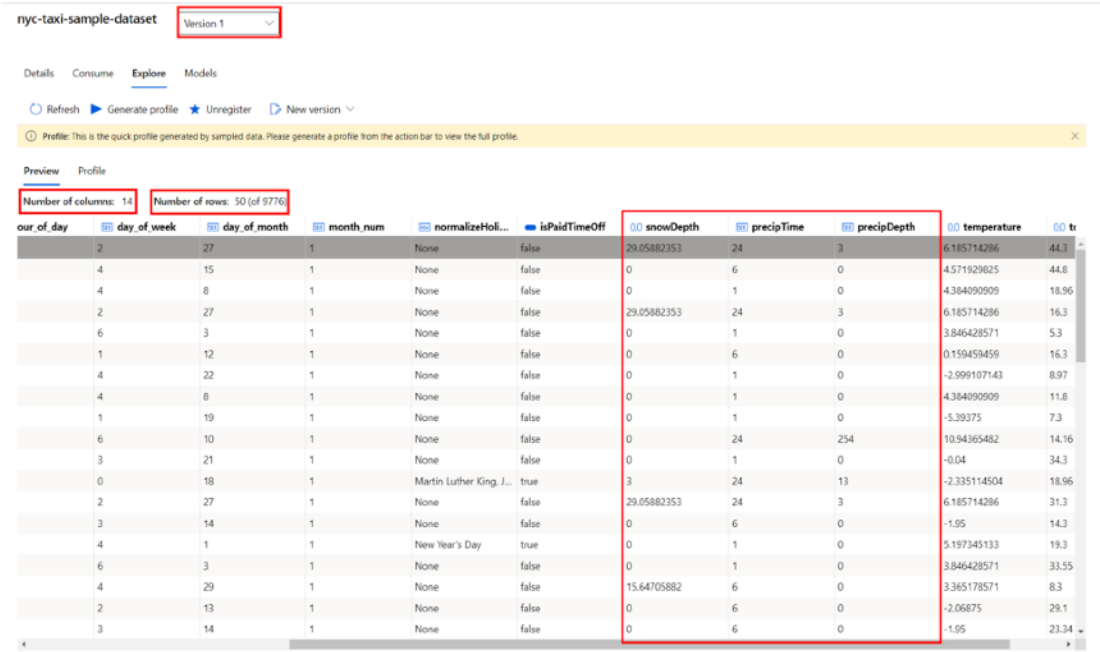
2. Select the `nyc-taxi-sample-dataset` dataset link to open the dataset details page, where **Version 2(latest)** is automatically selected. Go to the **Explore** section to observe the structure and content of the new version. Notice the columns and rows structure in the dataset preview pane:
   - **Number of columns**: 11

- **Number of rows**: 10000
- Scroll right to check that the three excluded columns are missing (**snowDepth**, **prcipTime**, **precipDepth**)



passengerCount	tripDistance	hour_of_day	day_of_week	day_of_month	month_num	normalizeHoli...	isPaidTimeOff	temperature	totalAmount
1	9.4	15	2	27	1	None	false	6.18571428571429	44.3
5	14.75	13	4	15	1	None	false	4.571929824561403	44.8
1	3.35	23	4	8	1	None	false	4.384090909090913	18.96
1	3.33	18	2	27	1	None	false	6.185714285714429	16.3
1	0.47	17	6	3	1	None	false	3.8464285714285689	5.3
1	3.07	9	1	12	1	None	false	0.1594594594594597	16.3
1	0.92	23	4	22	1	None	false	-2.999107142857142	8.97
1	1.9	12	4	8	1	None	false	4.384090909090913	11.8
1	0.77	0	1	19	1	None	false	-5.393749999999998	7.3
1	2.35	2	6	10	1	None	false	10.943654822335034	14.16
1	8.3	18	3	21	1	None	false	-0.040000000000000...	34.3
2	4.28	18	0	18	1	Martin Luther King, J...	true	2.3351145038167944	18.96
1	10.77	2	2	27	1	None	false	6.18571428571429	31.3
1	1.75	17	3	14	1	None	false	-1.9500000000000008	14.3
1	3.75	2	4	1	1	New Year's Day	true	5.197345132743359	19.3
1	5.79	14	6	3	1	None	false	3.8464285714285689	33.55
5	1.06	19	4	29	1	None	false	3.3651785714285696	8.3
1	5.7	11	2	13	1	None	false	2.06875	29.1
1	3.26	10	3	14	1	None	false	1.9500000000000008	23.34

3. Select **Version 1** from the drop-down near the dataset name title and notice the changing values for:
   - **Number of columns**: 14 (since the previous version still contains the three excluded columns)
   - **Number of rows**: 9776 (since the previous version contains only data for 5 months)



our_of_day	day_of_week	day_of_month	month_num	normalizeHoli...	isPaidTimeOff	snowDepth	precipTime	precipDepth	temperature	tr
	2	27	1	None	false	29.05882353	24	3	6.185714286	44.3
	4	15	1	None	false	0	6	0	4.571929825	44.8
	4	8	1	None	false	0	1	0	4.384090909	18.96
	2	27	1	None	false	29.05882353	24	3	6.185714286	16.3
	6	3	1	None	false	0	1	0	3.846428571	5.3
	1	12	1	None	false	0	6	0	0.159459459	16.3
	4	22	1	None	false	0	1	0	-2.999107143	8.97
	4	8	1	None	false	0	1	0	4.384090909	11.8
	1	19	1	None	false	0	1	0	-5.39375	7.3
	6	10	1	None	false	0	24	254	10.94365482	14.16
	3	21	1	None	false	0	1	0	-0.04	34.3
	0	18	1	Martin Luther King, J...	true	3	24	13	-2.335114504	18.96
	2	27	1	None	false	29.05882353	24	3	6.185714286	31.3
	3	14	1	None	false	0	6	0	-1.95	14.3
	4	1	1	New Year's Day	true	0	1	0	5.197345133	19.3
	6	3	1	None	false	0	1	0	3.846428571	33.55
	4	29	1	None	false	15.64705882	6	0	3.365178571	8.3
	2	13	1	None	false	0	6	0	-2.06875	29.1
	3	14	1	None	false	0	6	0	-1.95	23.34

Next Steps

Congratulations! You have now explored a first simple scenario for dataset versioning using the Azure Machine Learning studio. You found out how you can create and version a simple dataset when new training data is available. 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.

Lab 4 - Engineer and select features

Overview

This lab demonstrates the feature engineering process for building a regression model using bike rental demand prediction as an example. In machine learning predictions, effective feature engineering will lead to a more accurate model. We will use the Bike Rental UCI dataset as the input raw data for this experiment. This dataset is based on real data from the Capital Bikeshare company, which operates a bike rental network in Washington DC in the United States. The dataset contains 17,379 rows and 17 columns, each row representing the number of bike rentals within a specific hour of a day in the years 2011 or 2012. Weather conditions (such as temperature, humidity, and wind speed) were included in this raw feature set, and the dates were categorized as holiday vs. weekday etc.
The field to predict is `cnt` which contains a count value ranging from 1 to 977, representing the number of bike rentals within a specific hour. Our main goal is to construct effective features in the training data, so we build two models using the same algorithm, but with two different datasets. Using the Split Data module in the visual designer, we split the input data in such a way that the training data contains records for the year 2011, and the testing data, records for 2012. Both datasets have the same raw data at the origin, but we added different additional features to each training set:

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

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

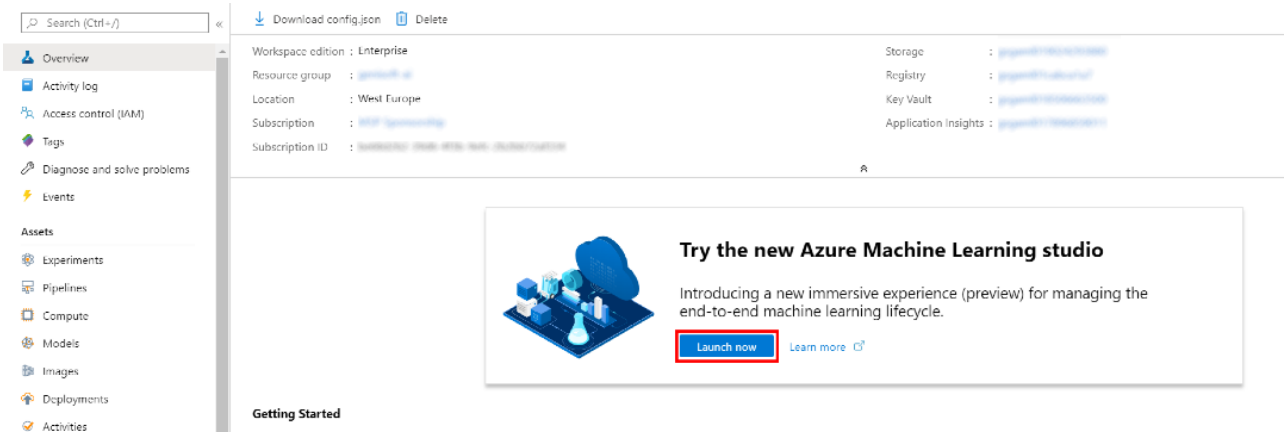
- Training set 1: feature set A only
- Training set 2: feature sets A+B

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

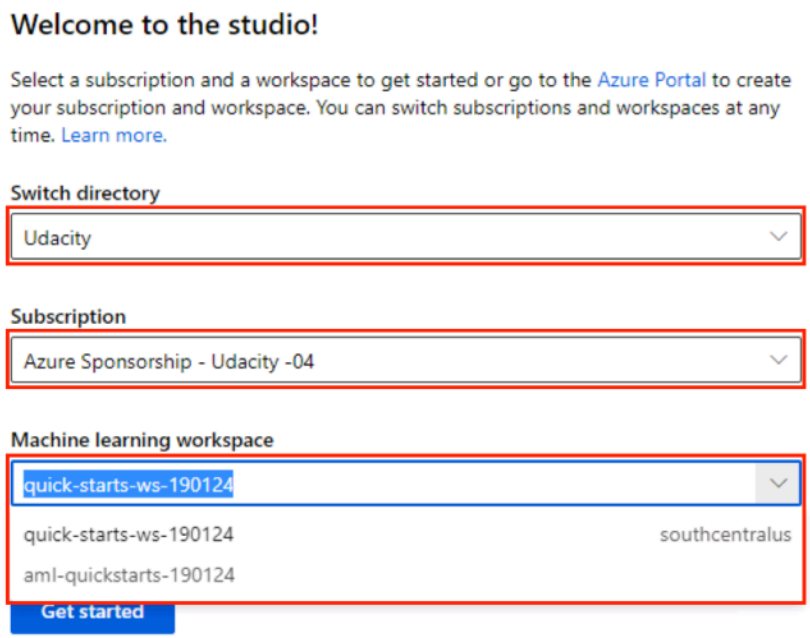
Exercise 1: Data pre-processing using the Pipeline Authoring Editor

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.

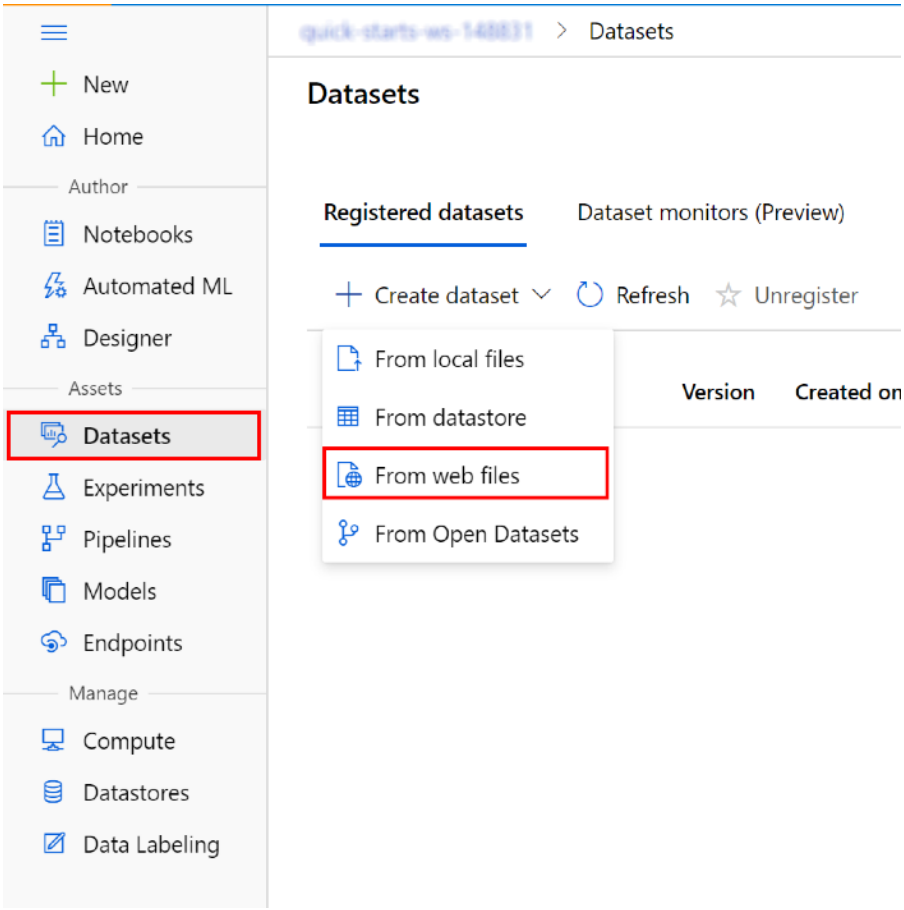


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:

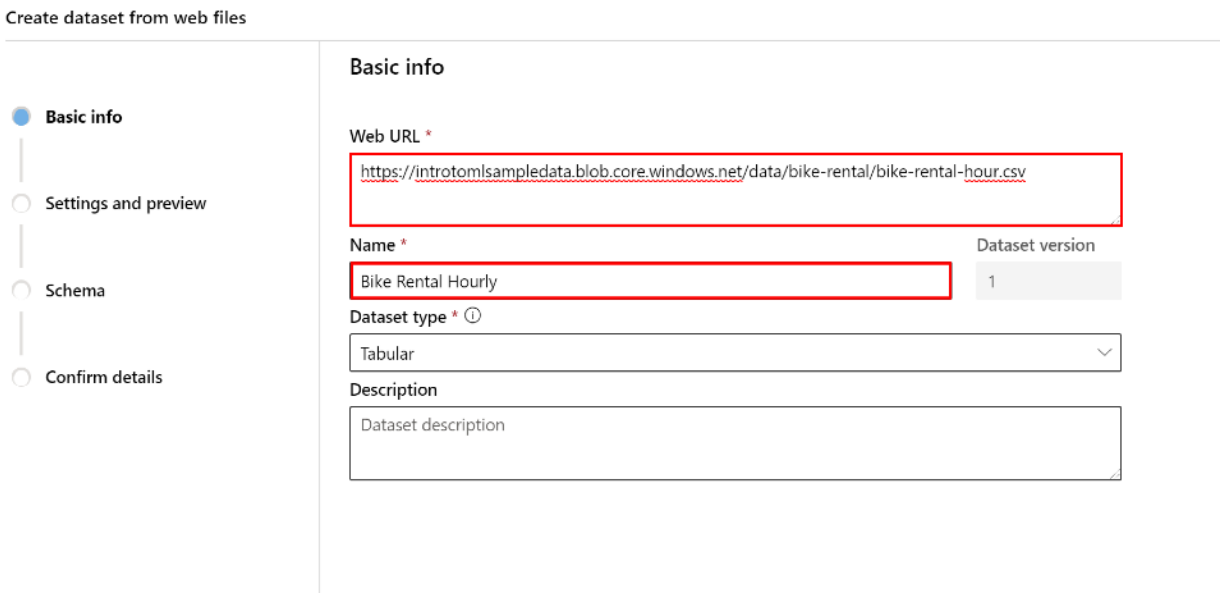


For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

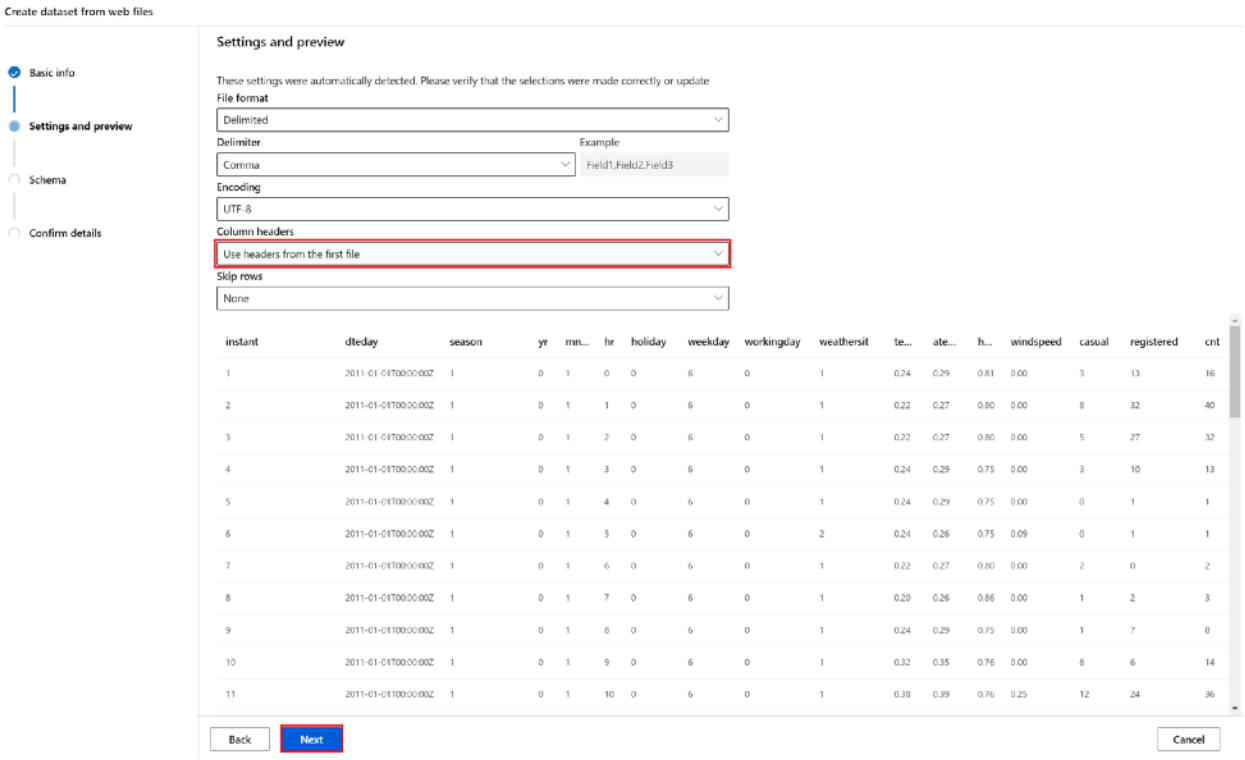
4. From the studio, select **Datasets, + Create dataset, From web files**. This will open the `Create dataset from web files` dialog on the right.



5. In the Web URL field provide the following URL for the training data file: `https://introtomlsampledata.blob.core.windows.net/data/bike-rental/bike-rental-hour.csv`
6. Provide `Bike Rental Hourly` as the Name, leave the remaining values at their defaults and select **Next**.

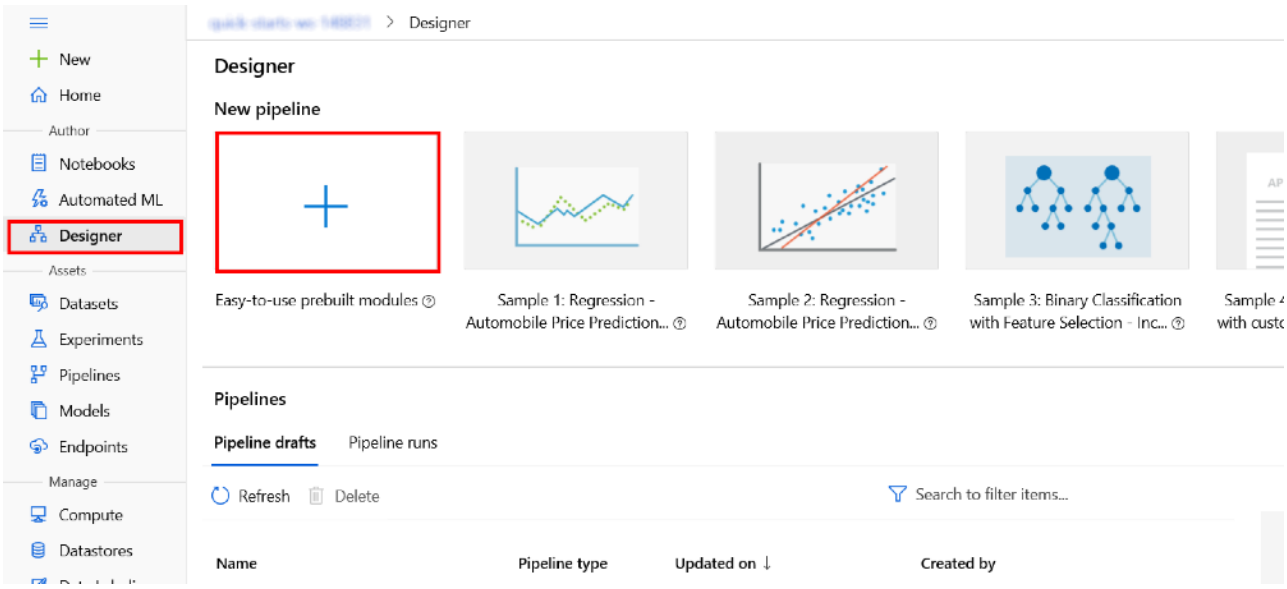


7. Select the option to `Use headers from the first file` in the **Settings and preview** dialog and then select **Next**, **Next** and **Create** to confirm all details in registering the dataset.



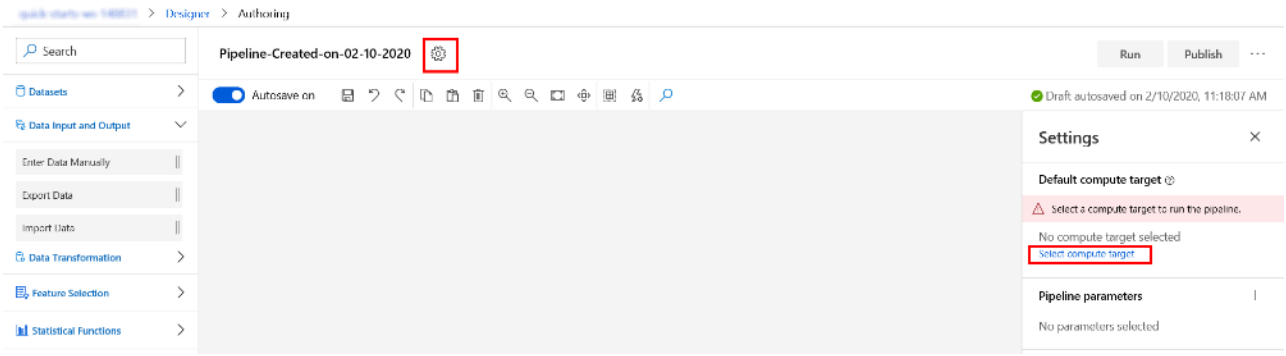
Task 2: Open Pipeline Authoring Editor

1. From the left navigation, select **Designer, +**. This will open a `visual pipeline authoring editor`.



Task 3: Setup Compute Target

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

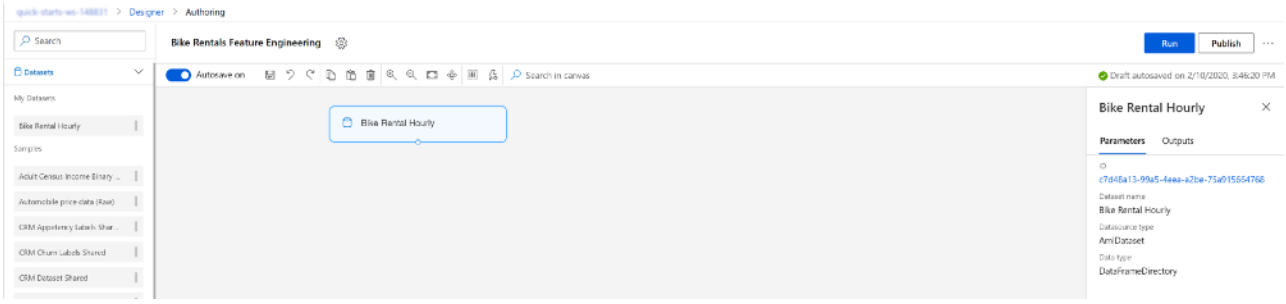


2. In the `Set up compute target` editor, select the existing compute target, choose a name for the pipeline draft: `Bike Rental Feature Engineering` and then select **Save**.

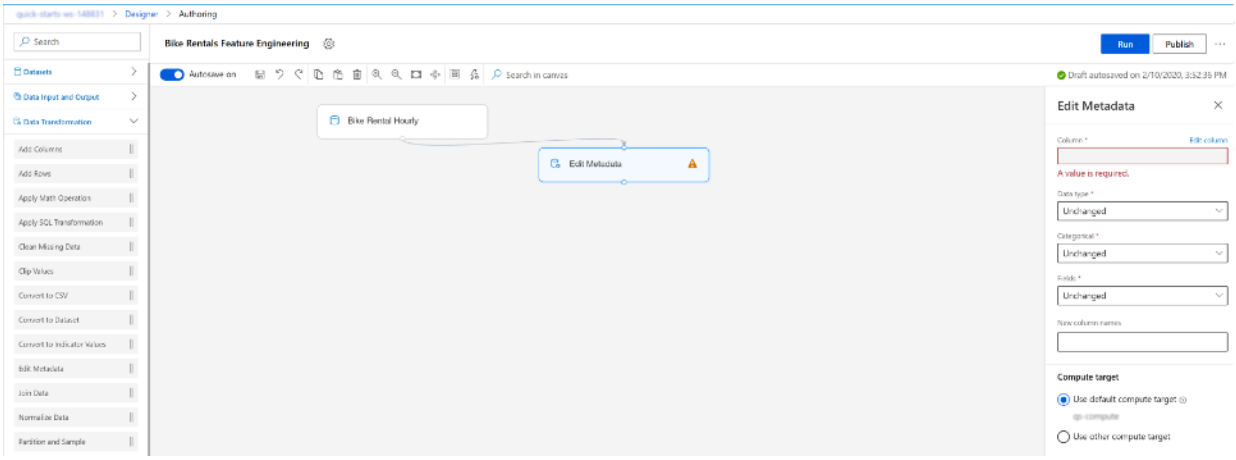
Note: If you are facing difficulties in accessing pop-up windows or buttons in the user interface, please refer to the Help section in the lab environment.

Task 4: Select columns in the dataset

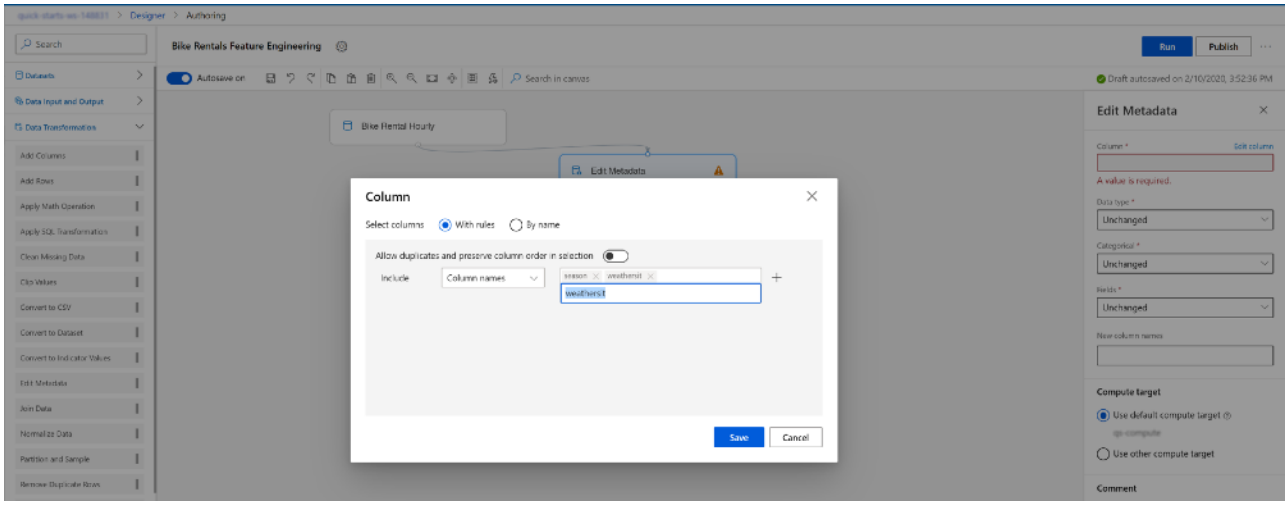
1. Drag and drop on the canvas, the available `Bike Rental Hourly` dataset under the **Datasets** category on the left navigation.



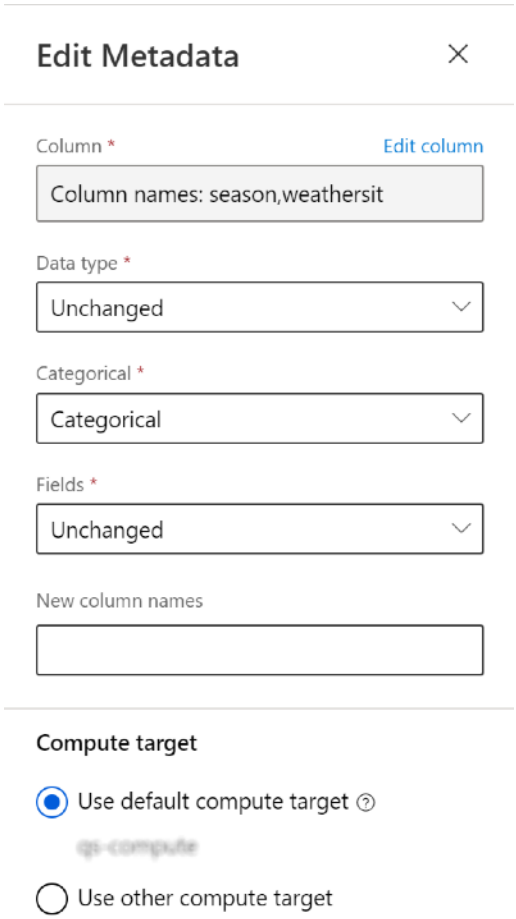
2. Under the **Data transformation** category drag and drop the `Edit metadata` module.



3. Edit the column list by selecting **Edit columns** on the right pane. Add the `season` and `weathersit` column and select **Save**.

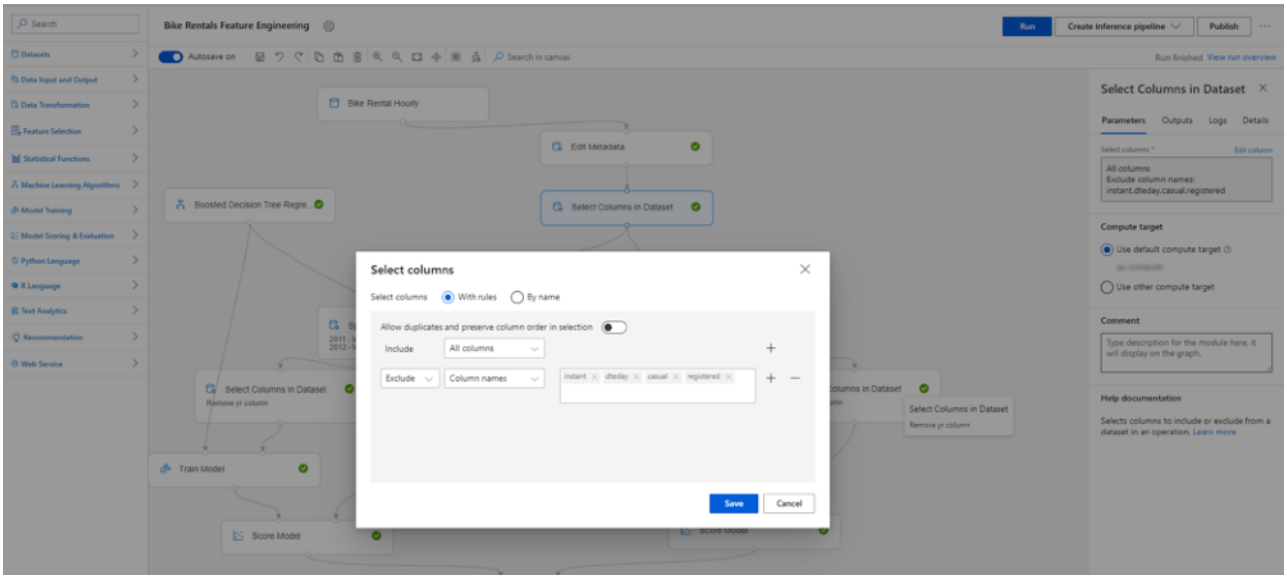


4. Configure the Edit metadata module by selecting the `Categorical` attribute for the two columns.

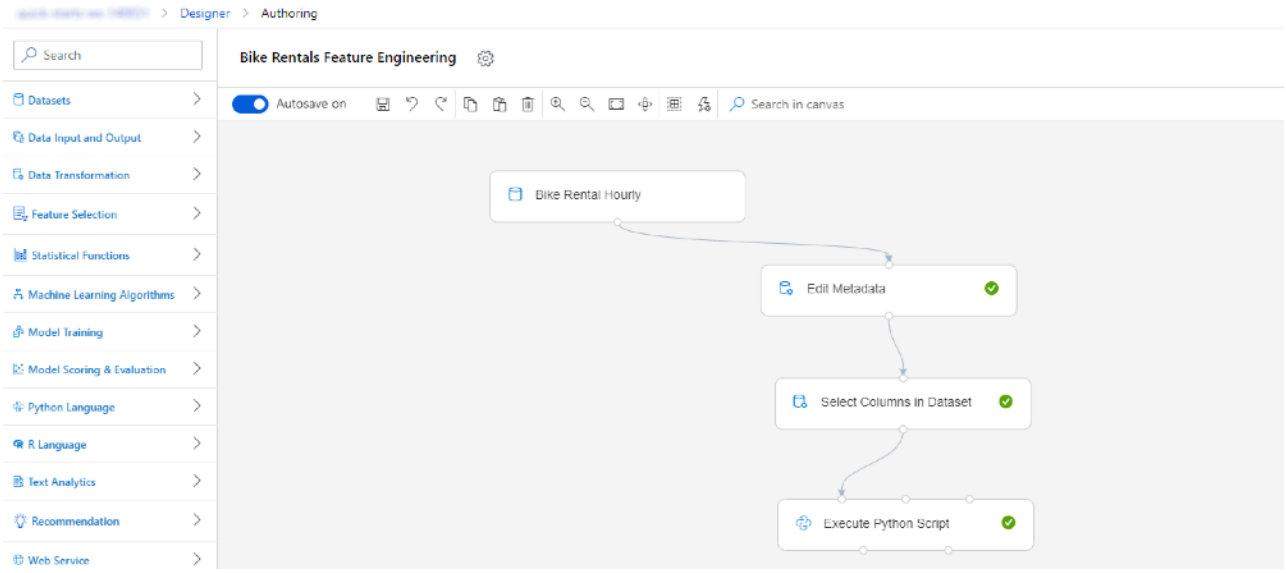


5. Next, use the **Select columns in Dataset** module under the **Data transformation** category and configure it as follows:

- Include all columns
- Exclude column names: `instant`, `dteday`, `casual` ,`registered`
- Use default compute target: `aml-compute`



6. Connect the output from the **Edit columns** module to the input of the **Select columns in Dataset** module.
7. Under the **Python Language** category on the left, select the **Execute Python Script** module and connect it with the **Select Columns in Dataset** module. Make sure the connector is connected to the very first input of the **Execute Python Script** module.



8. We are using the Python script to append a new set of features to the dataset: number of bikes that were rented in each of the previous 12 hours. Feature set B captures very recent demand for the bikes. This will be the B set in the described feature engineering approach.

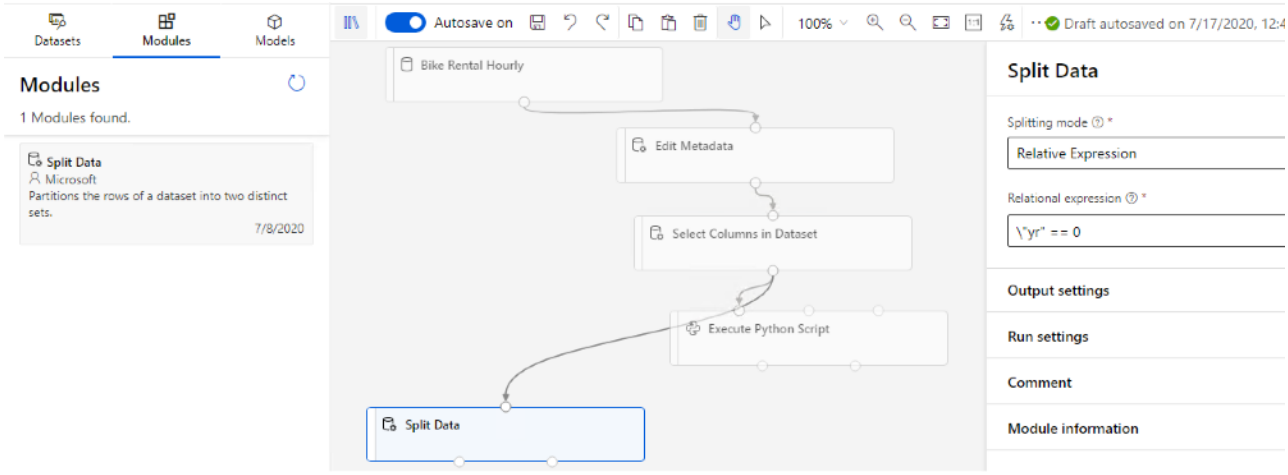
Select **Edit code** and use the following lines of code:

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

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

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

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

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

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

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

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

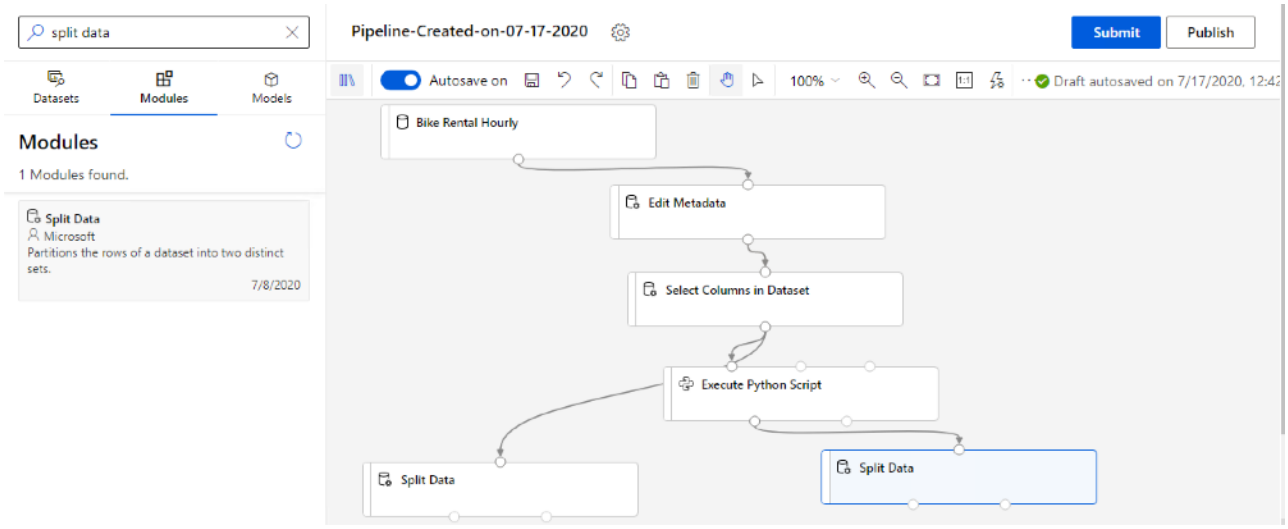
Don’t worry if you do not fully understand the details of the Python code above. For now, it’s enough to keep in mind that is adds 12 new columns to your dataset containing the number of bikes that were rented in each of the previous 12 hours.

Task 5: Split data into train and test datasets

1. Use the **Split Data** module under the **Data Transformation** module and connect its input with output from the **Select Columns in Dataset** module. Use the following configuration:
    - Splitting mode: `Relative Expression`
    - Relational expression: `\"yr" == 0`

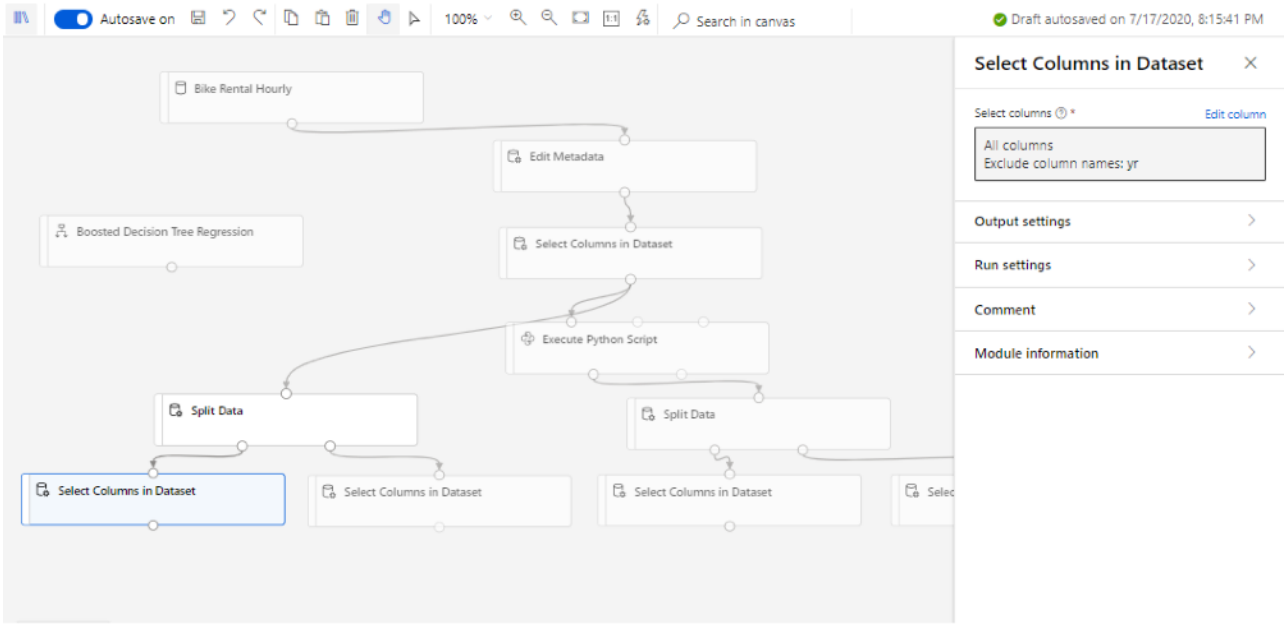


2. Select the **Split Data** module block and use the menu buttons to Copy and Paste it on the canvas. Connect the second one to the output of the Python Script execution step, which is the featured B set.

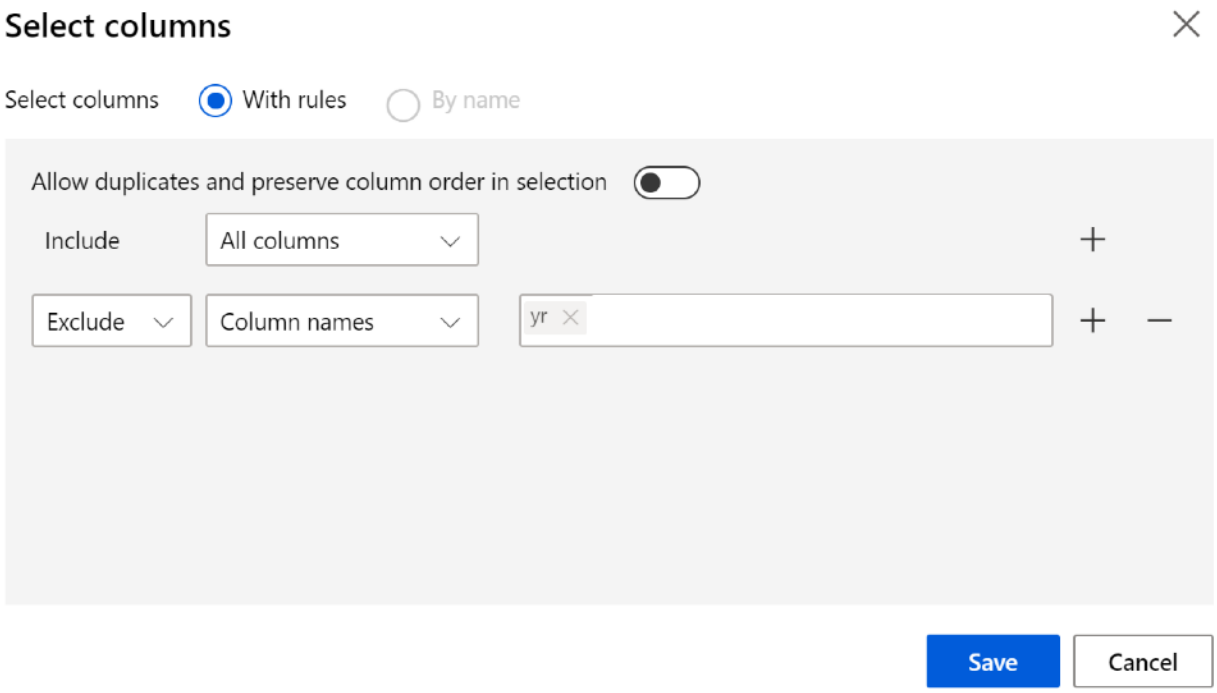


Task 6: Select columns from the test and training resulted sets

1. Next, using the **Select columns** module under the **Data transformation** category, create four identical modules to exclude the `yr` column from all the outputs: test and training sets in both branches: A and A+B.

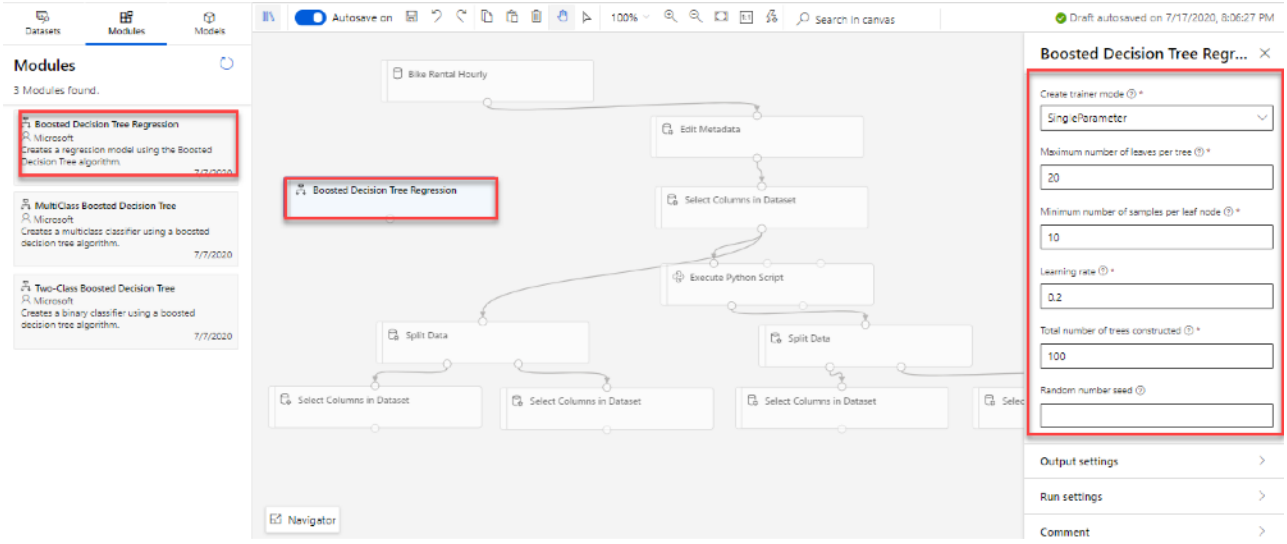


2. Use the following structure for the columns field in each module:

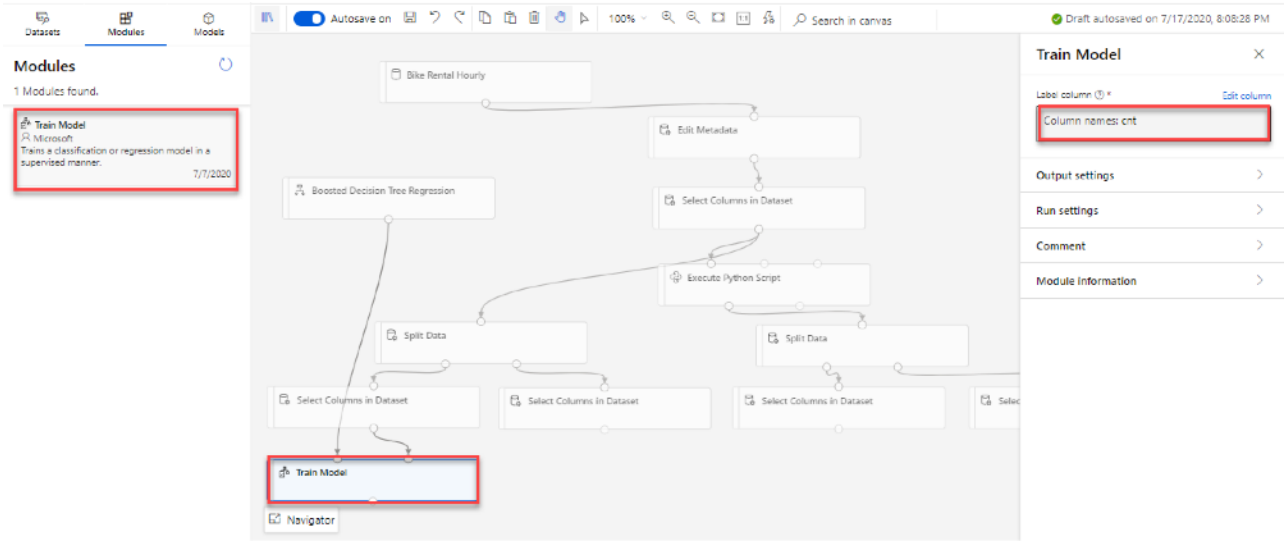


Task 7: Create the regression model

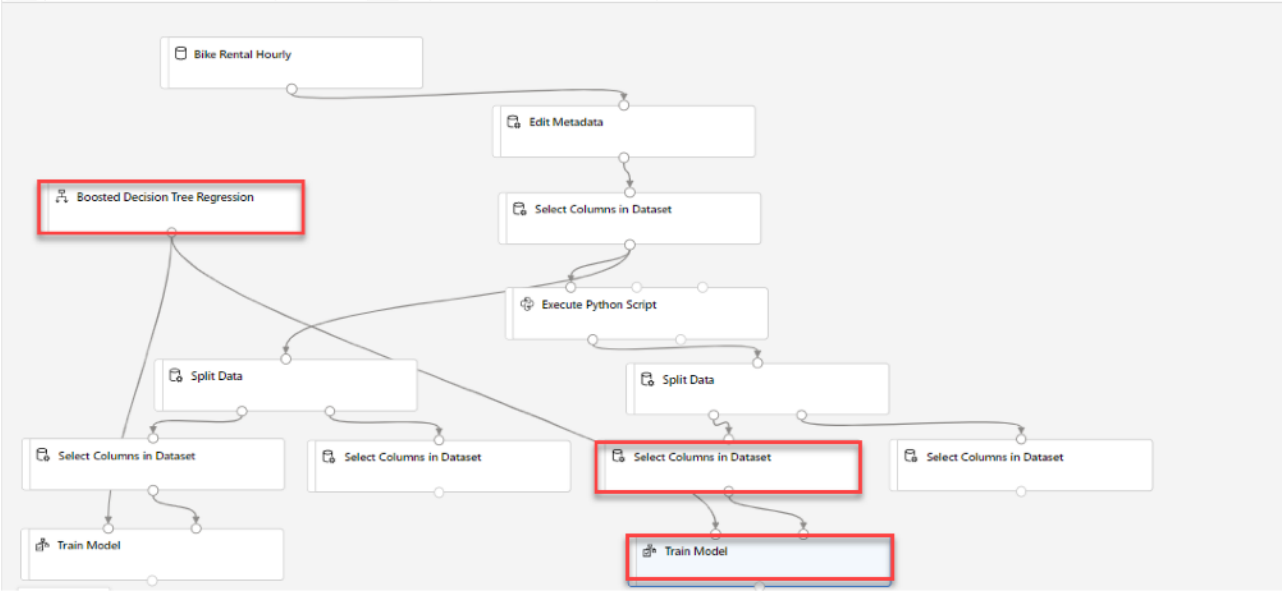
1. Under the **Machine Learning Algorithms, Regression** category, select the **Boosted Decision Tree Regression** module. Drag and drop it on the canvas and use the default settings provided.



2. Next, use the **Train model** module under the **Model training** category and enter the `cnt` column in the **Label column** field.
3. Link the **Boosted Decision Tree Regression** module as the first input and the training dataset as the second input like in the image below.

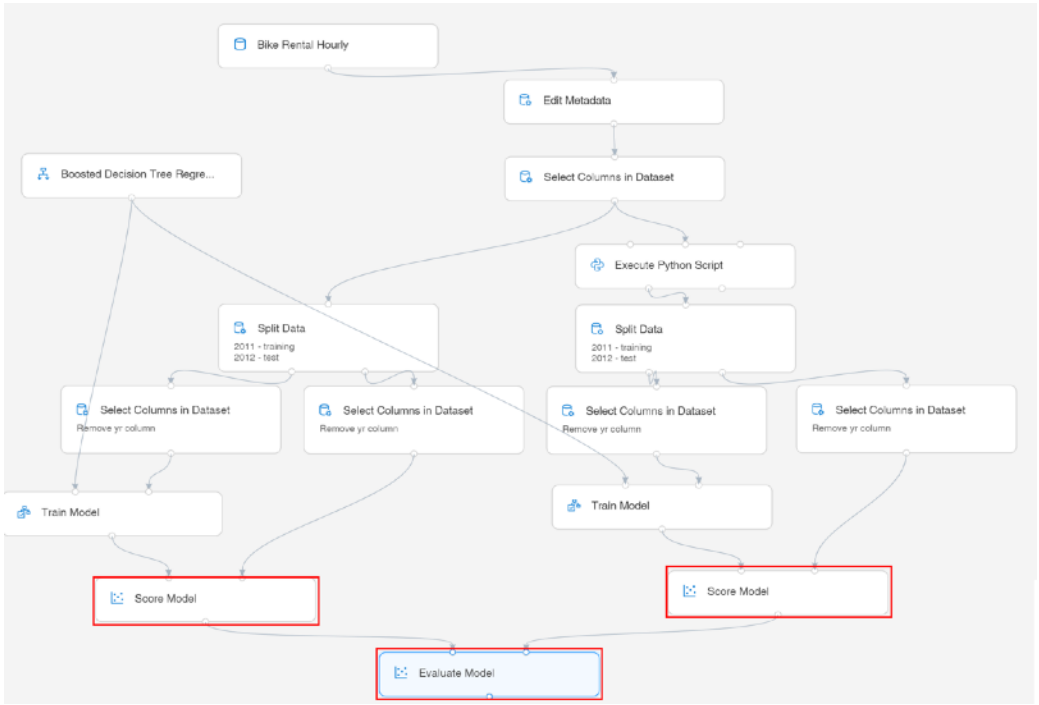


4. Use the exact same configuration on the right branch that uses the output from the Python Script.

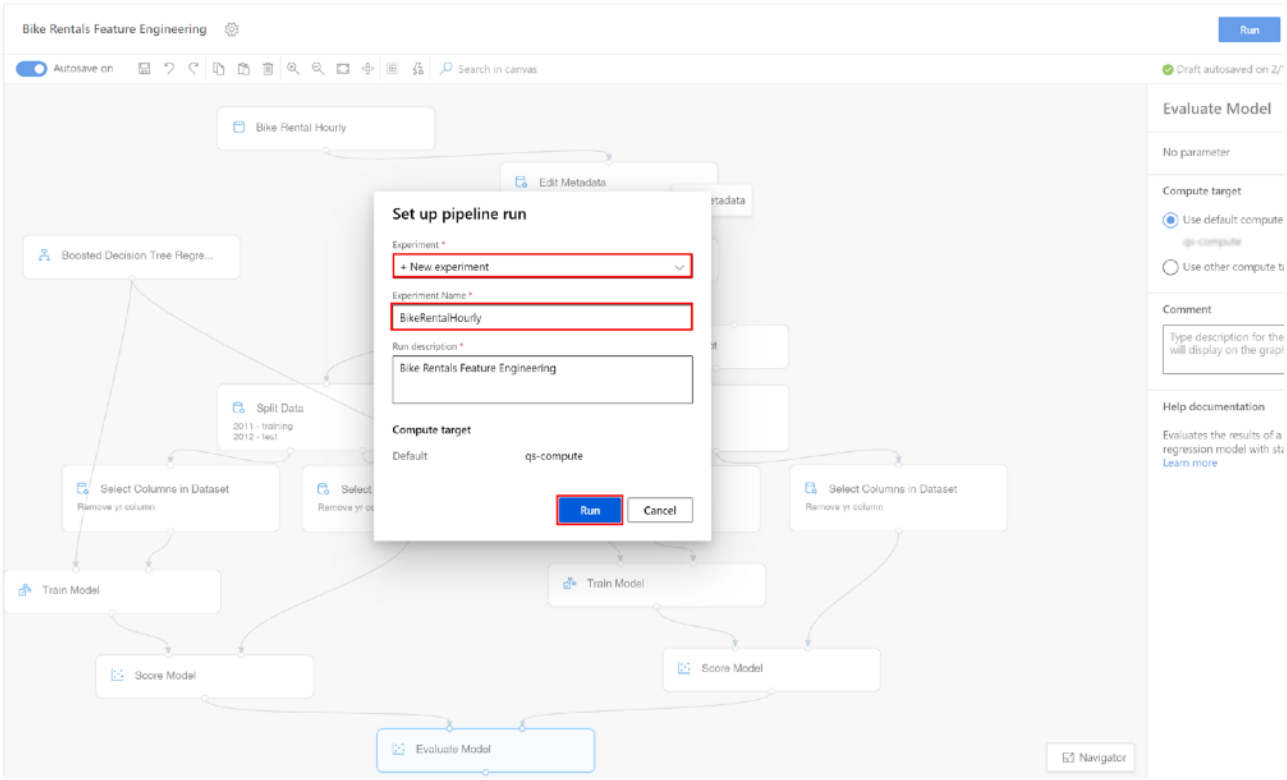


Task 8: Evaluate and score models

1. Use two **Score Model** modules (under the **Model Scoring and Evaluation** category) and link on the input the two trained models and the test datasets.
2. Drag the **Evaluate Model** module which stands in the same category, **Model Scoring and Evaluation** and link it to the two **Score Model** modules.

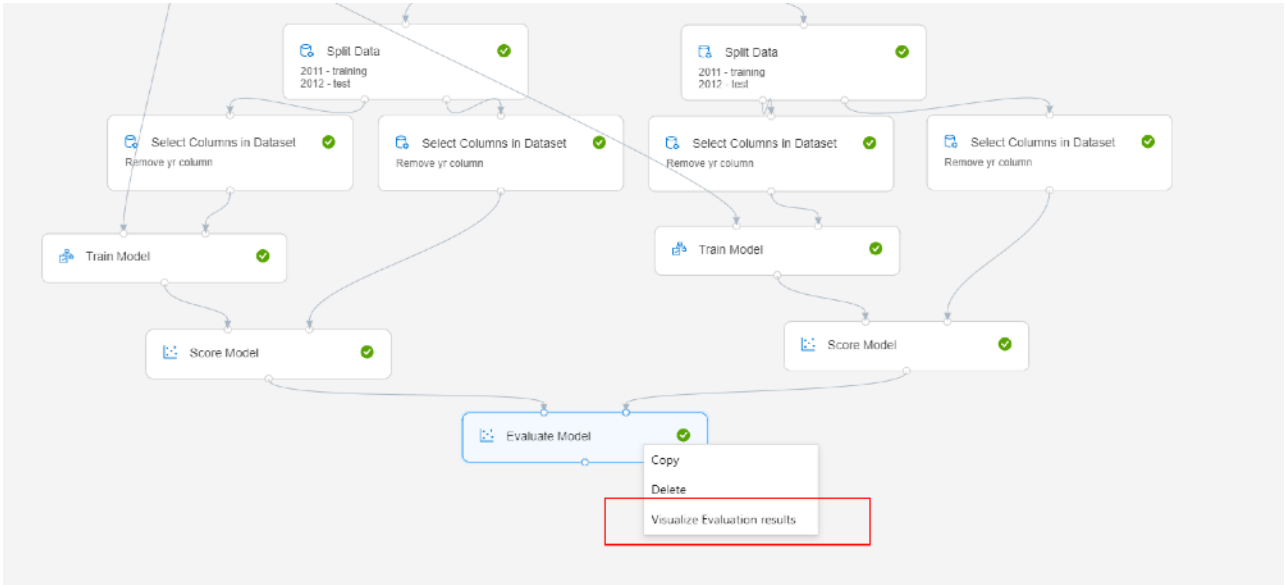


3. Select **Submit** to open the `Setup pipeline run` editor. In the `Setup pipeline run editor`, select **Experiment, Create new** and provide `New experiment name:` **BikeRentalHourly**.



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

4. Wait for pipeline run to complete. It will take around **10 minutes** to complete the run.
5. Once the pipeline execution completes, right click on the **Evaluate Model** module and select **Visualize Evaluation results**.



6. The **Evaluate Model result visualization** popup shows the results of the evaluation.

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

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

Next Steps

Congratulations! You have trained and compared performance of two models using the same algorithm, but with two different datasets. 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.

Lab 5 - Train and Evaluate a Model

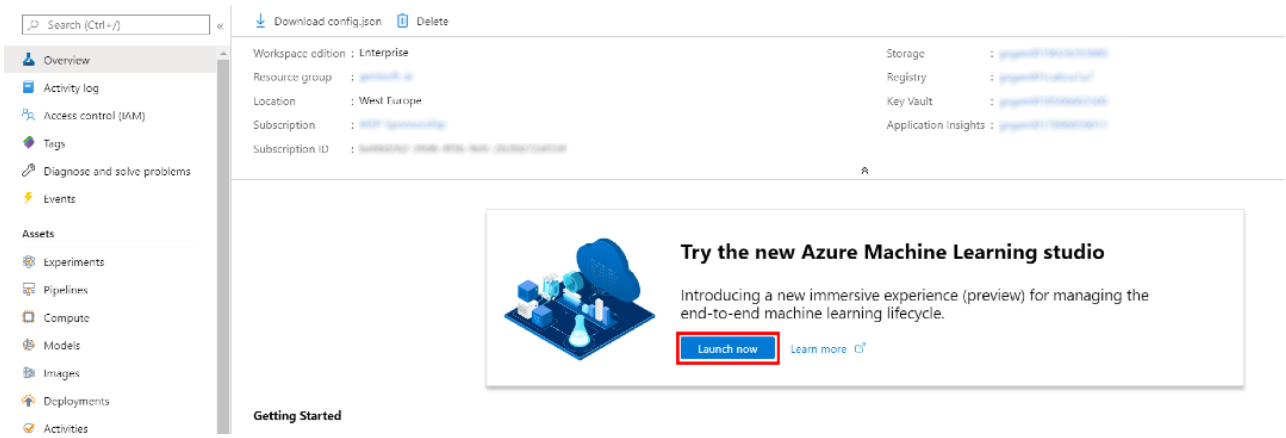
Overview

Azure Machine Learning designer (preview) gives you a cloud-based interactive, visual workspace that you can use to easily and quickly prep data, train and deploy machine learning models. It supports Azure Machine Learning compute, GPU or CPU. Machine Learning designer also supports publishing models as web services on Azure Kubernetes Service that can easily be consumed by other applications.
In this lab, we will be using the `Flight Delays` data set that is enhanced with the weather data. Based on the enriched dataset, we will learn to use the Azure Machine Learning Graphical Interface to process data, build, train, score, and evaluate a classification model to predict if a particular flight will be delayed by 15 minutes or more. To train the model, we will use Azure Machine Learning Compute resource. We will do all of this from the Azure Machine Learning designer without writing a single line of code.

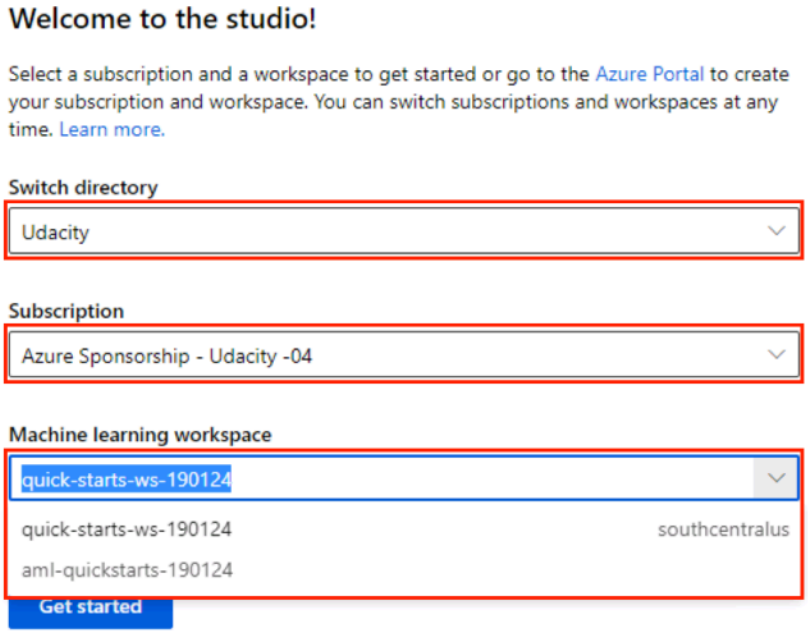
Exercise 1: Register Dataset with Azure Machine Learning studio

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.

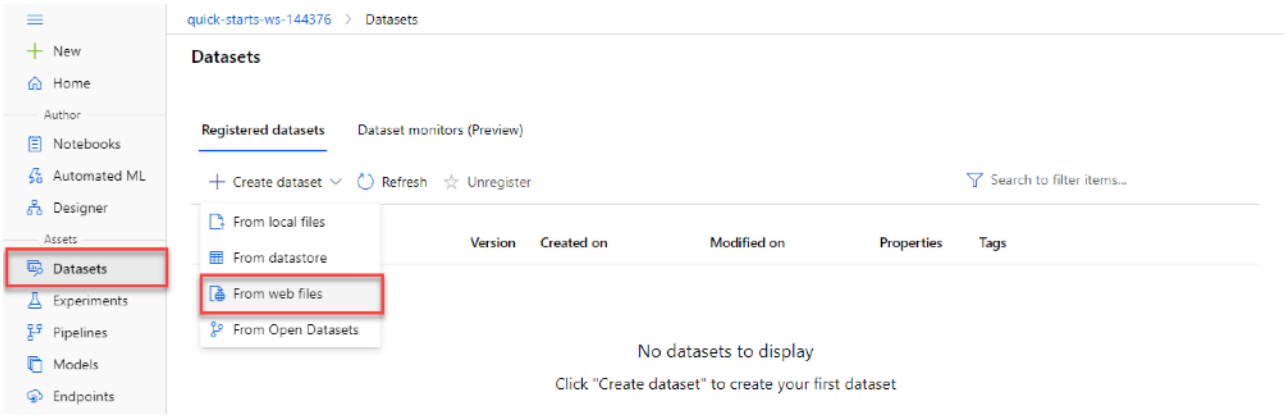


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



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**.

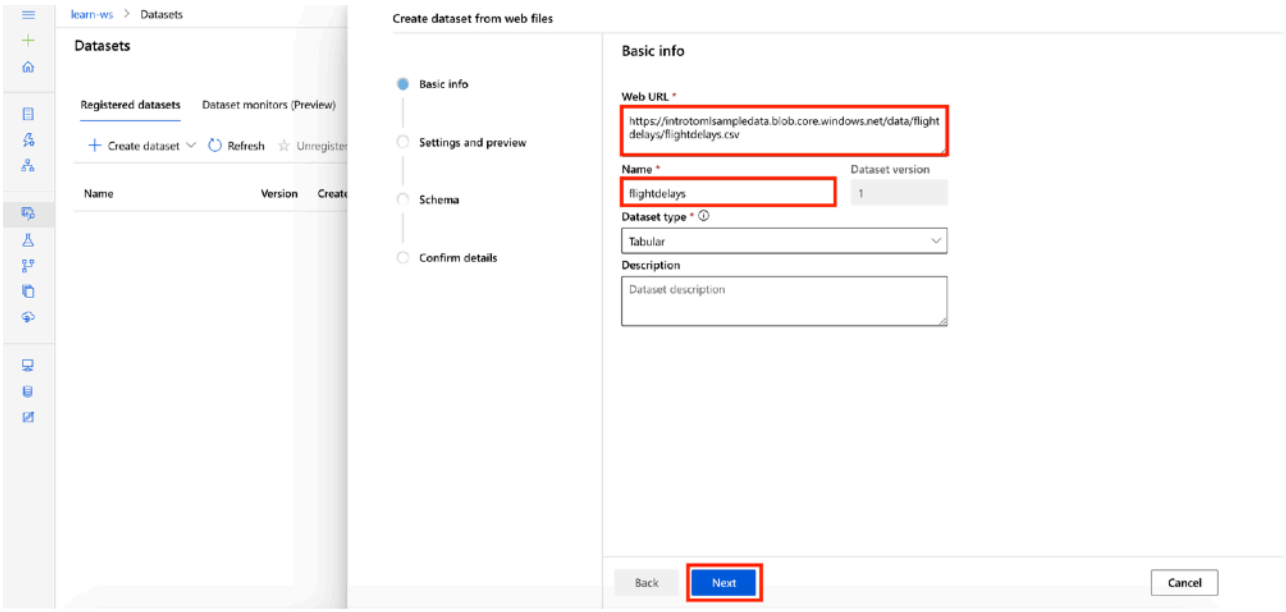
4. From the studio, select **Datasets, + Create dataset, From web files**. This will open the `Create dataset from web files` dialog on the right.



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

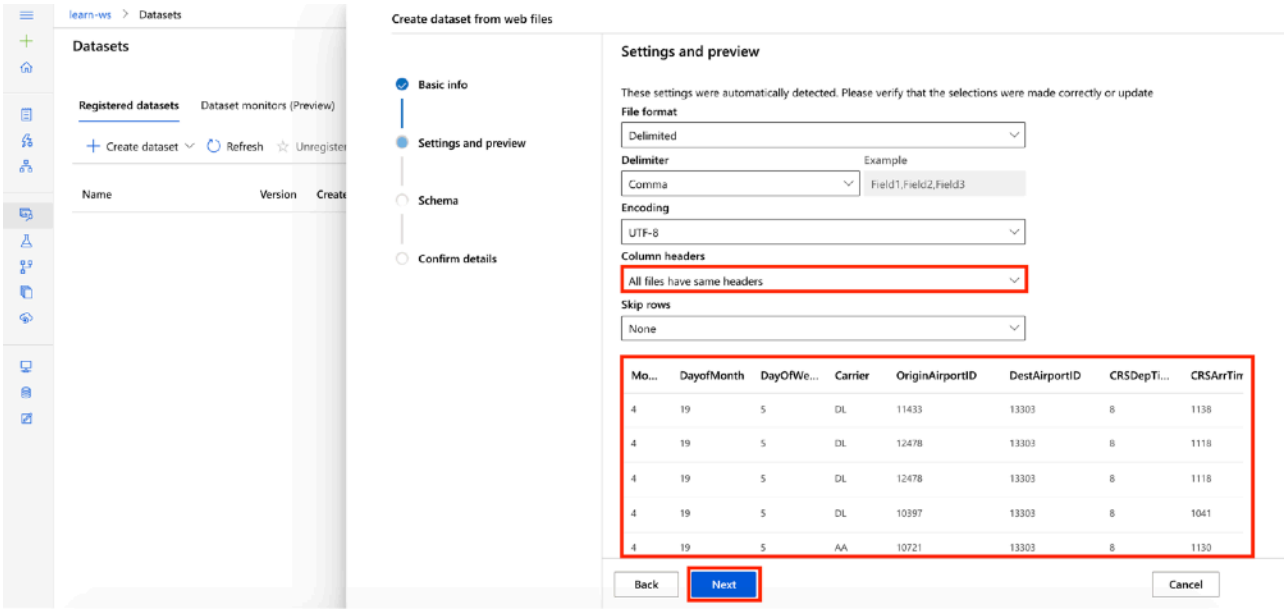
   `https://introtomlsampledata.blob.core.windows.net/data/flightdelays/flightdelays.csv`

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



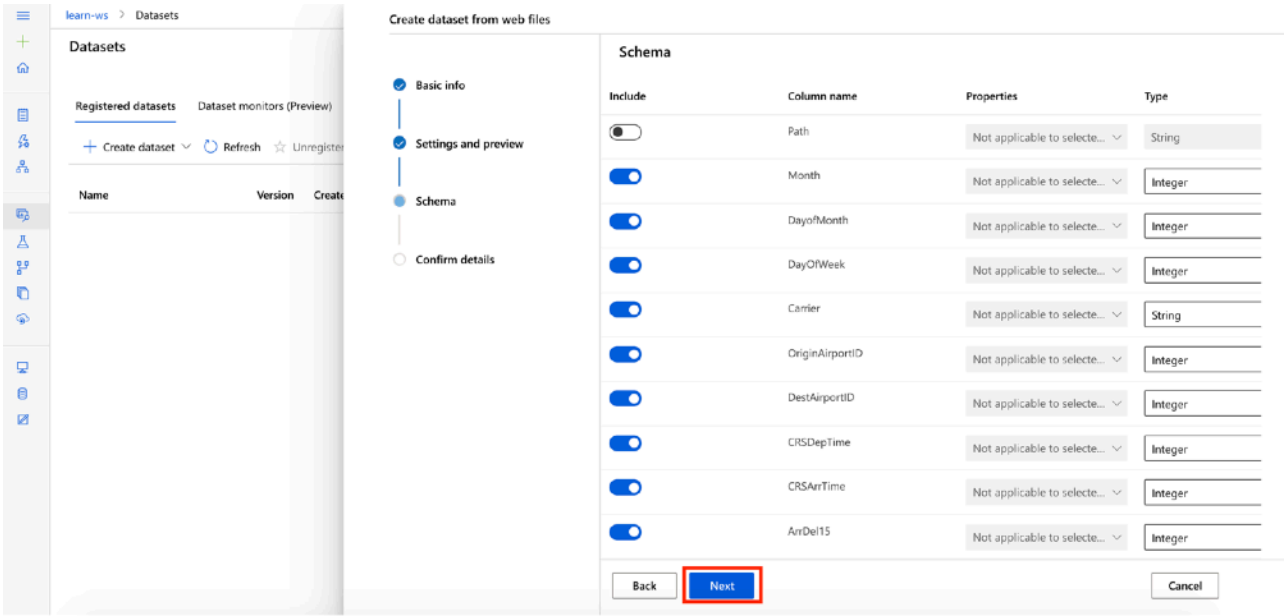
Task 2: Preview Dataset

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**



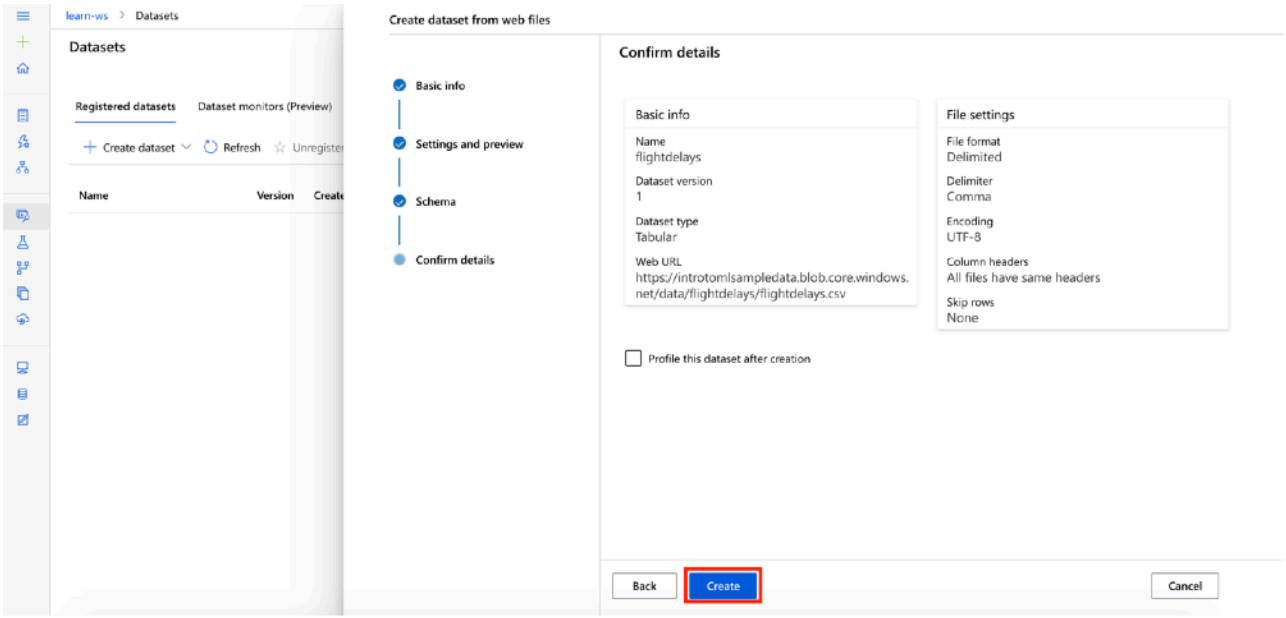
Task 3: Select Columns

1. Select columns from the dataset to include as part of your training data. Leave the default selections and select **Next**



Task 4: Create Dataset

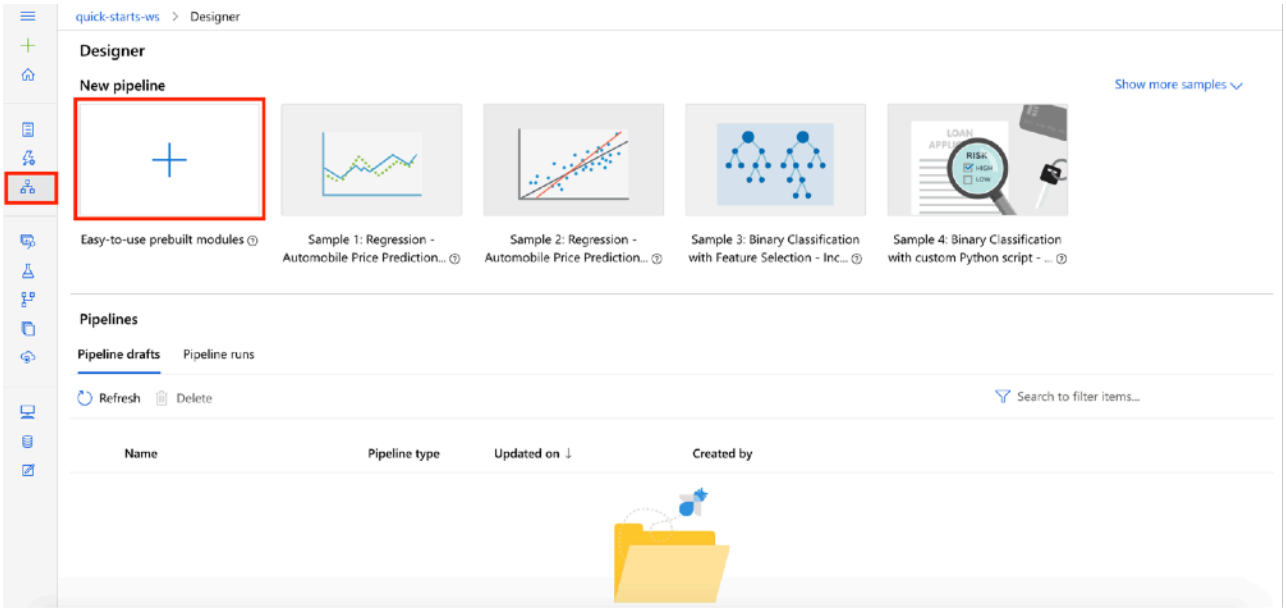
1. Confirm the dataset details and select **Create**



Exercise 2: Create New Training Pipeline

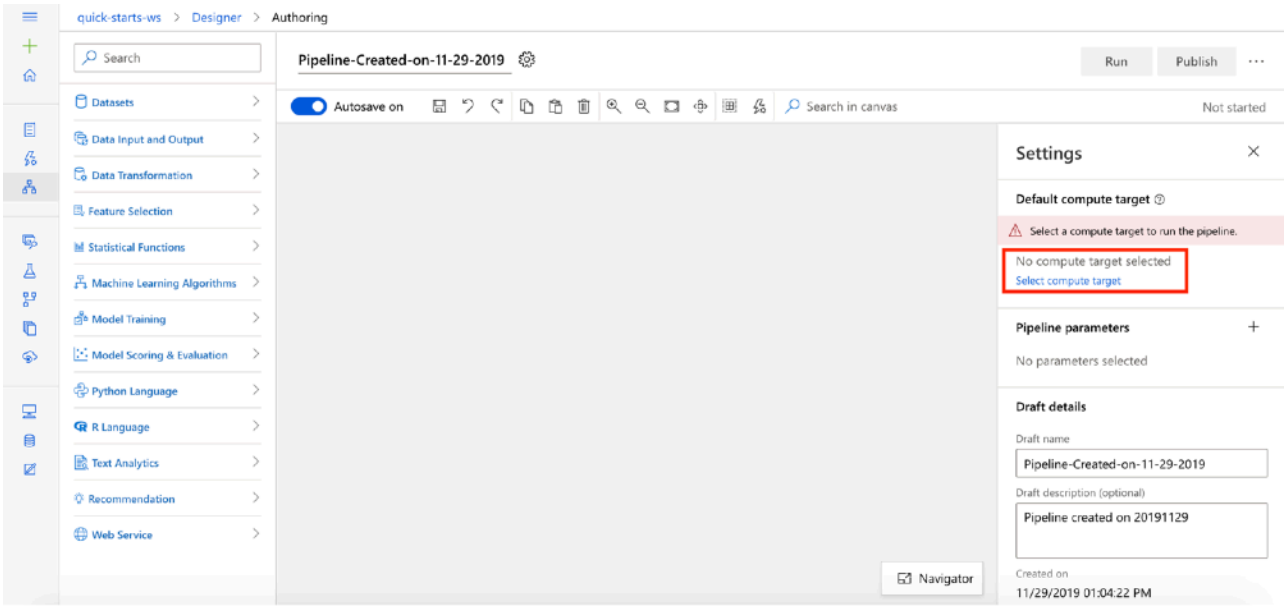
Task 1: Open Pipeline Authoring Editor

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



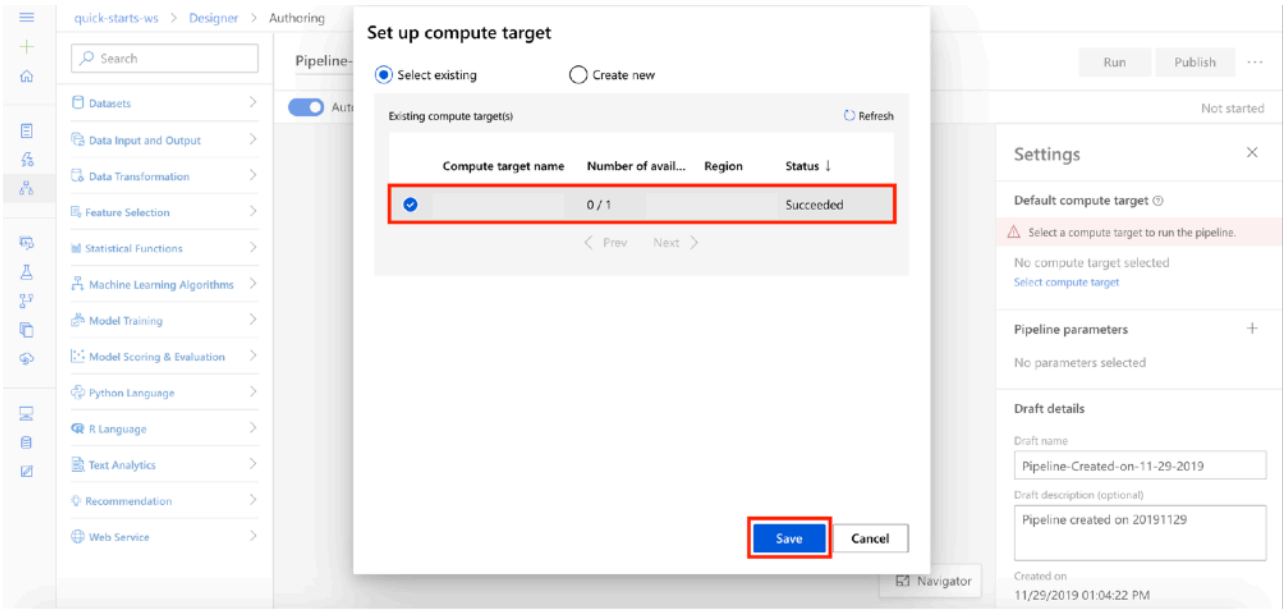
Task 2: Setup Compute Target

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



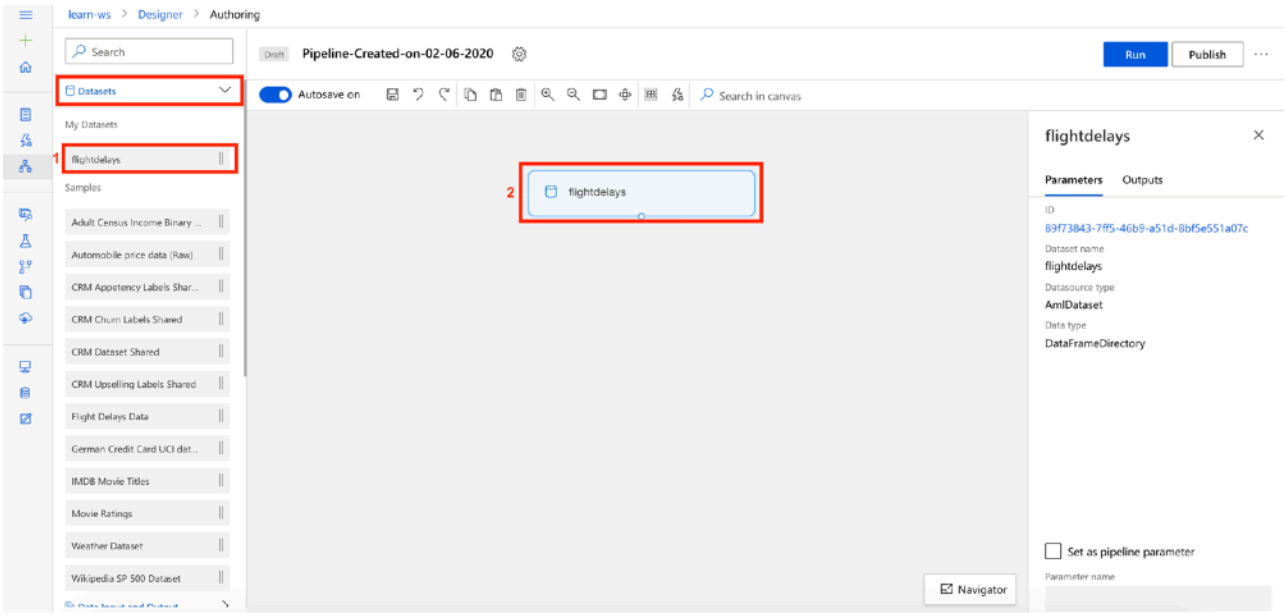
2. In the `Set up compute target` editor, select the available compute, and then select **Save**.

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.



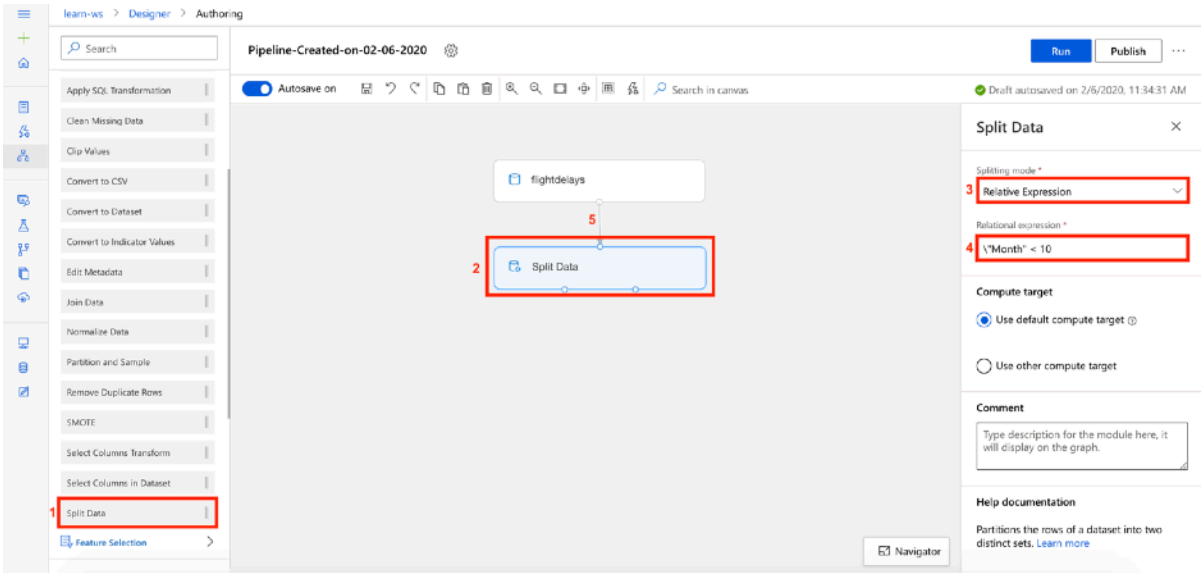
Task 3: Add Dataset

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



Task 4: Split Dataset

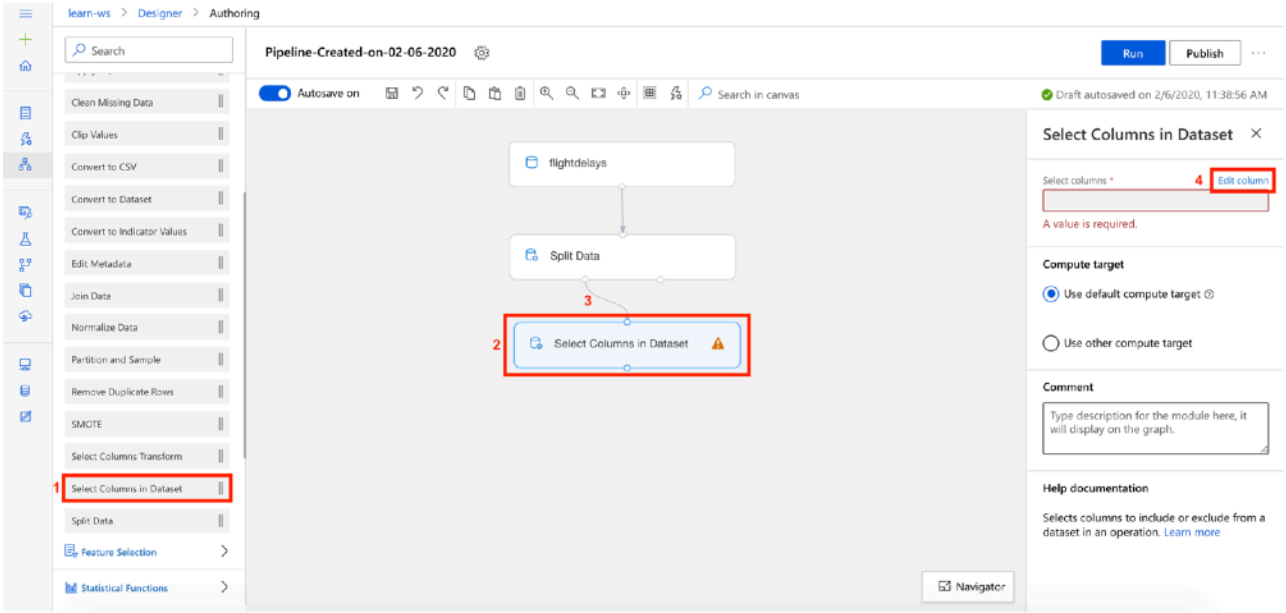
1. We will split the dataset such that months prior to October will be used for model training and months October to December will be used for model testing.
2. Select **Data Transformation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Split Data** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Splitting mode: **Relative Expression**
    4. Relational expression: **\"Month" < 10**
    5. Connect the `Dataset` to the `Split Data` module



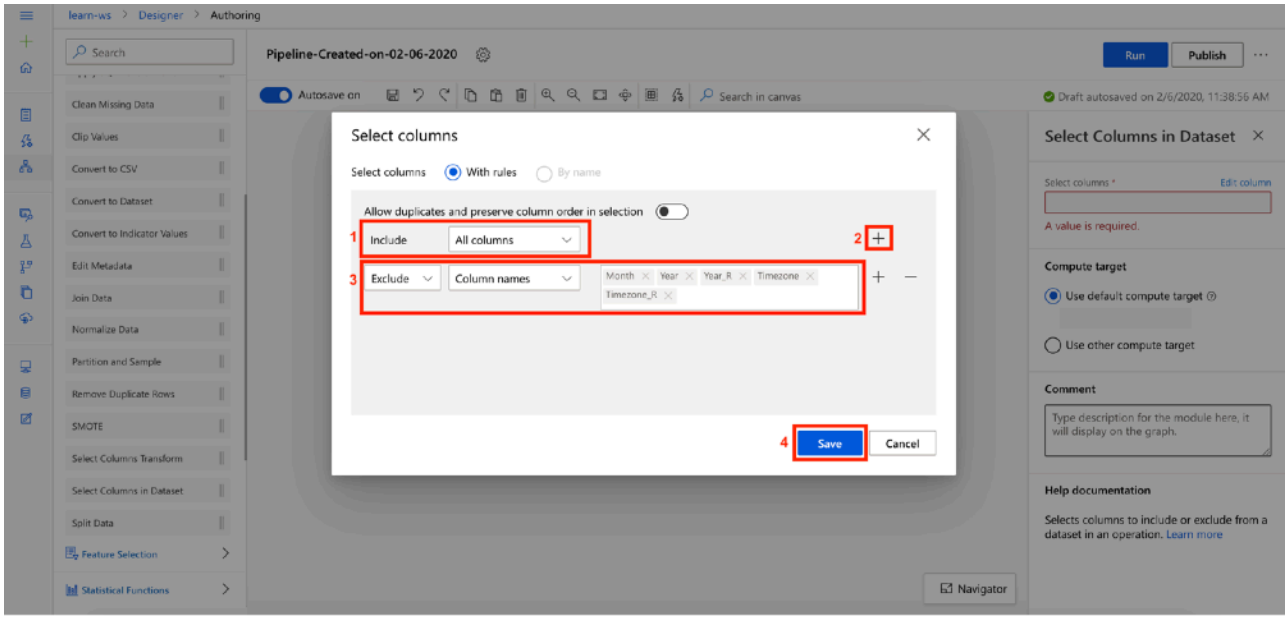
Note that you can submit the pipeline at any point to peek at the outputs and activities. Running pipeline also generates metadata that is available for downstream activities such selecting column names from a list in selection dialogs. Please refer ahead to Exercise 3, Task 1, Step 2 on details of submitting the pipeline. It can take up to 5-10 minutes to run the pipeline.

Task 5: Select Columns in Dataset

1. Select **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 first output of the `Split Data` module to the `Select Columns in Dataset` module
    4. Select **Edit column** link to open the Select columns` editor

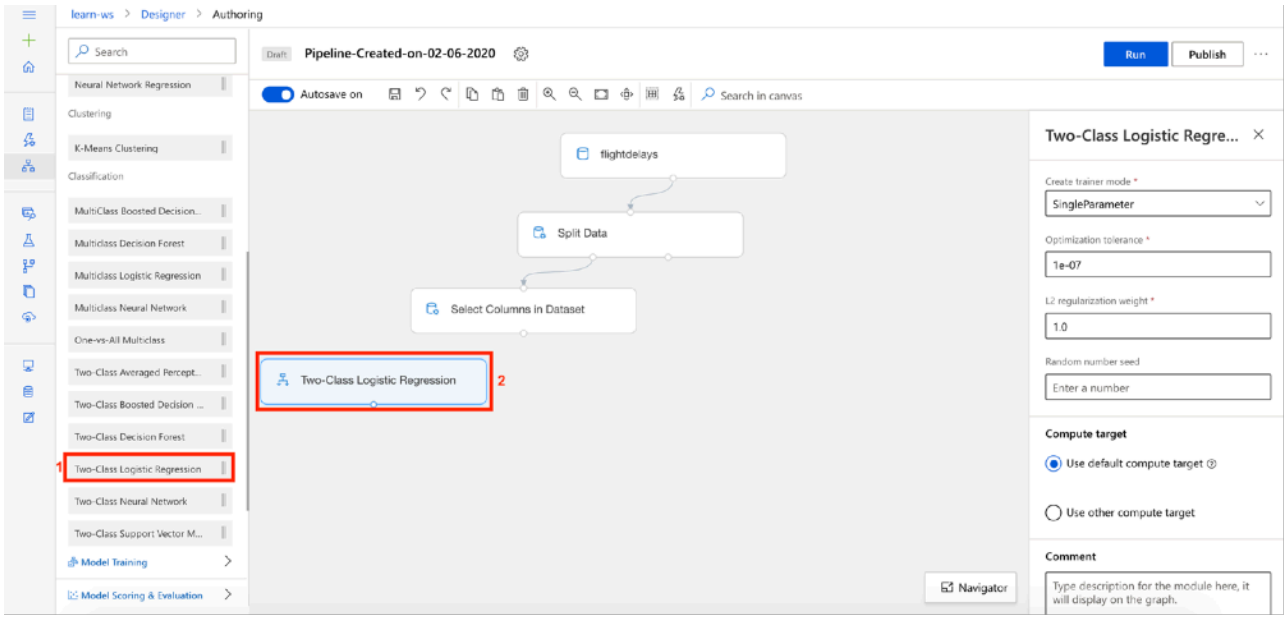


2. In the `Select columns` editor, follow the steps outlined below:
    1. Include: **All columns**
    2. Select **+**
    3. Exclude: **Column names**, provide the following column names to exclude: **Month, Year, Year_R, Timezone, Timezone_R**
    4. Select **Save**



Task 6: Initialize Classification Model

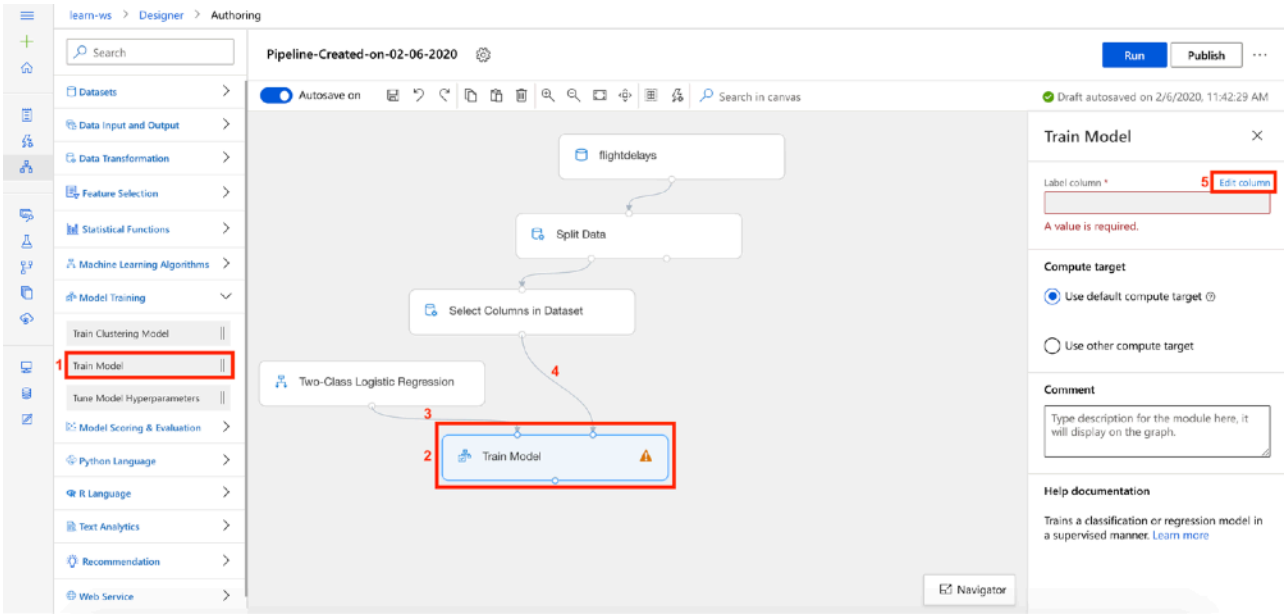
1. Select **Machine Learning Algorithms** section in the left navigation. Follow the steps outlined below:
    1. Select the **Two-Class Logistic Regression** prebuilt module
    2. Drag and drop the selected module on to the canvas



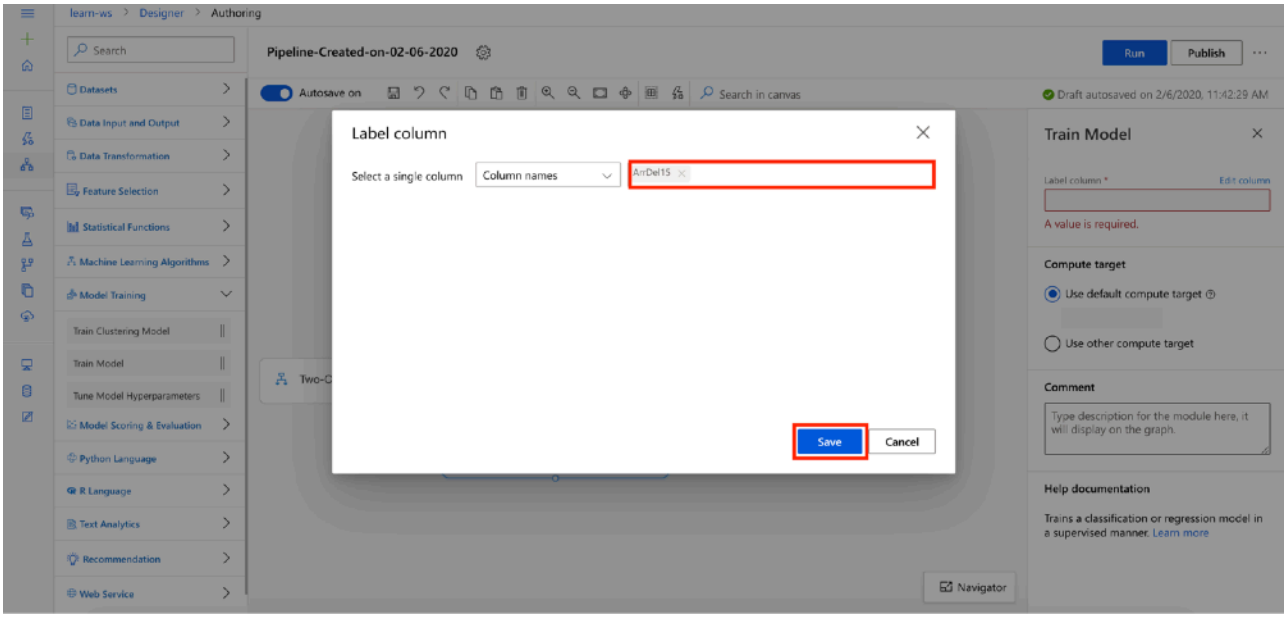
Task 7: Setup Train Model Module

1. Select **Model Training** section in the left navigation. Follow the steps outlined below:
    1. Select the **Train Model** 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 `Select Columns in Dataset` module to the second input of the `Train Model` module
5. Select the **Edit column** link to open the `Label column` editor



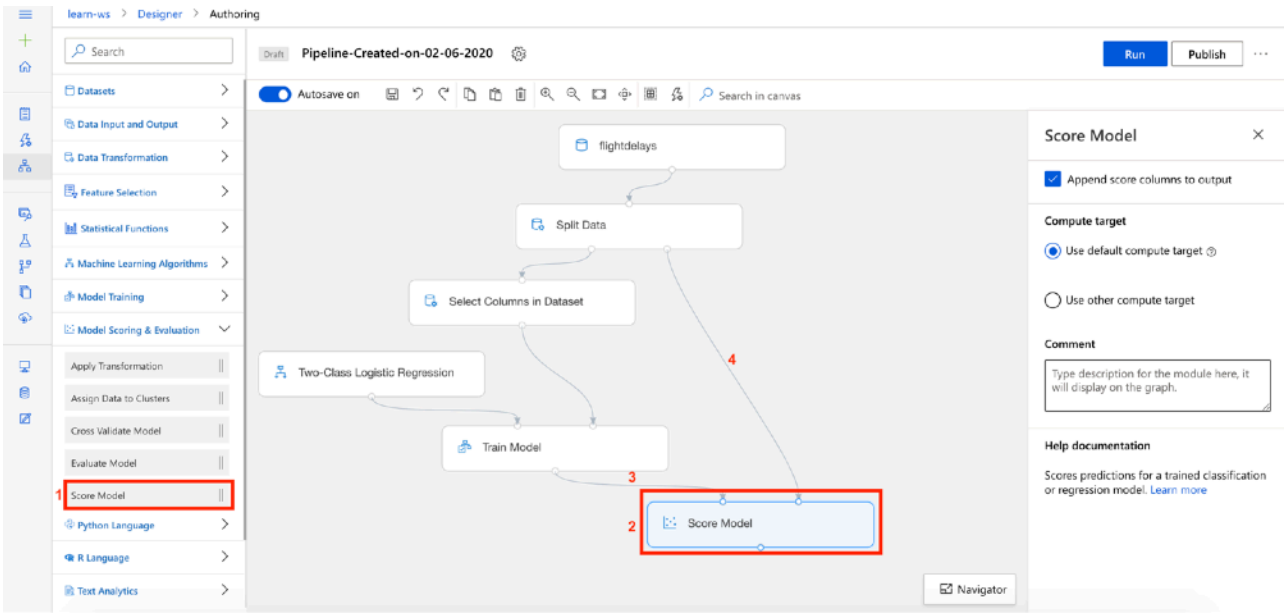
2. The `Label column` editor allows you to specify your `Label or Target column`. Type in the label column name **ArrDel15** and then select **Save**.



Task 8: Setup Score Model Module

1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Score Model** prebuilt module
    2. Drag and drop the selected module on to the canvas

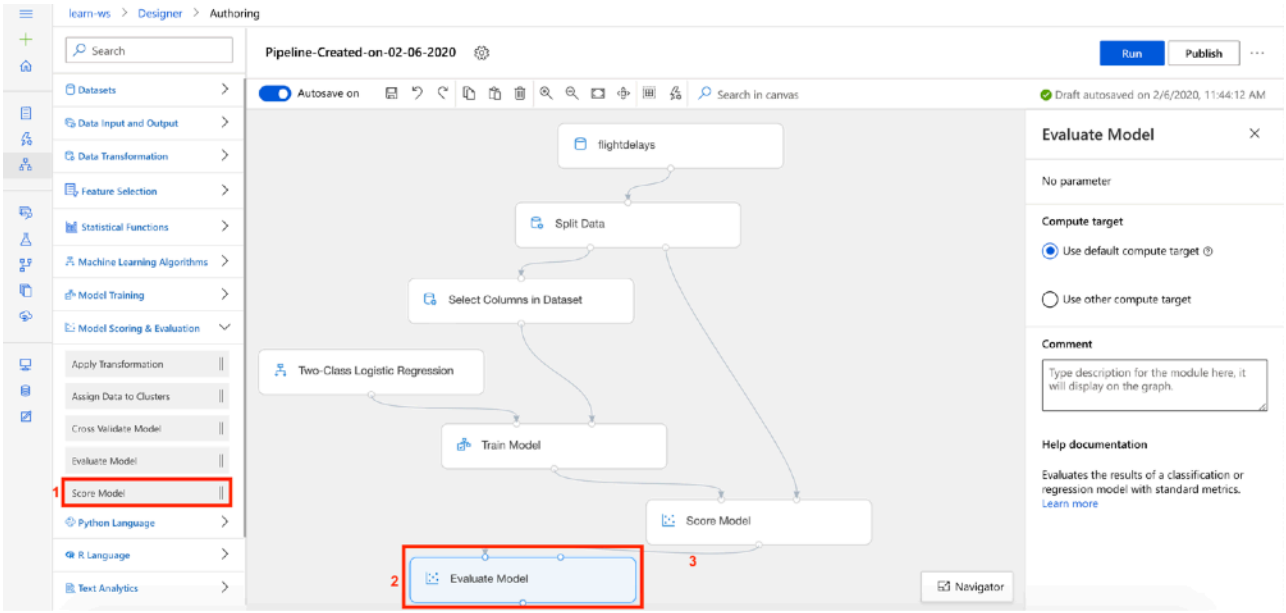
3. Connect the `Train Model` module to the first input of the `Score Model` module
4. Connect the second output of the `Split Data` module to the second input of the `Score Model` module



Note that *`Split Data`* module will feed data for both model training and model scoring.

Task 9: Setup Evaluate Model Module

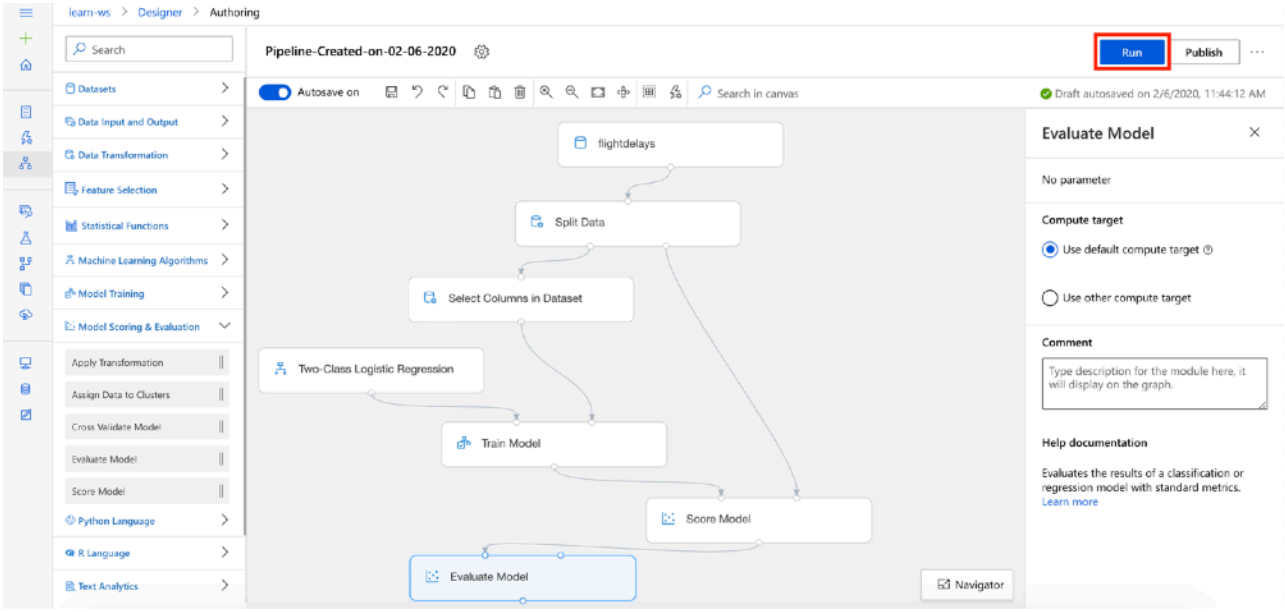
1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Evaluate Model** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the `Score Model` module to the first input of the `Evaluate Model` module



Exercise 3: Submit Training Pipeline

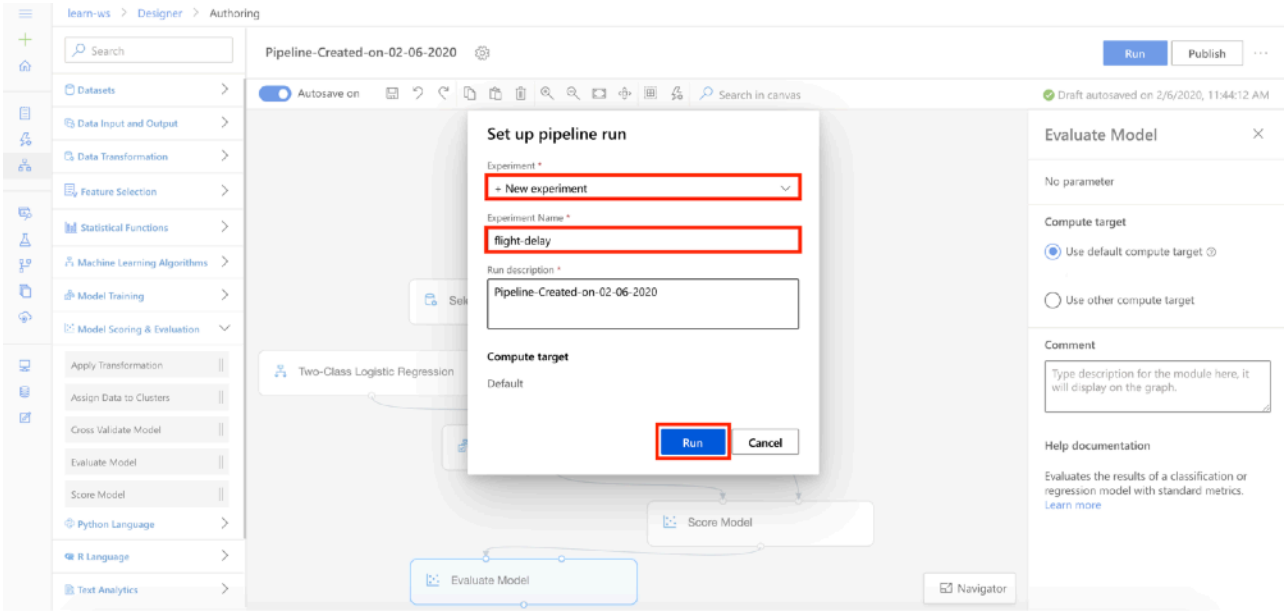
Task 1: Create Experiment and Submit Pipeline

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:` **flight-delay**, and then select **Submit**.

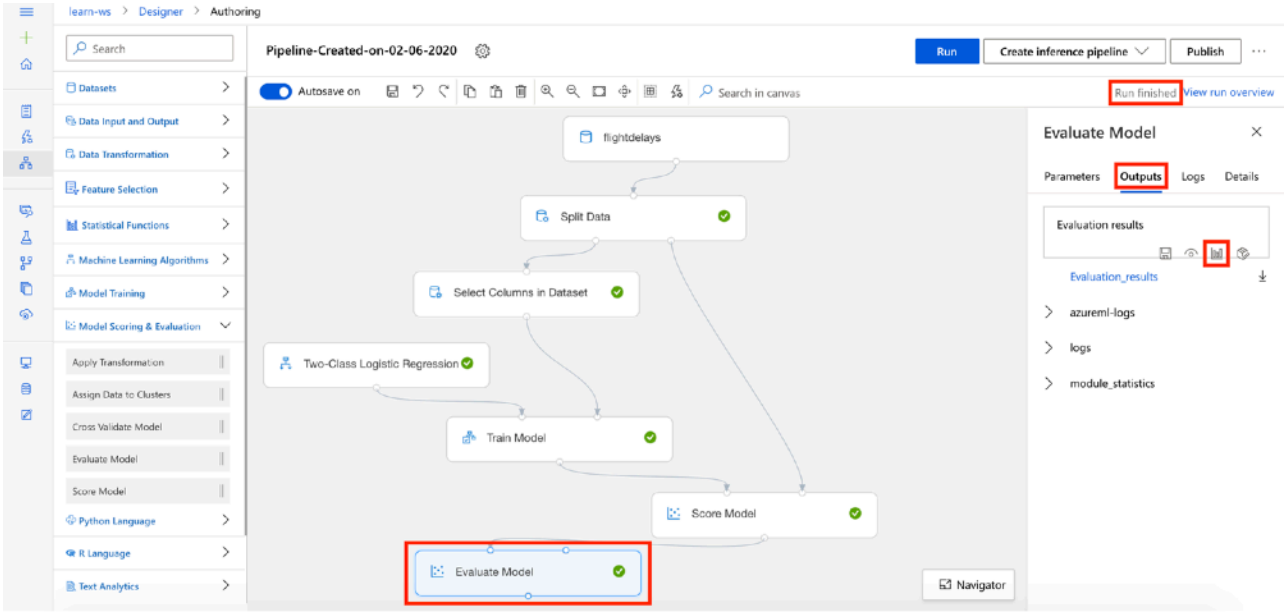


3. Wait for pipeline run to complete. It will take around **10 minutes** to complete the run.
4. While you wait for the model training to complete, you can learn more about the evaluation metrics for the classification algorithm used in this lab by selecting Metrics for classification models.

Exercise 4: Visualize the Evaluation Results

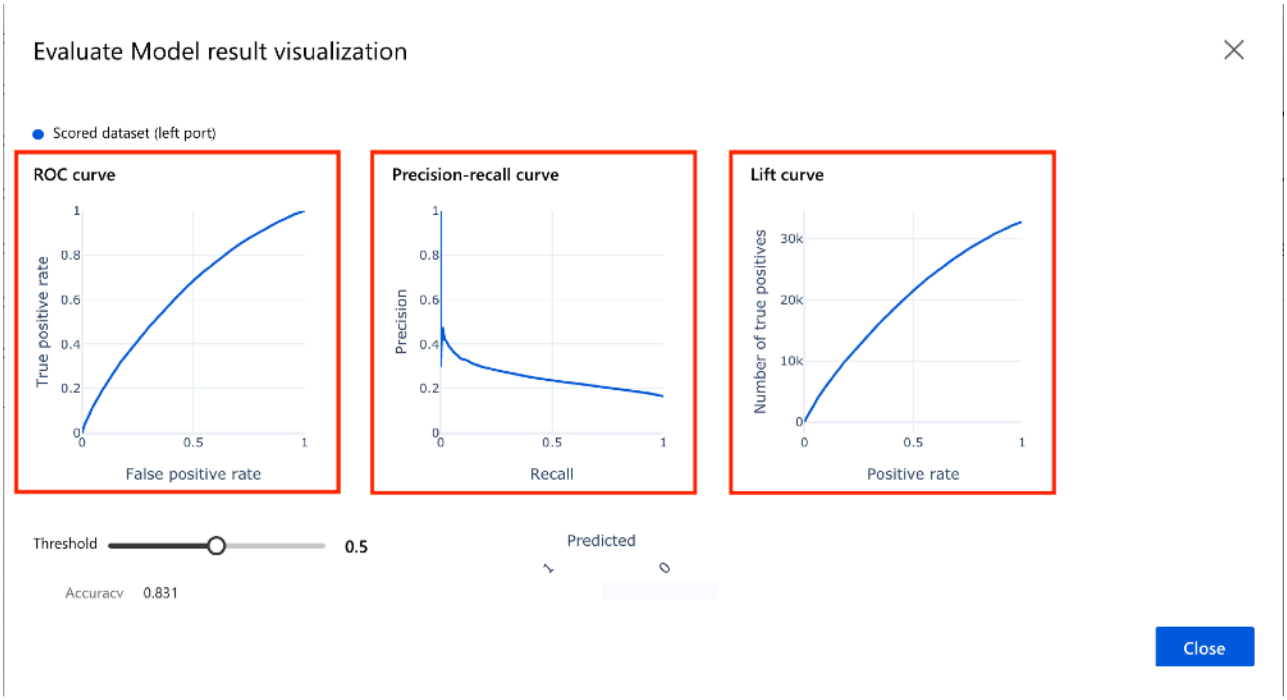
Task 1: Open the Result Visualization Dialog

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



Task 2: Evaluate Model Performance

1. Evaluate the model performance by reviewing the various evaluation curves, such as **ROC curve**, **Precision-recall curve**, and **Lift curve**.



2. Scroll down to review the following:
    1. Review the key metrics for classifiers: **Accuracy, Precision, Recall, F1 Score, and AUC**
    2. Review the binary classifier's **Confusion Matrix**

Next Steps

Congratulations! You have trained and evaluated your first classification machine learning model. 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.

Lab 6 - Train a Two-Class Boosted Decision Tree

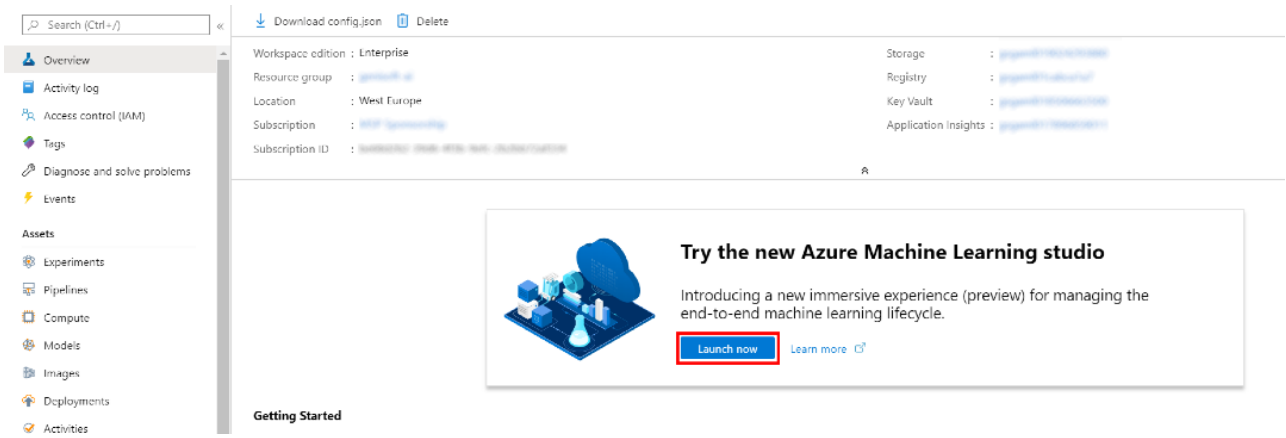
Overview

Azure Machine Learning designer (preview) gives you a cloud-based interactive, visual workspace that you can use to easily and quickly prep data, train and deploy machine learning models. It supports Azure Machine Learning compute, GPU or CPU. Machine Learning designer also supports publishing models as web services on Azure Kubernetes Service that can easily be consumed by other applications.
In this lab, we will be using the `Flight Delays` data set that is enhanced with the weather data. Based on the enriched dataset, we will learn to use the Azure Machine Learning Graphical Interface to process data, build, train, score, and evaluate a classification model to predict if a particular flight will be delayed by 15 minutes or more. The classification algorithm used in this lab will be the ensemble algorithm: **Two-Class Boosted Decision Tree**. To train the model, we will use Azure Machine Learning Compute resource. We will do all of this from the Azure Machine Learning designer without writing a single line of code.

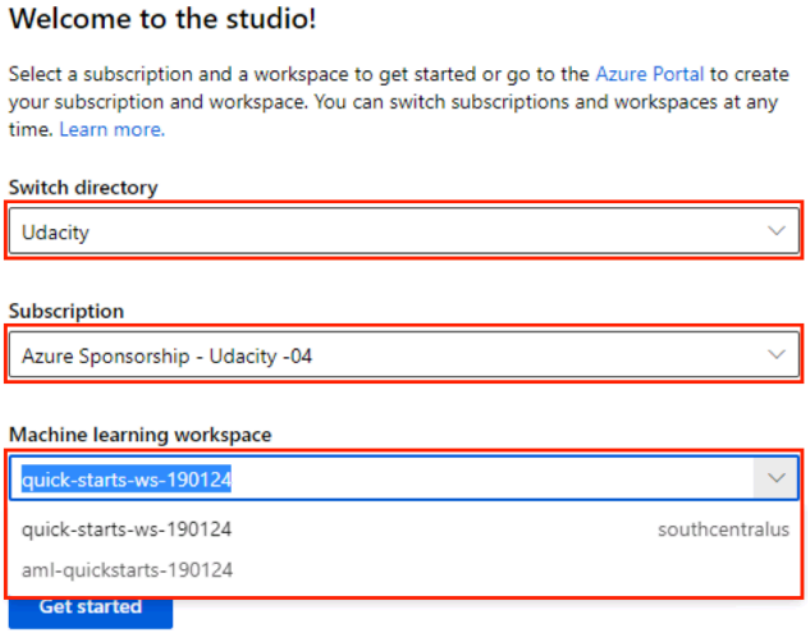
Exercise 1: Register Dataset with Azure Machine Learning studio

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.

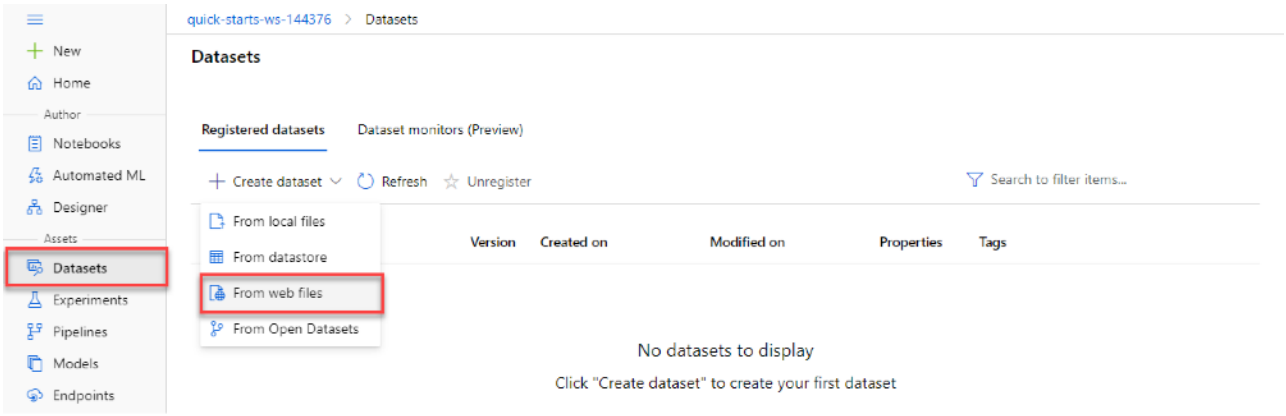


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



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**.

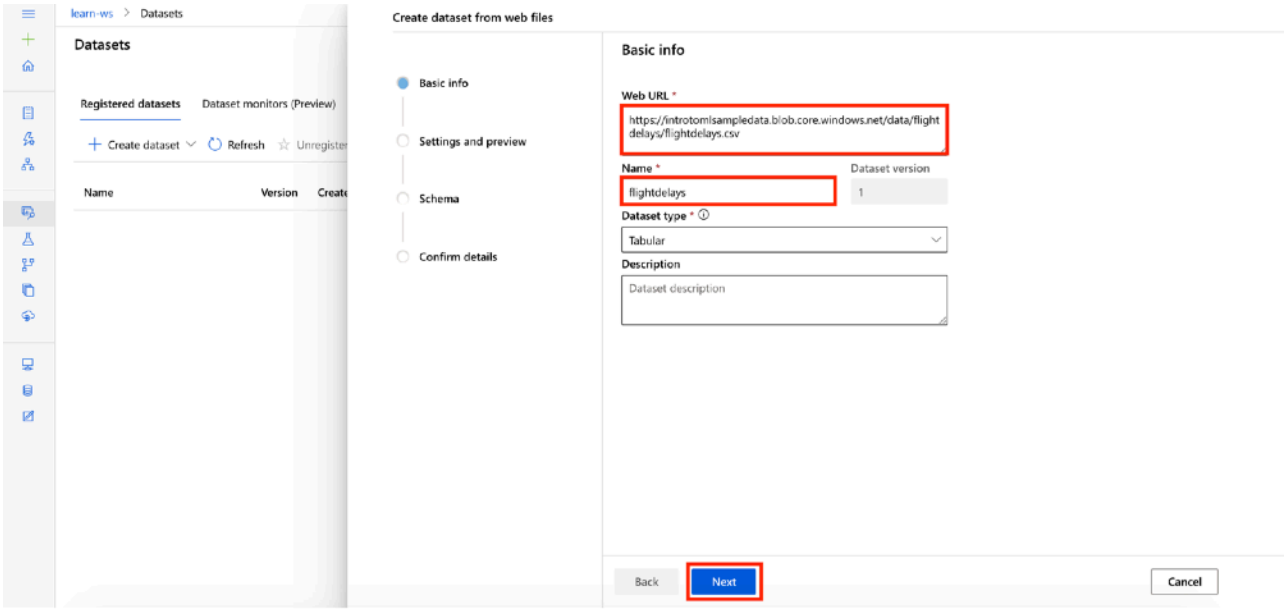
4. From the studio, select **Datasets, + Create dataset, From web files**. This will open the `Create dataset from web files` dialog on the right.



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

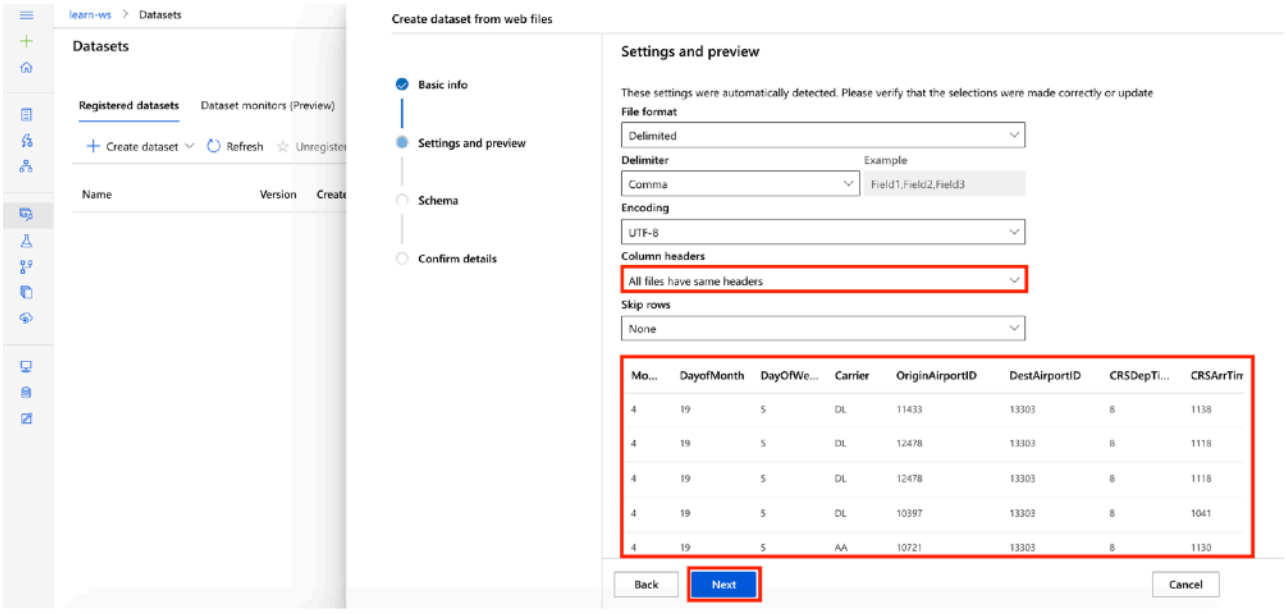
   `https://introtomlsampledata.blob.core.windows.net/data/flightdelays/flightdelays.csv`

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



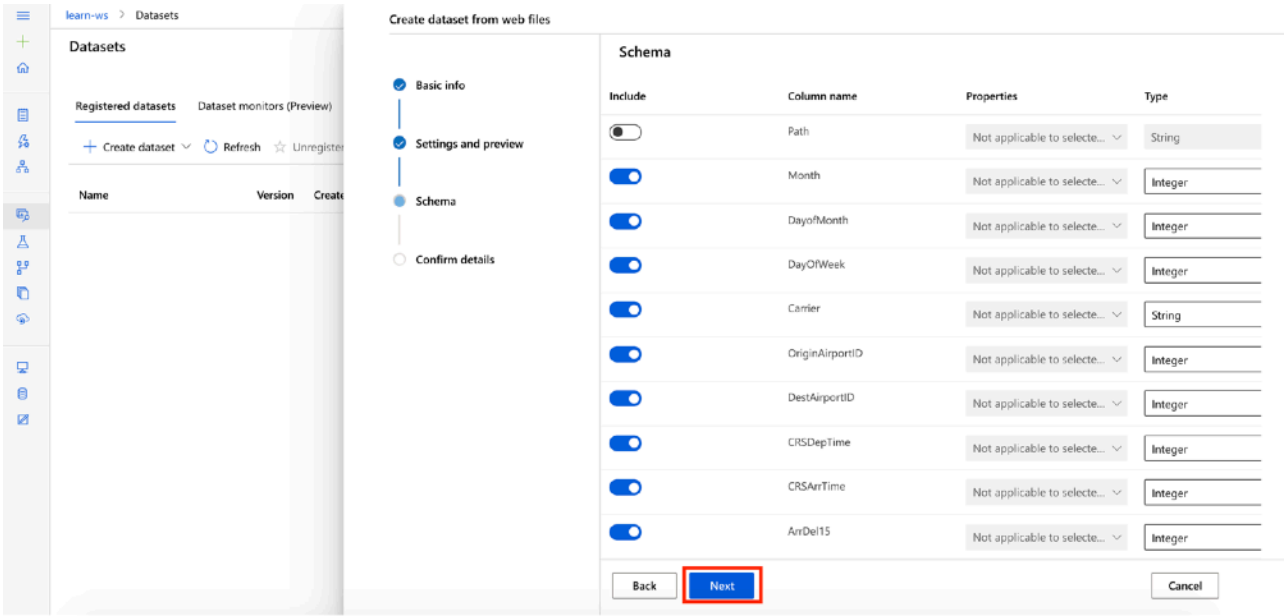
Task 2: Preview Dataset

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**



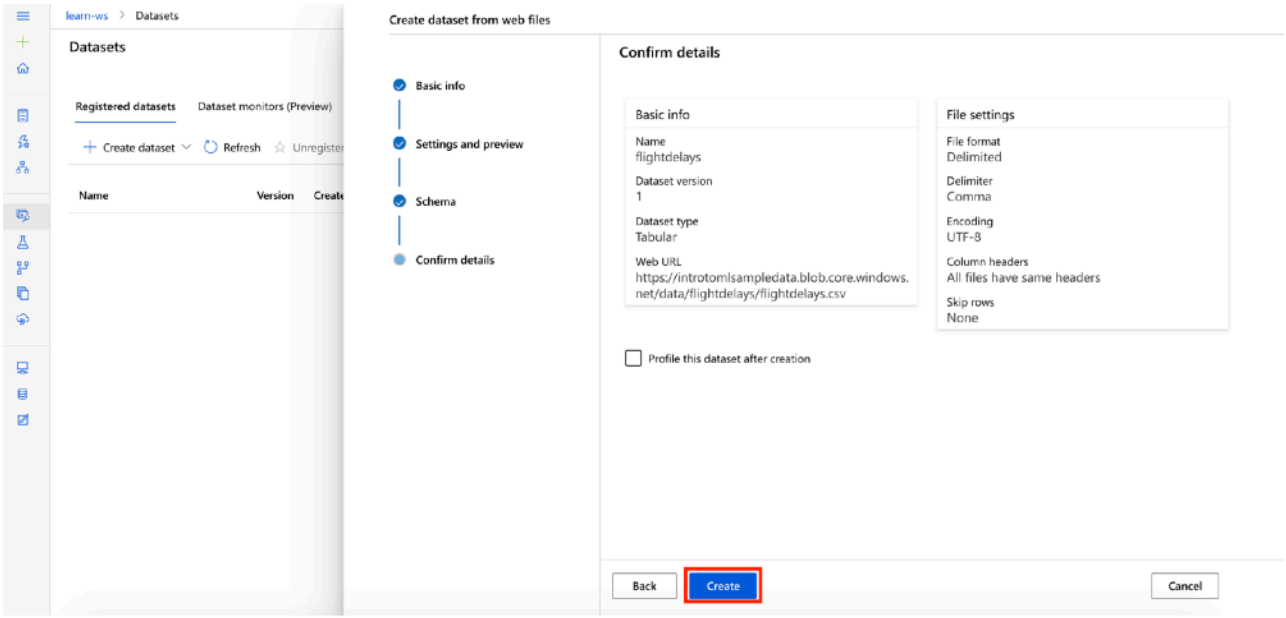
Task 3: Select Columns

1. Select columns from the dataset to include as part of your training data. Leave the default selections and select **Next**



Task 4: Create Dataset

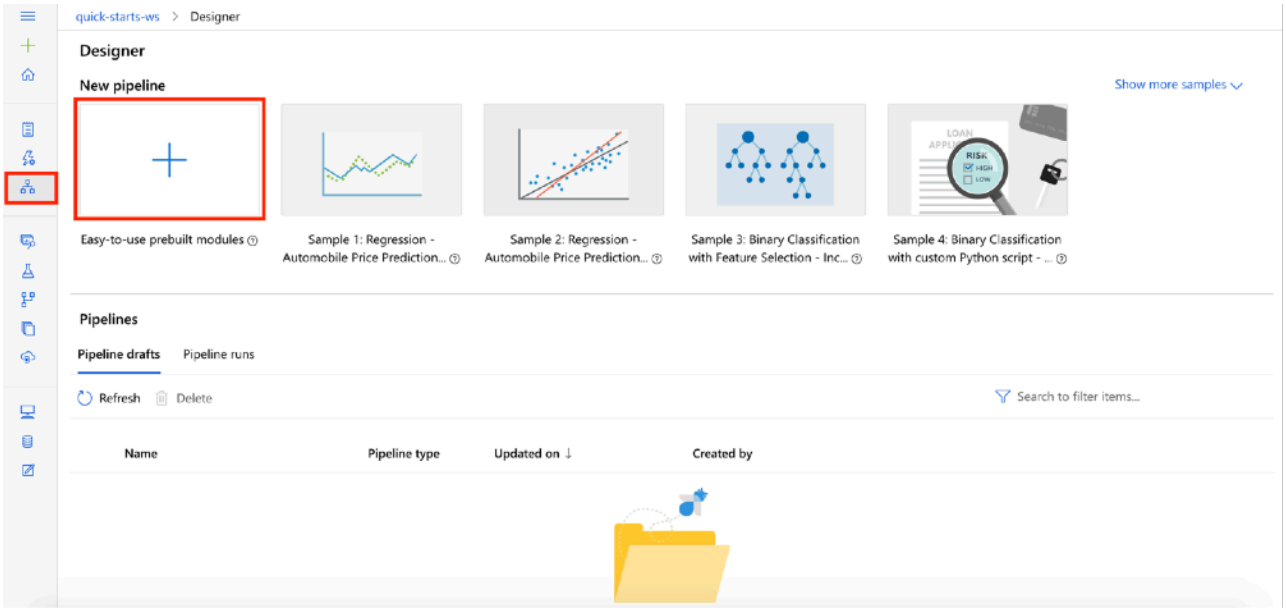
1. Confirm the dataset details and select **Create**



Exercise 2: Create New Training Pipeline

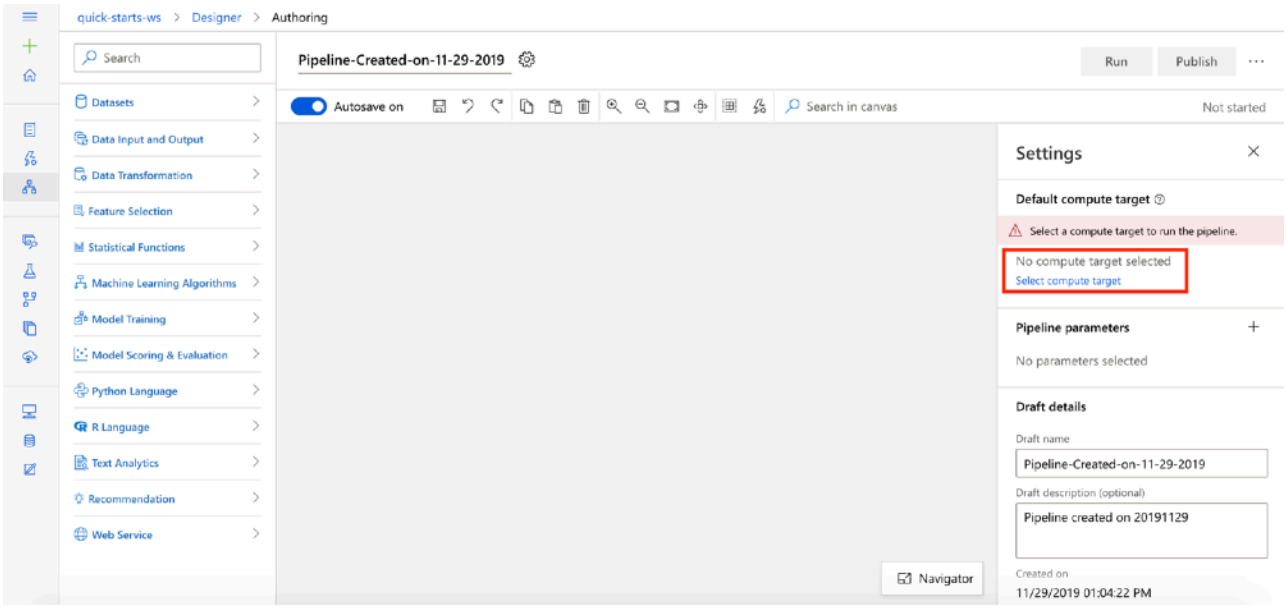
Task 1: Open Pipeline Authoring Editor

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



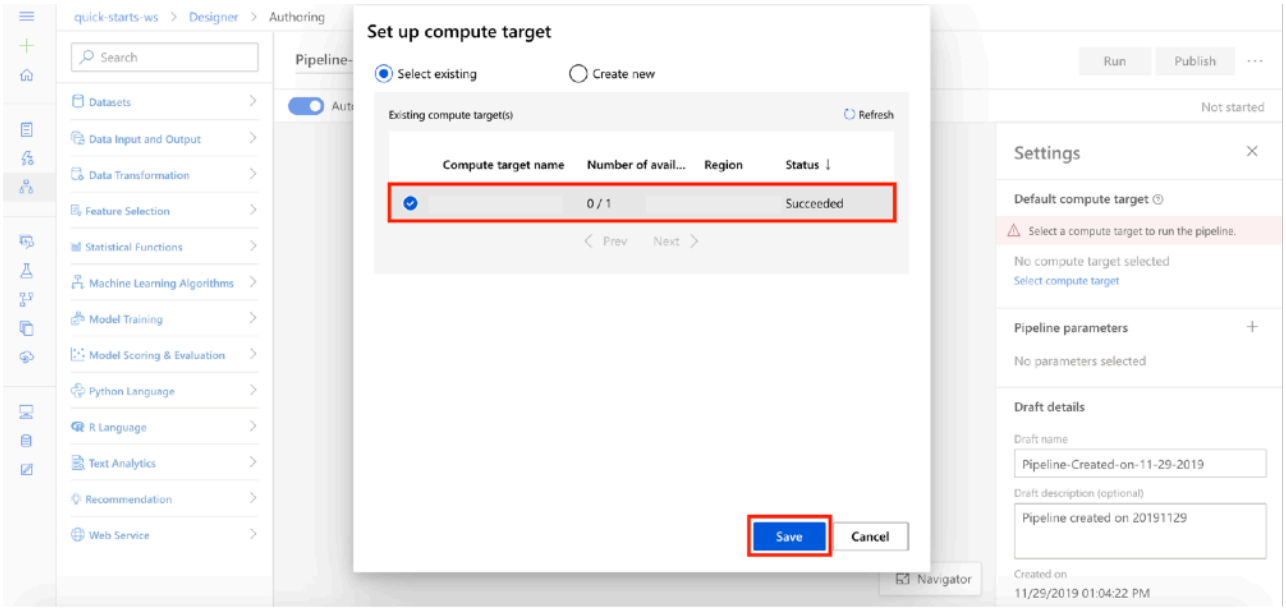
Task 2: Setup Compute Target

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



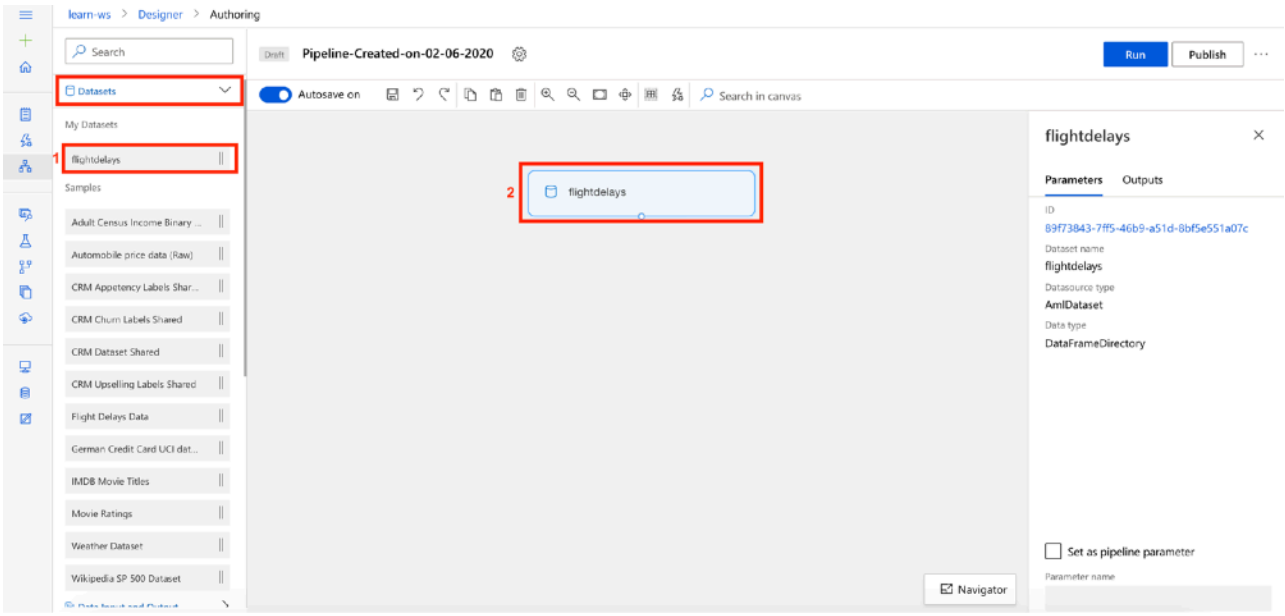
2. In the `Set up compute target` editor, select the available compute, and then select **Save**.

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.



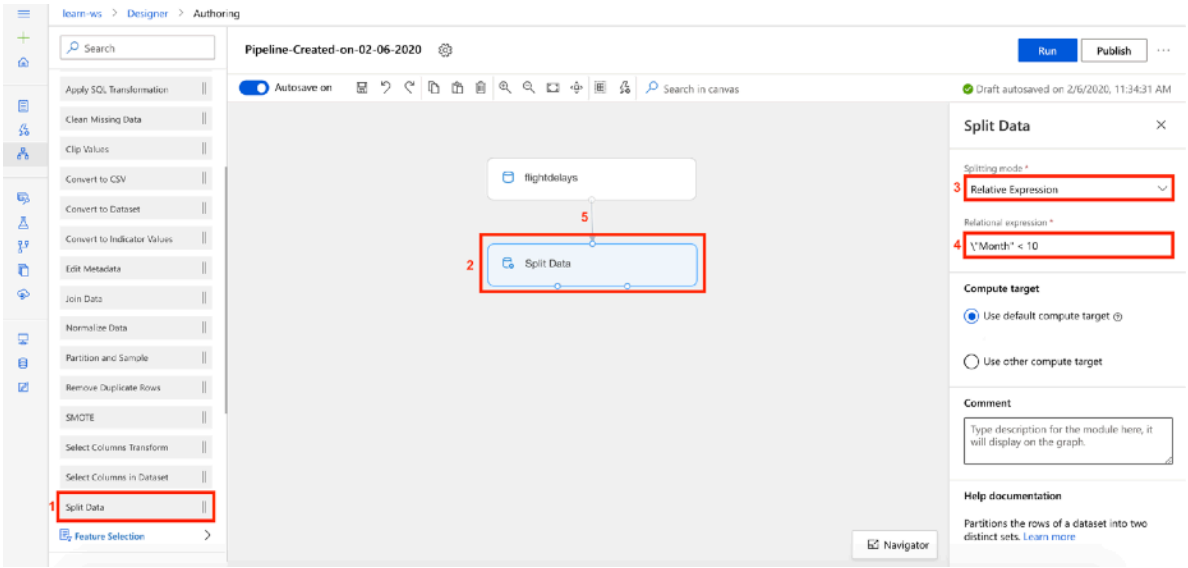
Task 3: Add Dataset

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



Task 4: Split Dataset

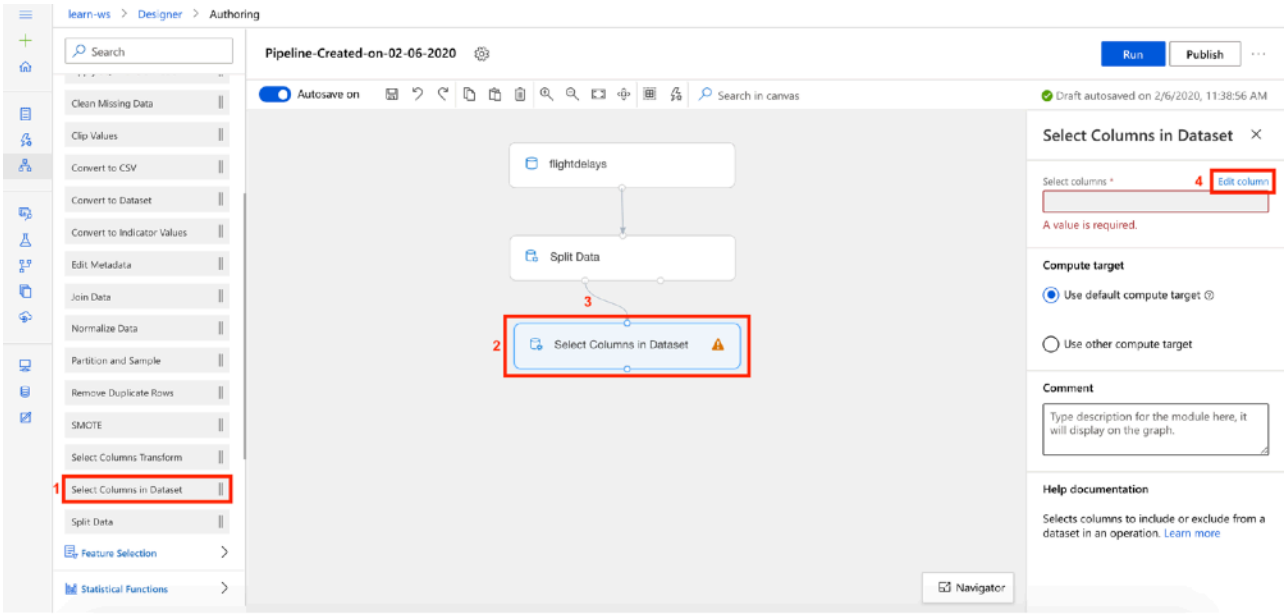
1. We will split the dataset such that months prior to October will be used for model training and months October to December will be used for model testing.
2. Select **Data Transformation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Split Data** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Splitting mode: **Relative Expression**
    4. Relational expression: **\"Month" < 10**
    5. Connect the `Dataset` to the `Split Data` module



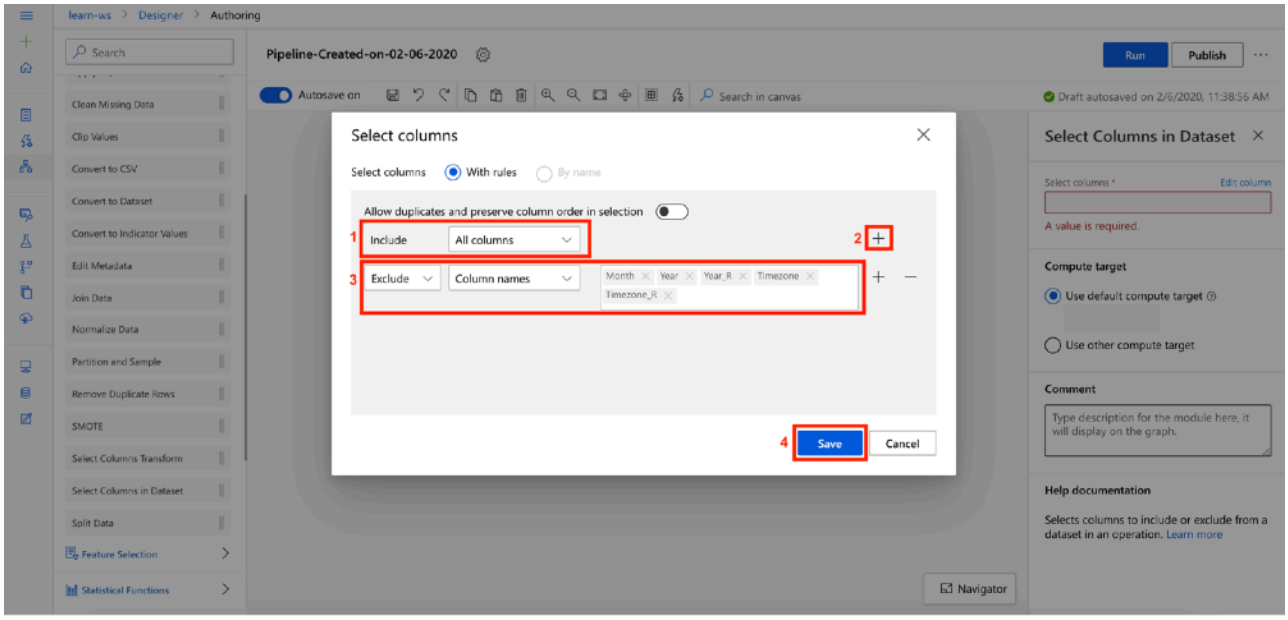
Note that you can submit the pipeline at any point to peek at the outputs and activities. Running pipeline also generates metadata that is available for downstream activities such selecting column names from a list in selection dialogs.

Task 5: Select Columns in Dataset

1. Select **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 first output of the `Split Data` module to the `Select Columns in Dataset` module
    4. Select **Edit column** link to open the Select columns` editor

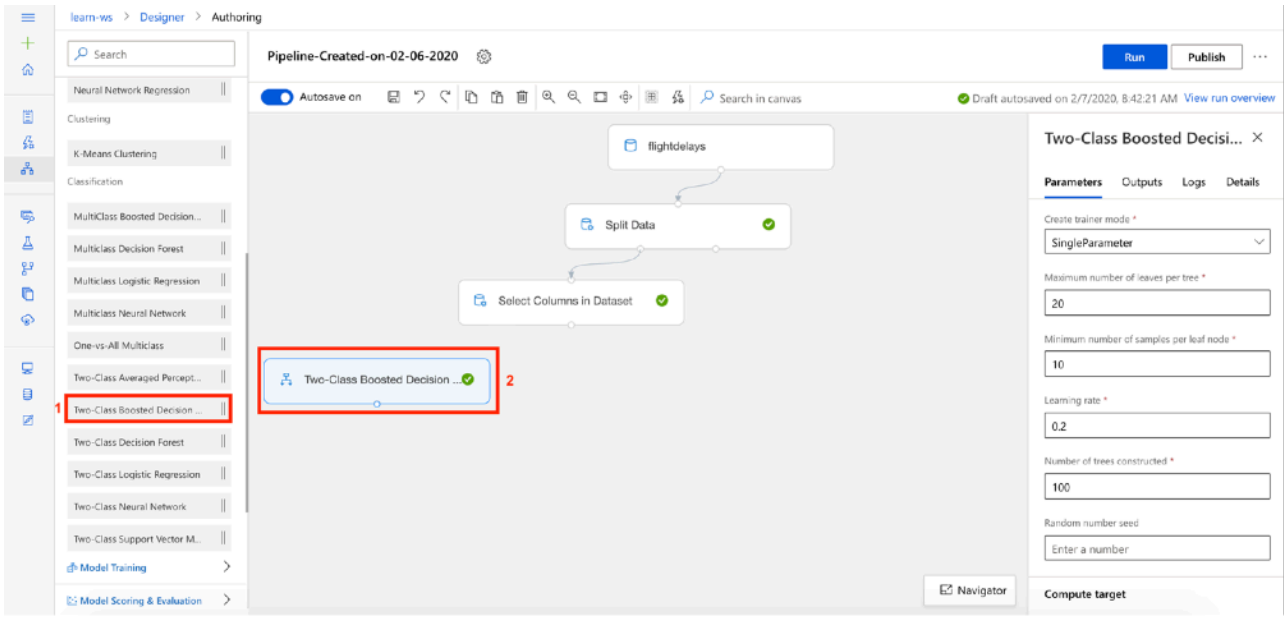


2. In the `Select columns` editor, follow the steps outlined below:
    1. Include: **All columns**
    2. Select **+**
    3. Exclude: **Column names**, provide the following column names to exclude: **Month, Year, Year_R, Timezone, Timezone_R**
    4. Select **Save**



Task 6: Initialize Classification Model

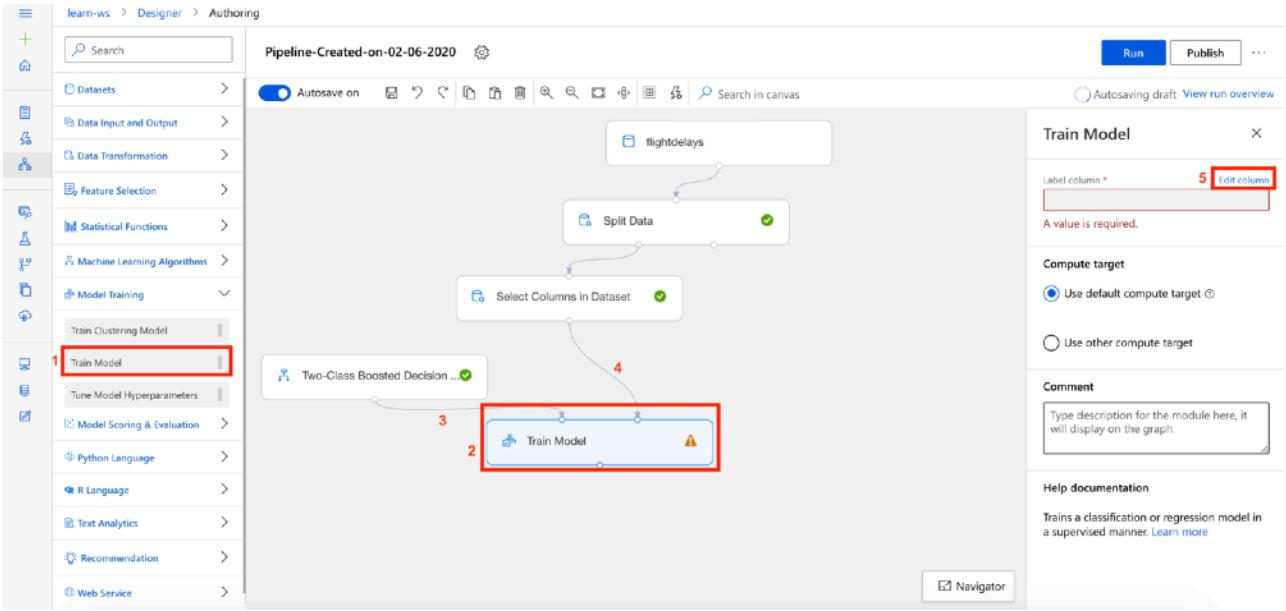
1. Select **Machine Learning Algorithms** section in the left navigation. Follow the steps outlined below:
    1. Select the **Two-Class Boosted Decision Tree** prebuilt module
    2. Drag and drop the selected module on to the canvas



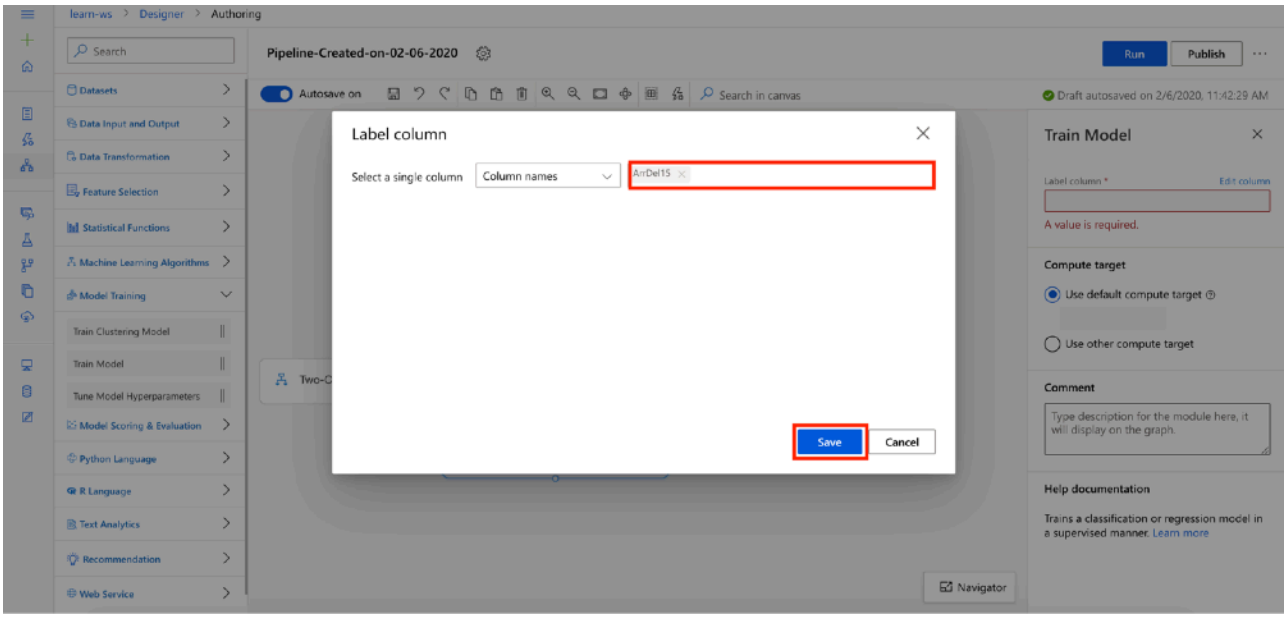
Task 7: Setup Train Model Module

1. Select **Model Training** section in the left navigation. Follow the steps outlined below:
    1. Select the **Train Model** prebuilt module
    2. Drag and drop the selected module on to the canvas

3. Connect the `Two-Class Boosted Decision Tree` module to the first input of the `Train Model` module
4. Connect the `Select Columns in Dataset` module to the second input of the `Train Model` module
5. Select the **Edit column** link to open the `Label column` editor



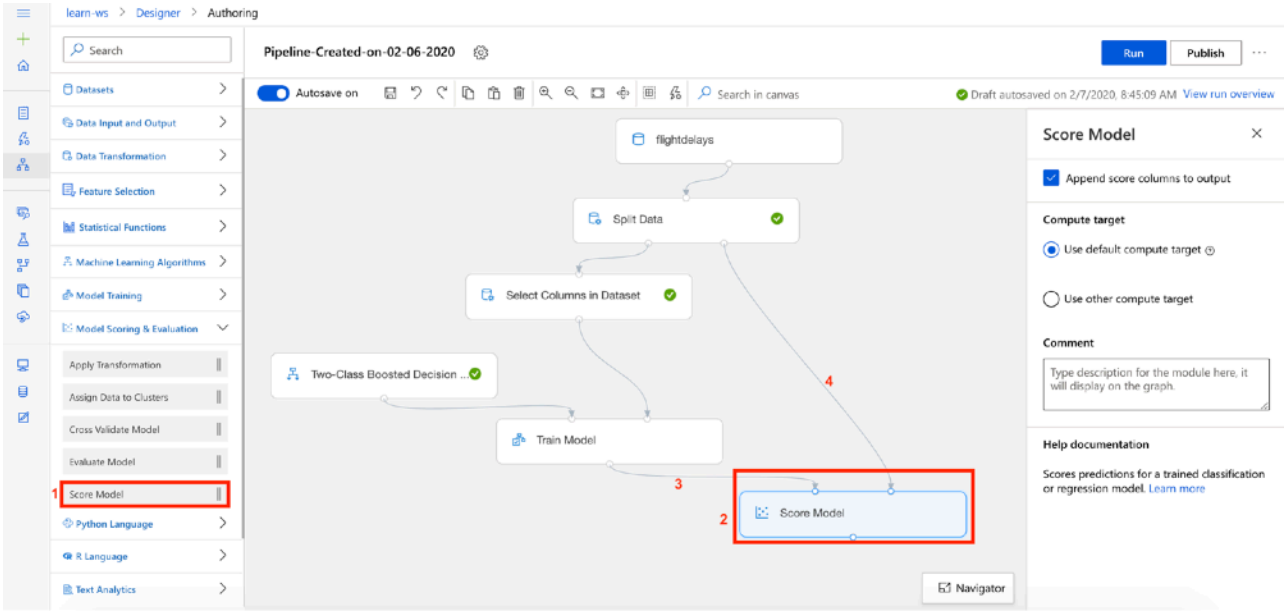
2. The `Label column` editor allows you to specify your `Label or Target column`. Type in the label column name **ArrDel15** and then select **Save**.



Task 8: Setup Score Model Module

1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Score Model** prebuilt module
    2. Drag and drop the selected module on to the canvas

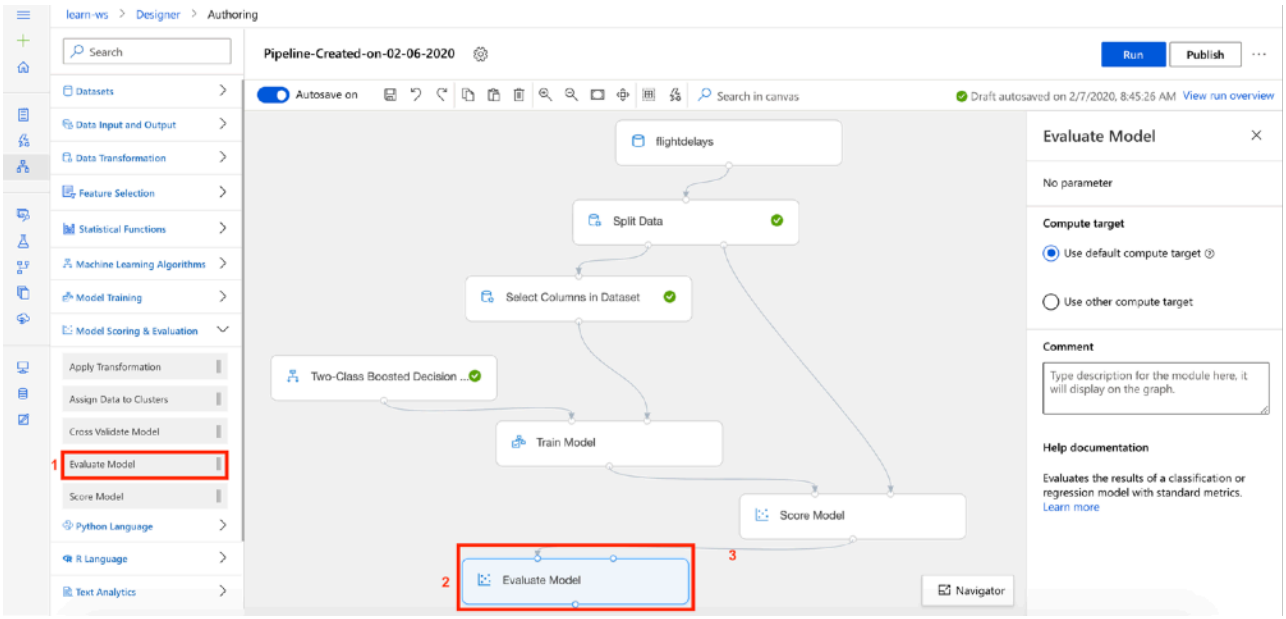
3. Connect the `Train Model` module to the first input of the `Score Model` module
4. Connect the second output of the `Split Data` module to the second input of the `Score Model` module



Note that *`Split Data`* module will feed data for both model training and model scoring.

Task 9: Setup Evaluate Model Module

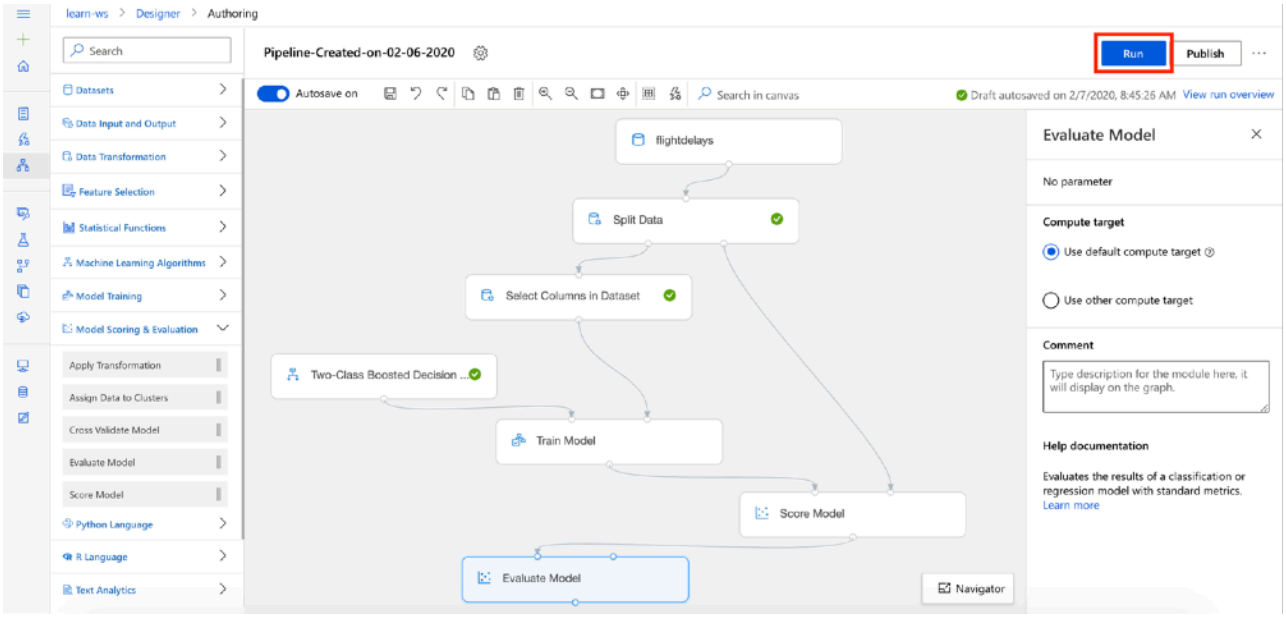
1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Evaluate Model** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the `Score Model` module to the first input of the `Evaluate Model` module



Exercise 3: Submit Training Pipeline

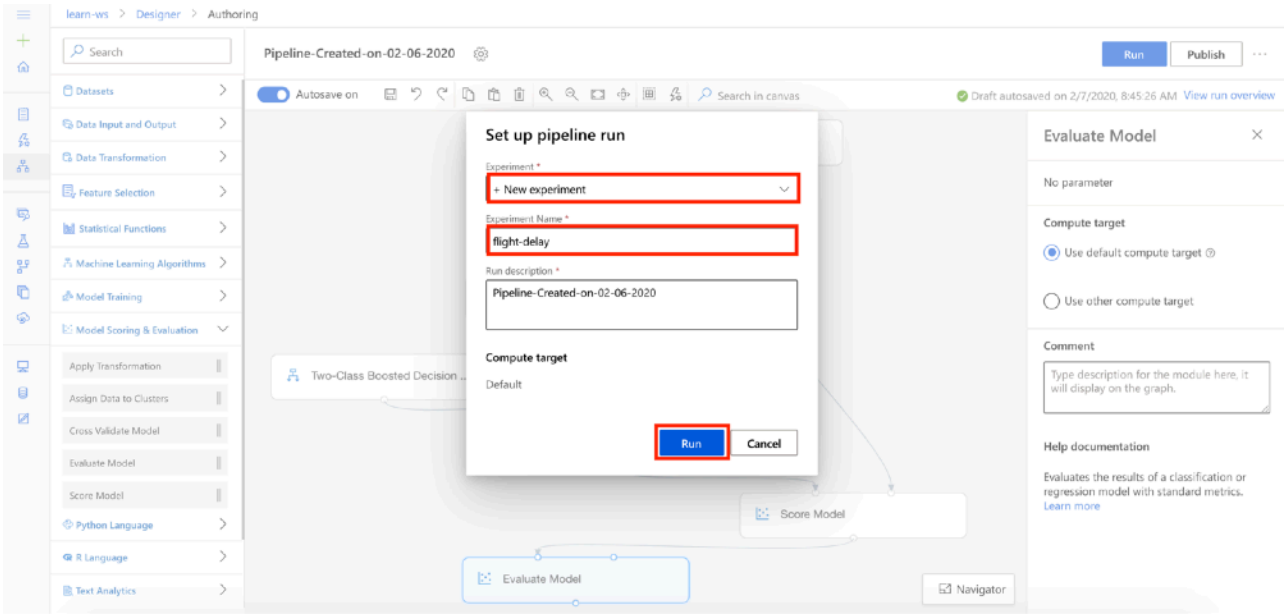
Task 1: Create Experiment and Submit Pipeline

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:` **flight-delay**, and then select **Submit**.

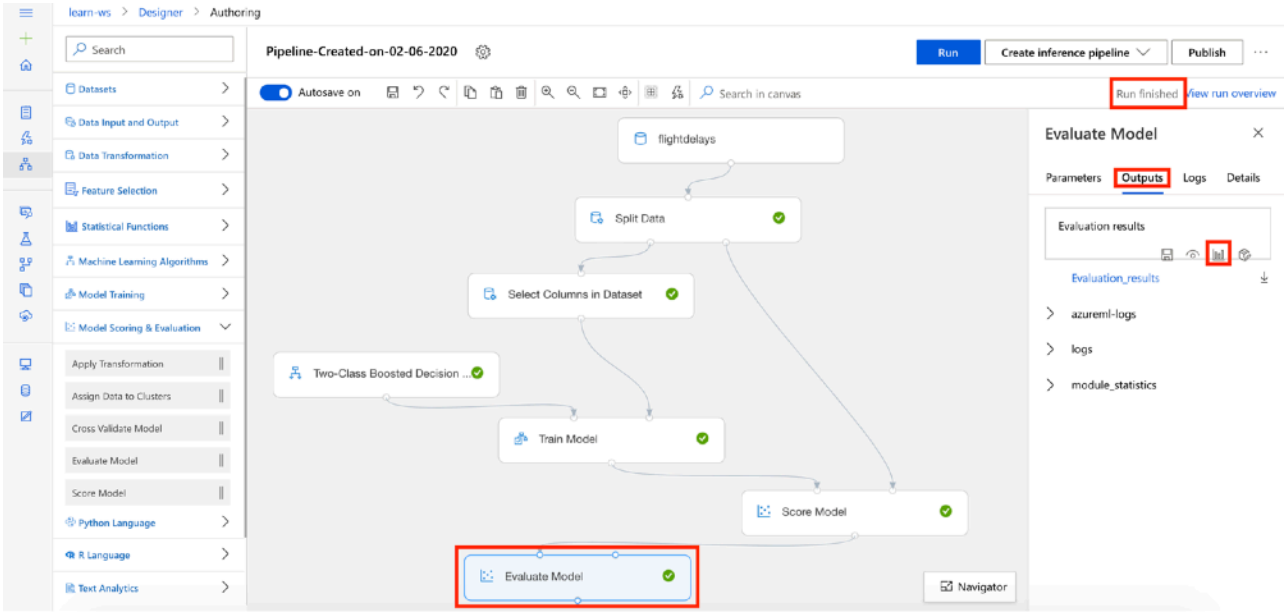


3. Wait for pipeline run to complete. It will take around **10 minutes** to complete the run.
4. While you wait for the model training to complete, you can learn more about the classification algorithm used in this lab by selecting Two-Class Boosted Decision Tree.

Exercise 4: Visualize the Evaluation Results

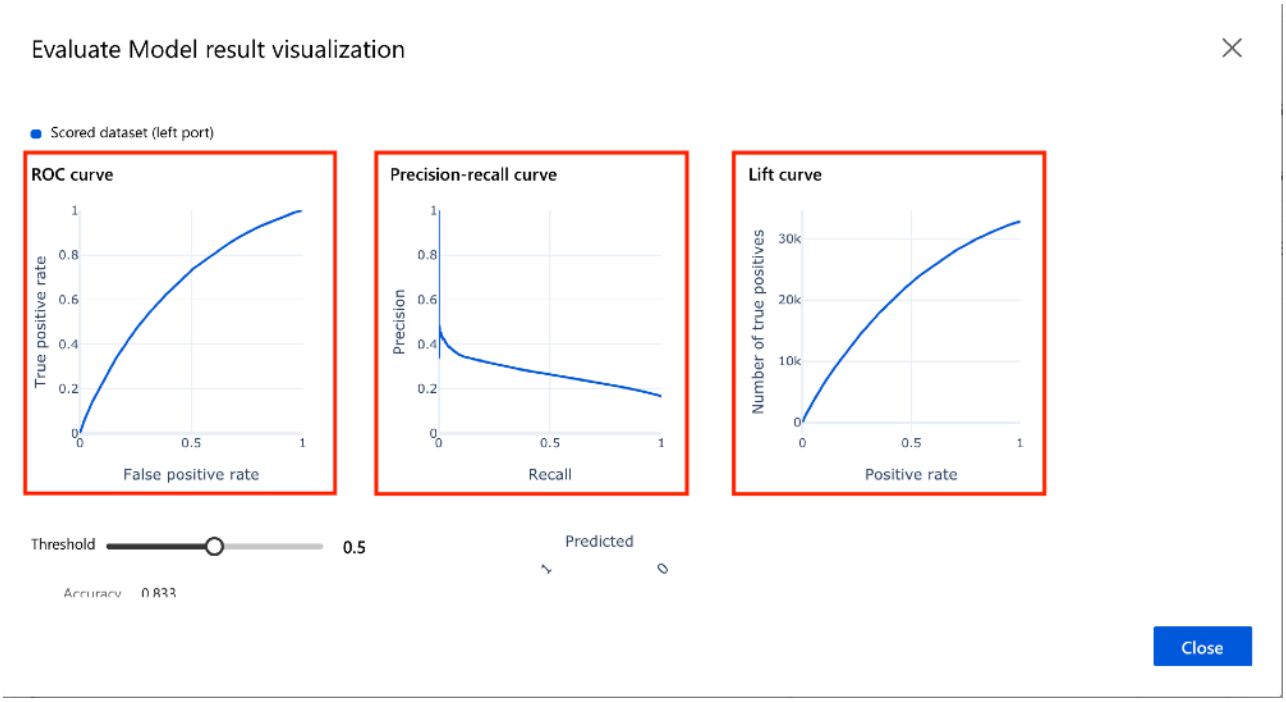
Task 1: Open the Result Visualization Dialog

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

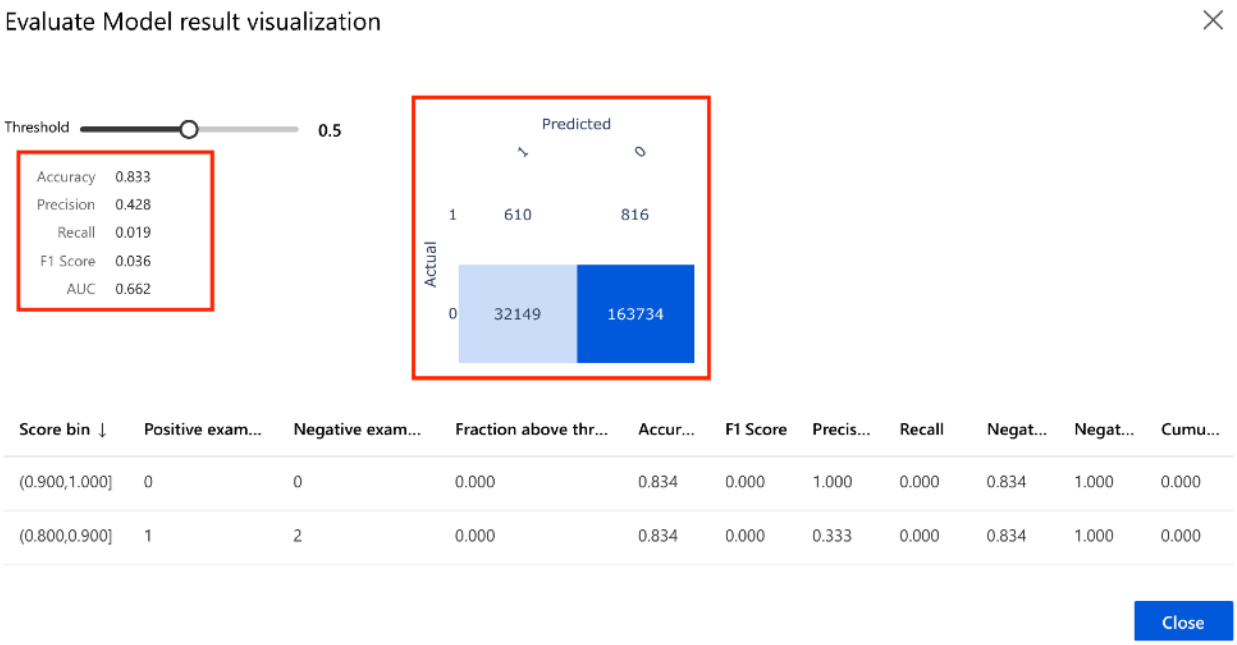


Task 2: Evaluate Model Performance

1. Evaluate the model performance by reviewing the various evaluation curves, such as **ROC curve**, **Precision-recall curve**, and **Lift curve**.



2. Scroll down to review the following:
    1. Review the key metrics for classifiers: **Accuracy, Precision, Recall, F1 Score, and AUC**
    2. Review the binary classifier's **Confusion Matrix**



Next Steps

Congratulations! You have trained and evaluated your first ensemble machine learning model. 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.

Lab 7 - Train a Simple Classifier with Automated ML

Overview

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

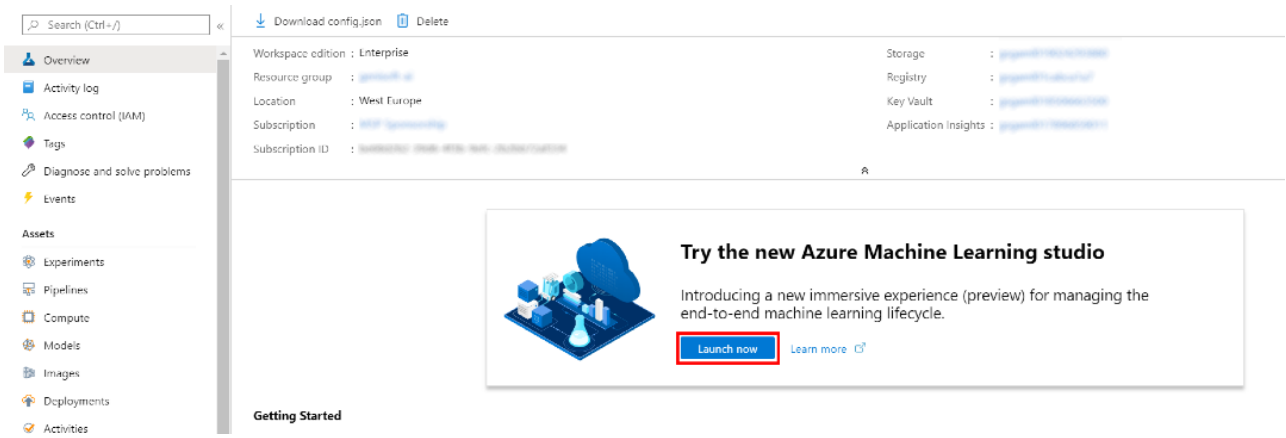
- Select your experiment type: Classification, Regression or Time Series Forecasting
- Data source, formats, and fetch data
- Choose your compute target
- Automated machine learning experiment settings
- Run an automated machine learning experiment
- Explore model metrics
- Register and deploy model

You can create and run automated machine learning experiments in code using the Azure ML Python SDK or if you prefer a no code experience, you can also create your automated machine learning experiments in Azure Machine Learning Studio.
In this lab, you learn how to create, run, and explore automated machine learning experiments in the Azure Machine Learning Studio without a single line of code. As part of this lab, we will be using the `Flight Delays` data set that is enhanced with the weather data. Based on the enriched dataset, we will use automated machine learning to find the best performing classification model to predict if a particular flight will be delayed by 15 minutes or more.

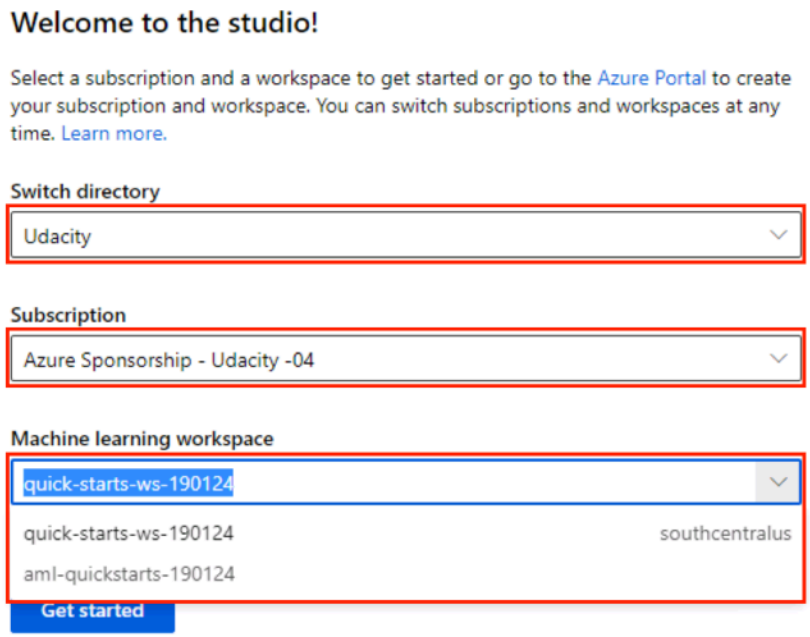
Exercise 1: Register Dataset with Azure Machine Learning studio

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.

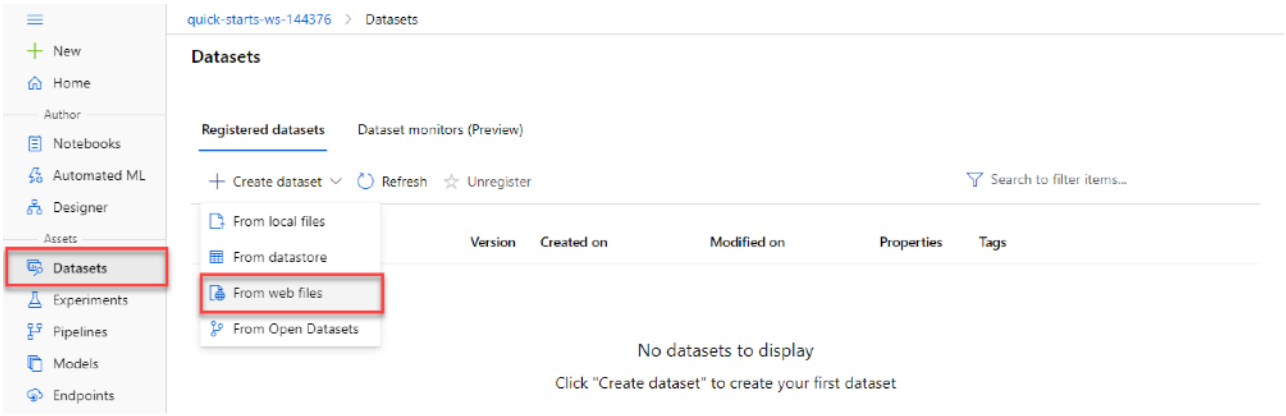


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



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**.

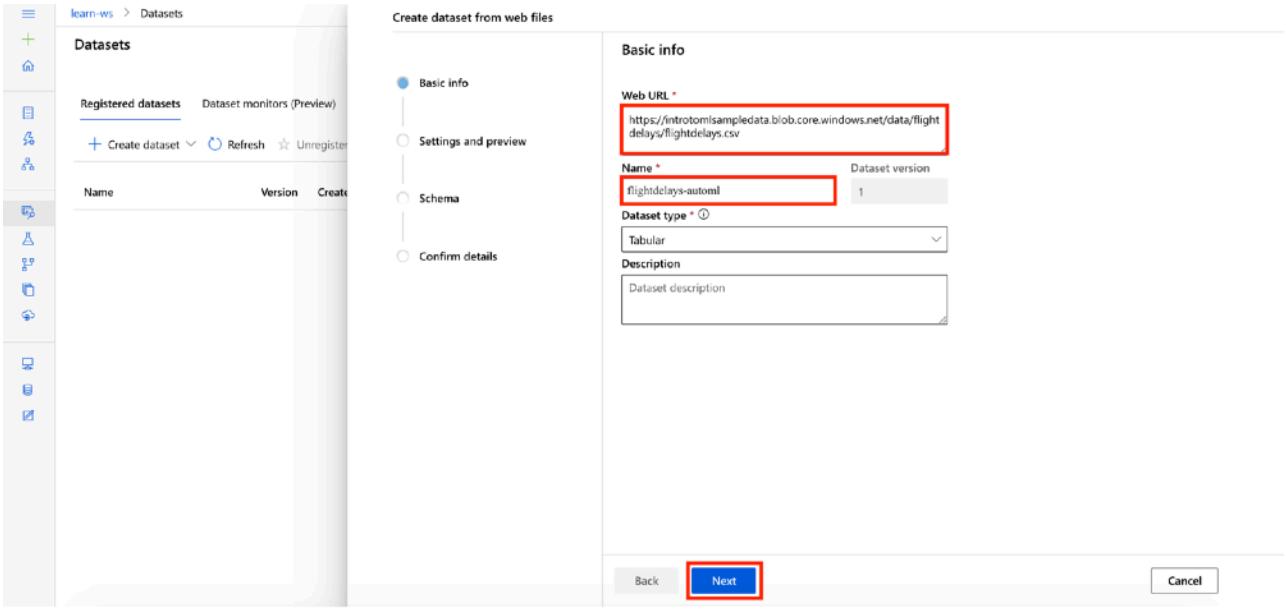
4. From the studio, select **Datasets, + Create dataset, From web files**. This will open the `Create dataset from web files` dialog on the right.



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

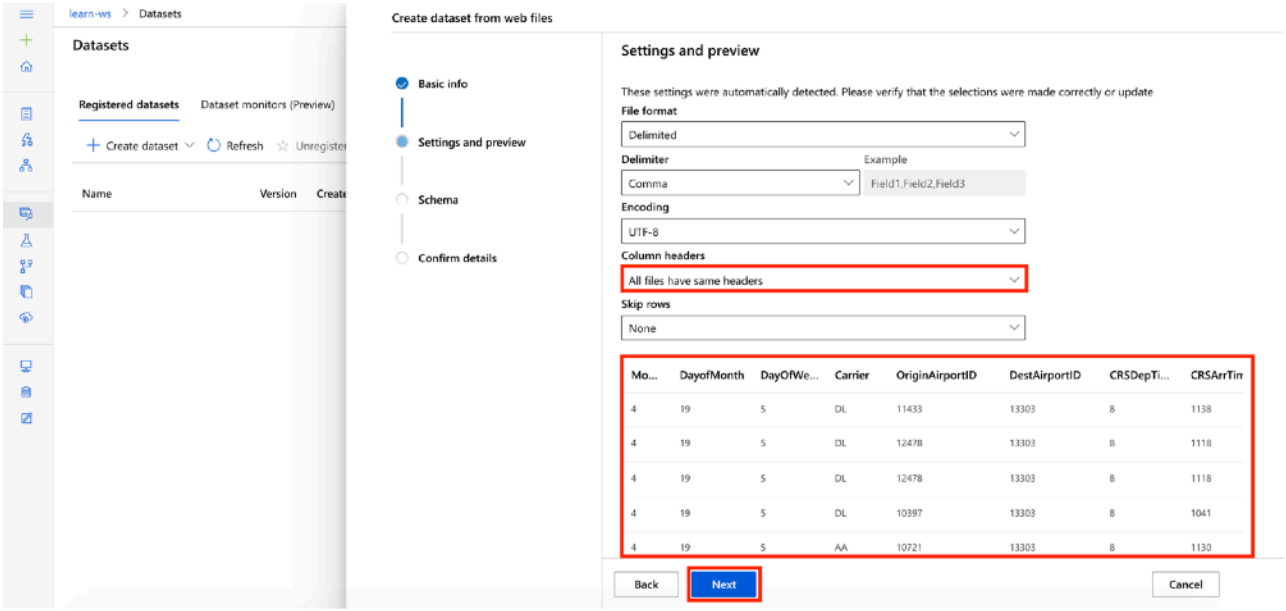
```
https://introtomlsampledata.blob.core.windows.net/data/flightdelays/flightdelays.csv
```

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



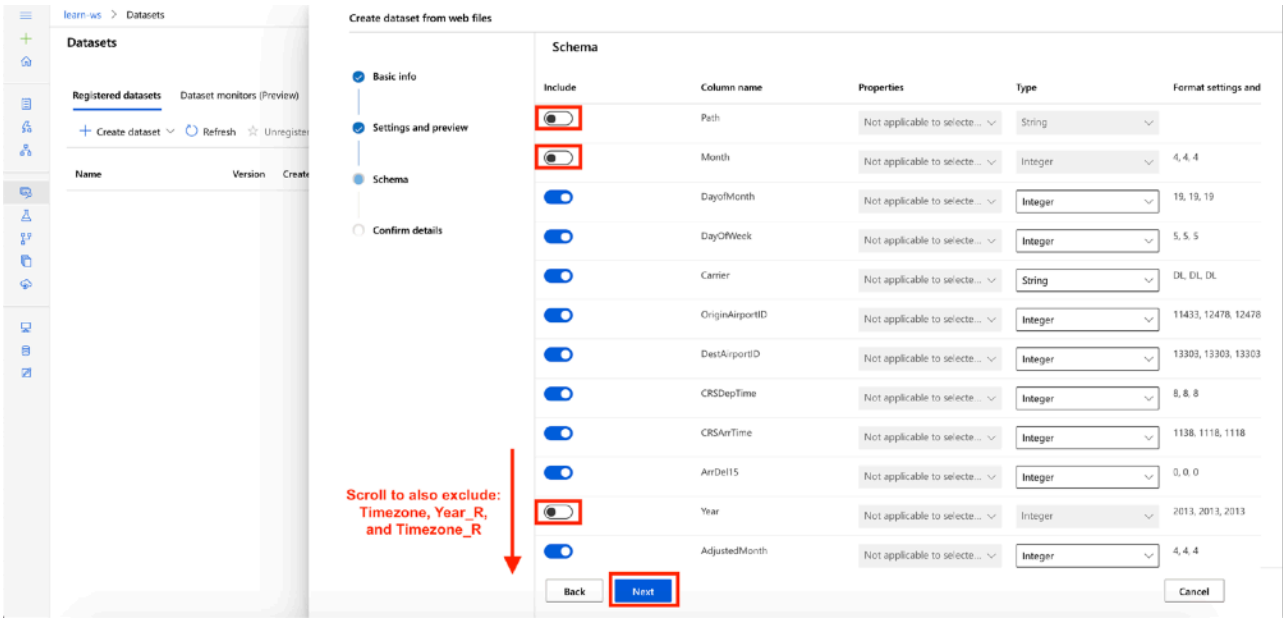
Task 2: Preview Dataset

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**



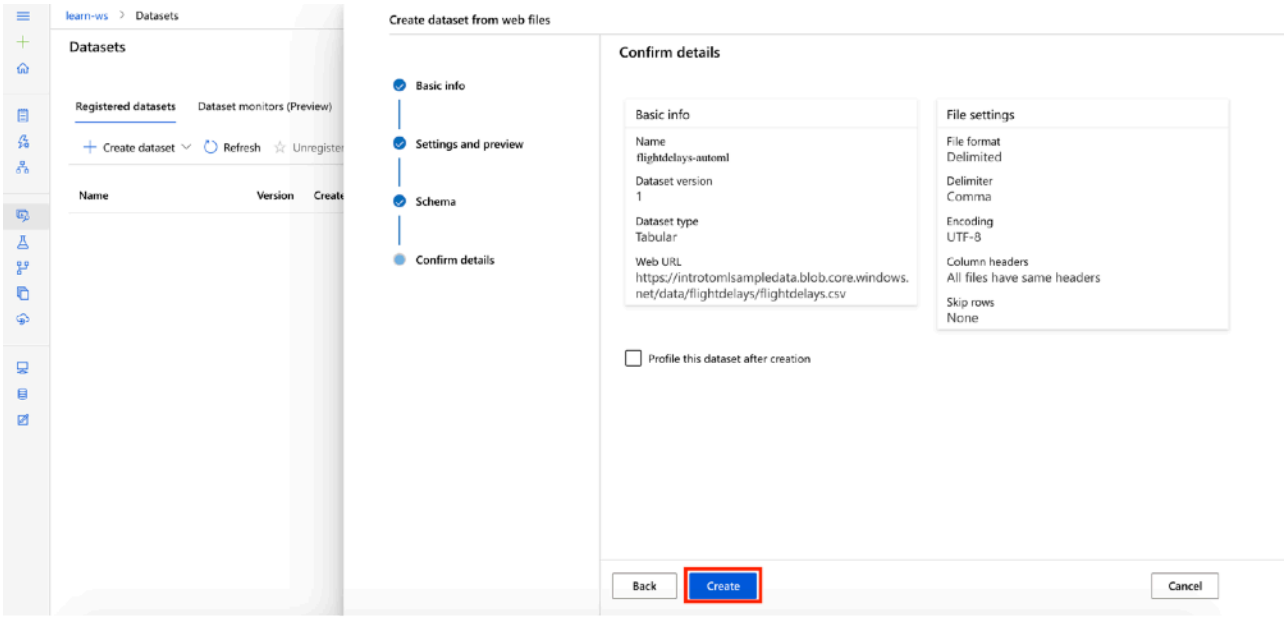
Task 3: Select Columns

1. Select columns from the dataset to include as part of your training data. Exclude the following columns: **Path, Month, Year, Timezone, Year_R, Timezone_R**, and then select **Next**



Task 4: Create Dataset

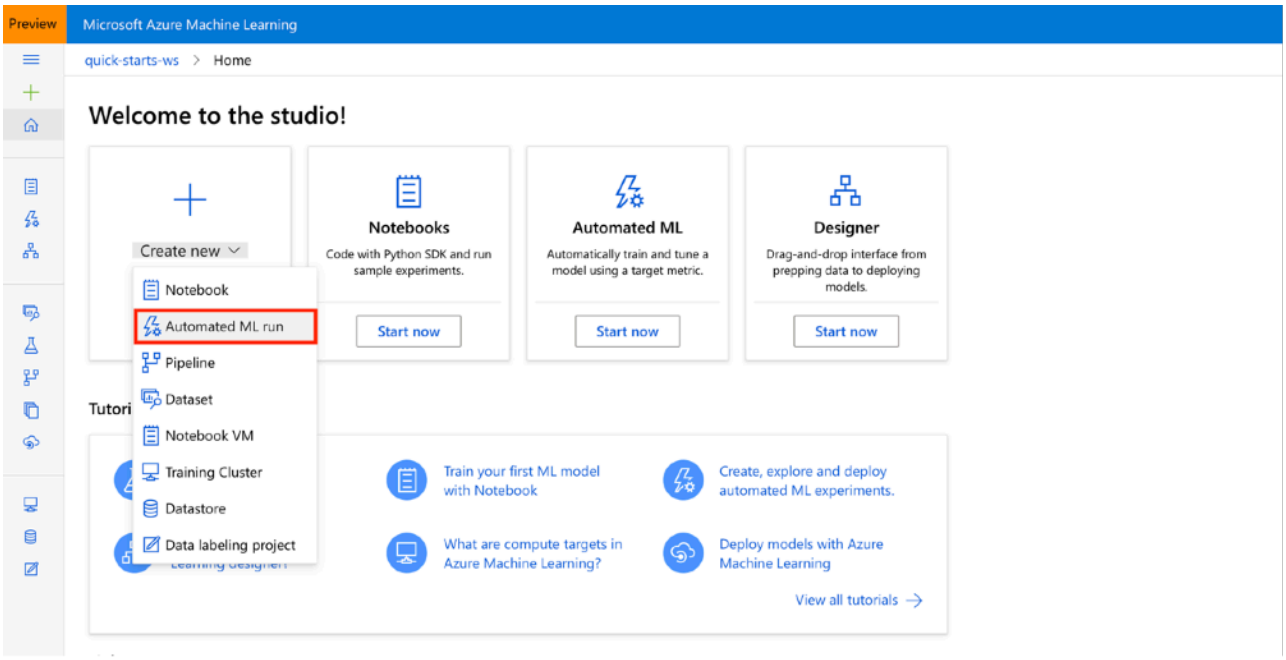
1. Confirm the dataset details and select **Create**



Exercise 2: Setup New Automated Machine Learning Experiment

Task 1: Create New Automated Machine Learning Experiment

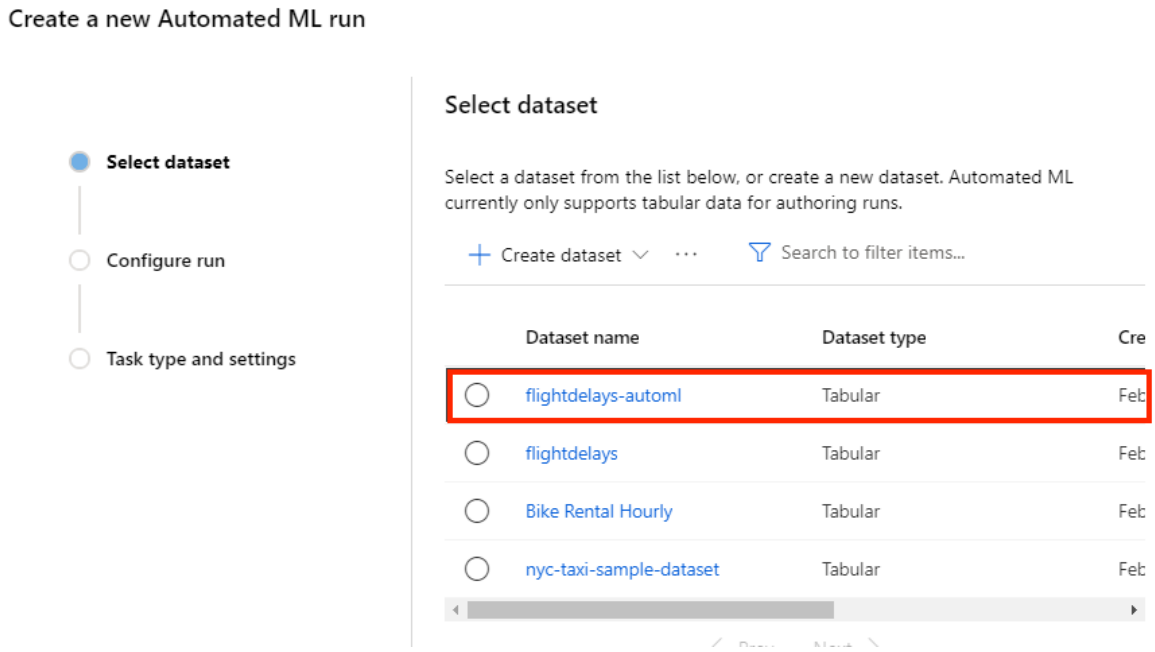
1. From the studio home, select **Create new, Automated ML run**



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

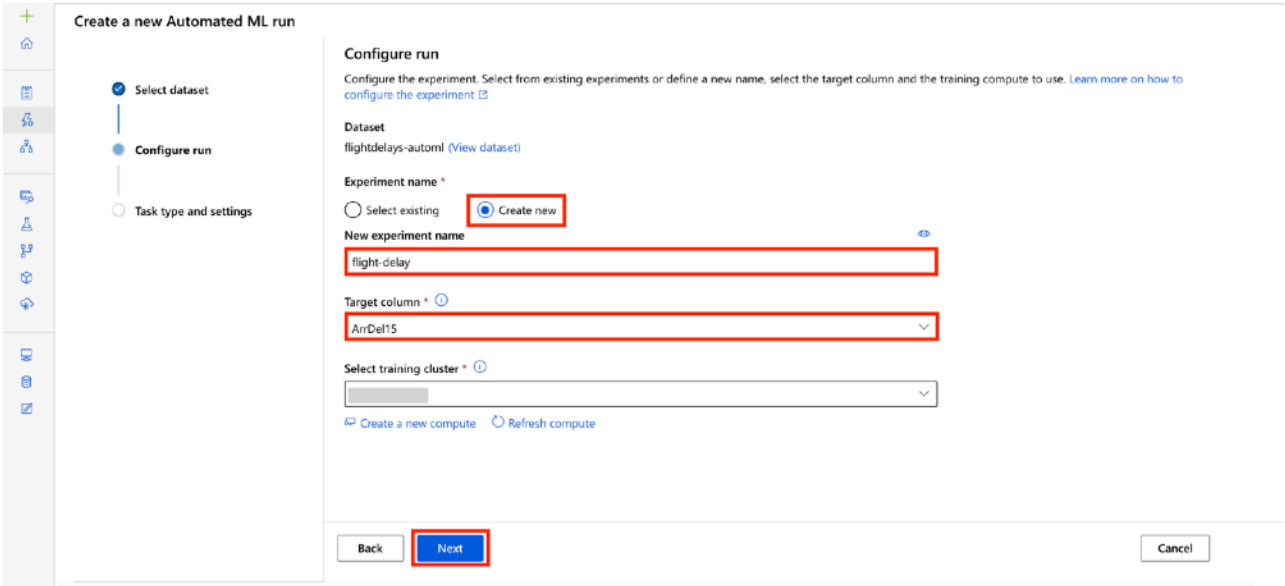
Task 2: Select Training Data

1. Select the dataset `flightdelays-automl` and then select **Next**



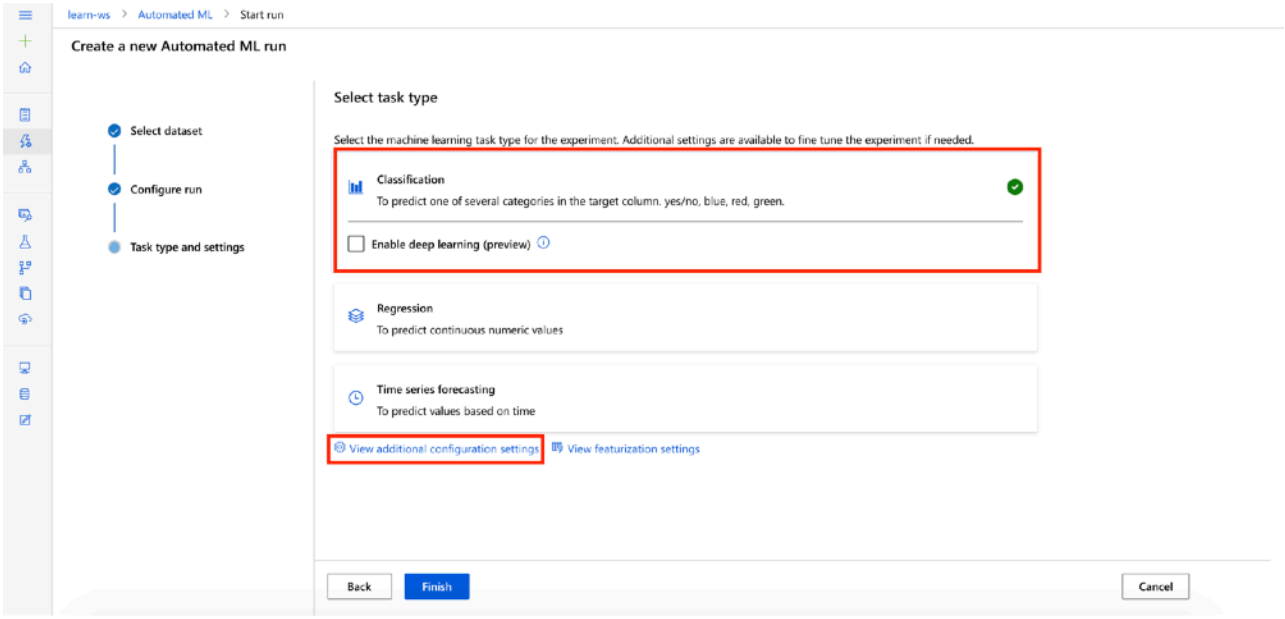
Task 3: Create a new Automated ML run

1. Provide an experiment name: **flight-delay**
2. Select target column: **ArrDel15**
3. Select compute target: **select the available compute**
4. Select **Next**

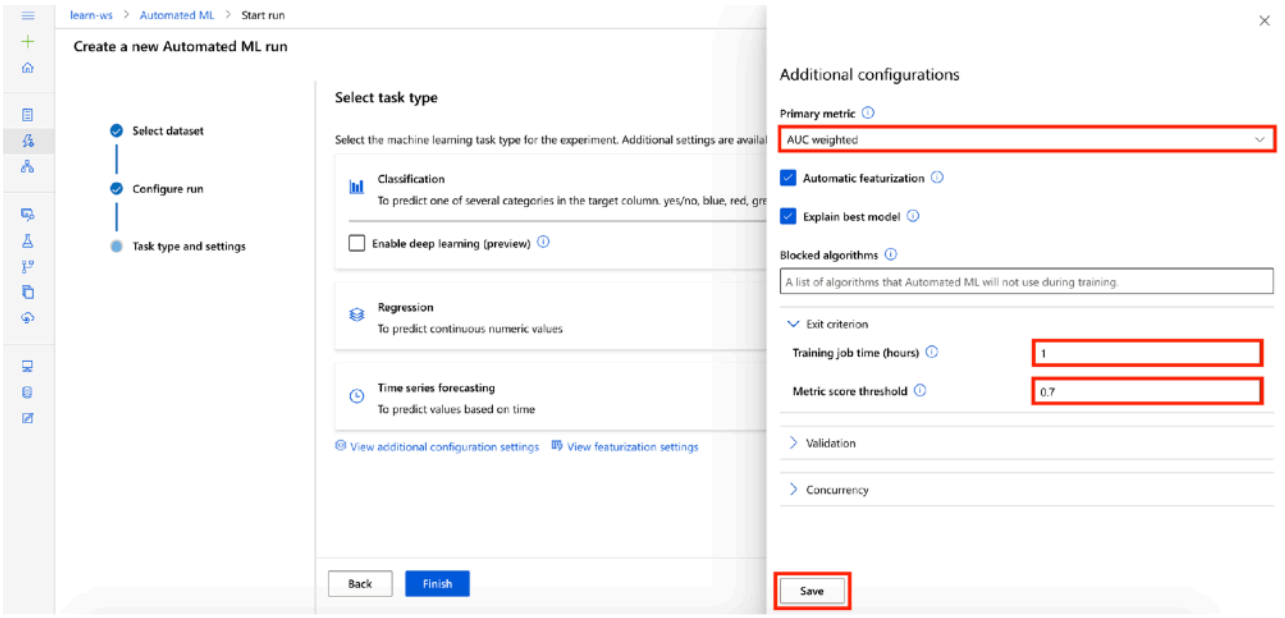


Task 4: Setup Task type and Settings

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, Training job time (hours): `1`
    3. Exit criteria, Metric score threshold: `0.7`

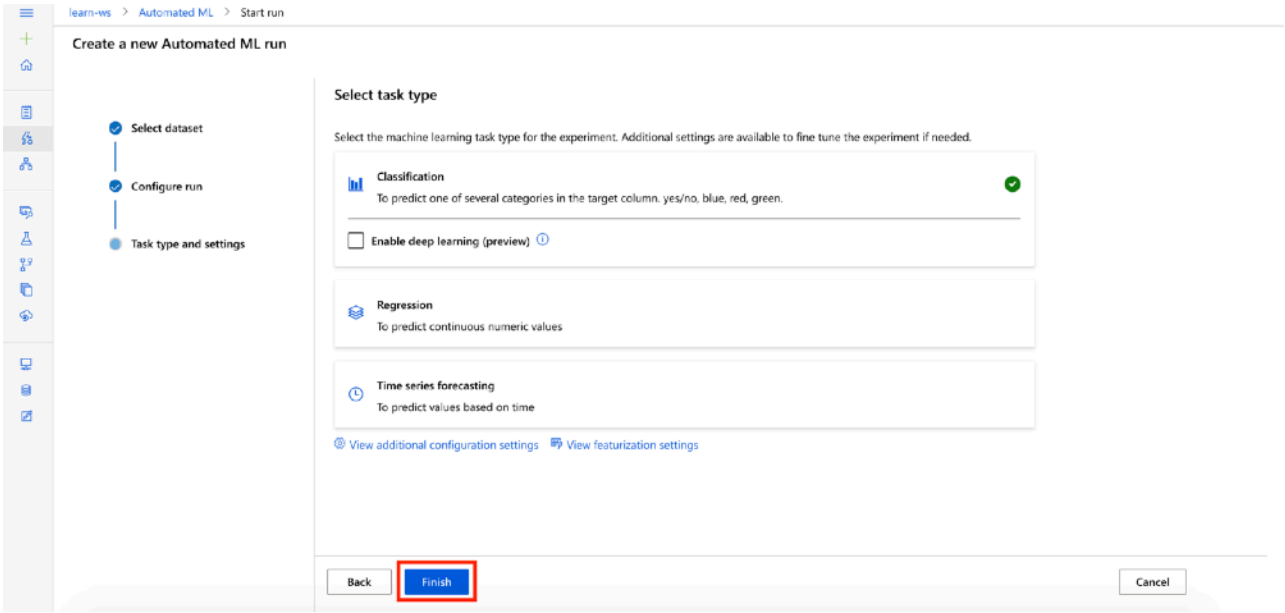


4. Note that we are setting a metric score threshold to limit the training time. In practice, for initial experiments, you will typically only set the training job time to allow AutoML to discover the best algorithm to use for your specific data.

Exercise 3: Start and Monitor Experiment

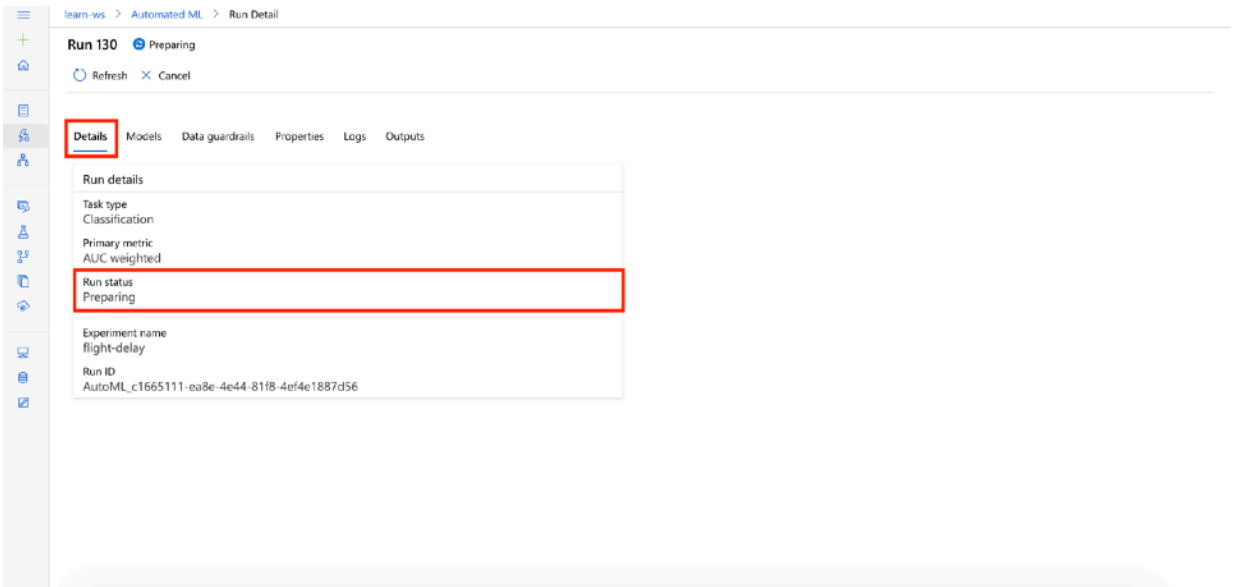
Task 1: Start Experiment

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

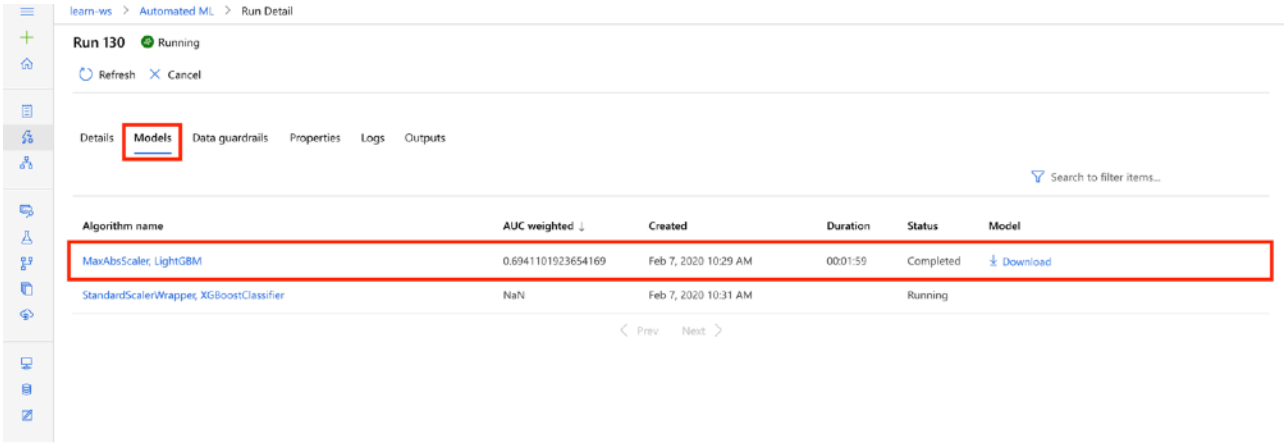


Task 2: Monitor Experiment

1. The experiment will run for about 30 min. Note that most of the time will be spent in the data preparation step and once the data preparation is done, the experiment will take an additional 1-2 minutes to complete.
2. In the **Details** tab, observe the **run status** of the job.

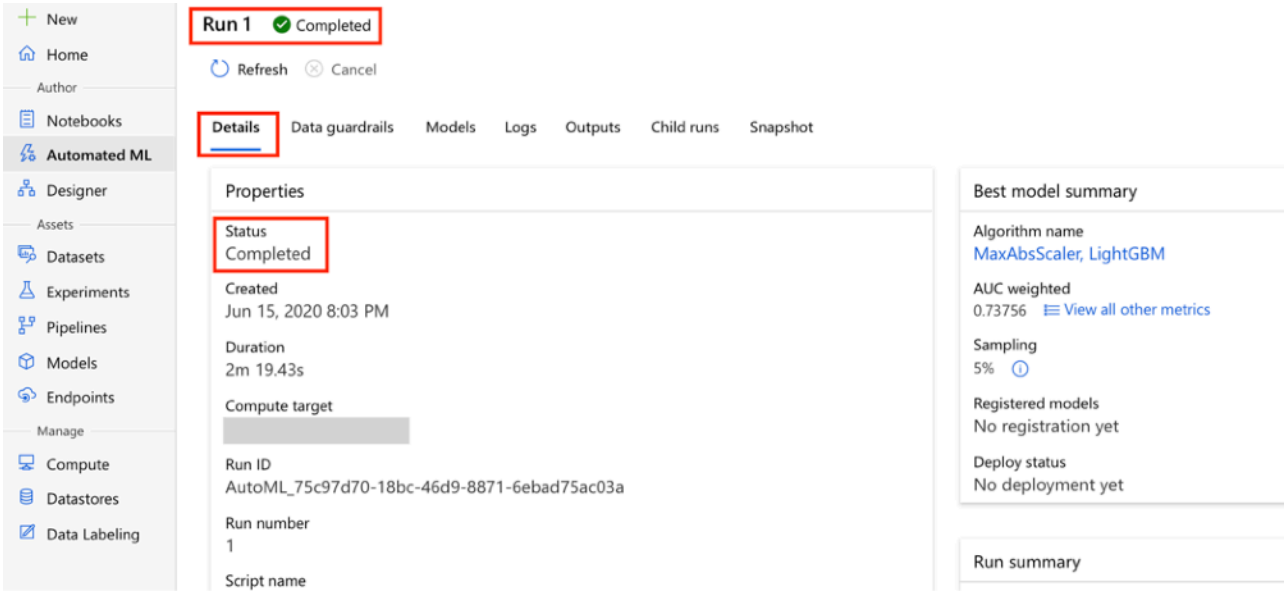


3. Select the **Models** tab, and observe the various algorithms the AutoML is evaluating. You can also observe the corresponding **AUC weighted** scores for each algorithm.



Note that we have set a metric score threshold to limit the training time. As a result you might see only one algorithm in your models list.

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

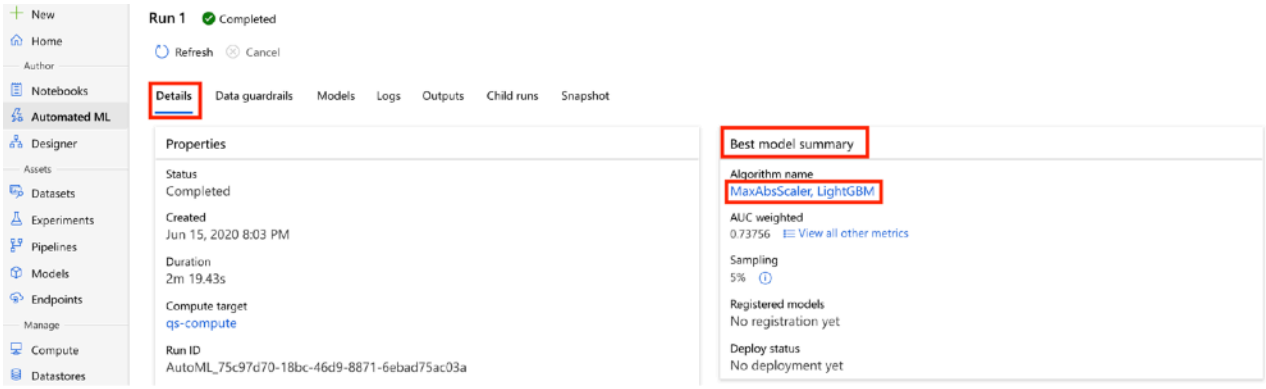


5. While you wait for the model training to complete, you can learn to view and understand the charts and metrics for your automated machine learning run by selecting Understand automated machine learning results.

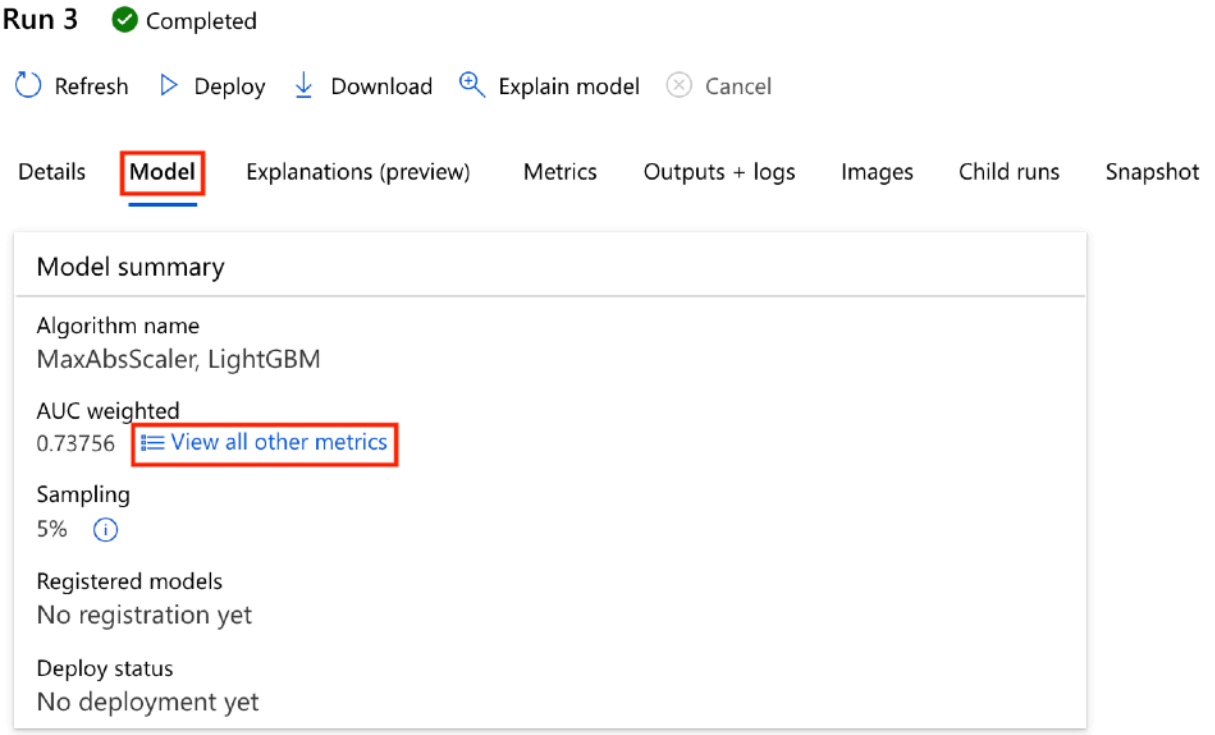
Exercise 4: Review Best Model's Performance

Task 1: Review Best Model Performance

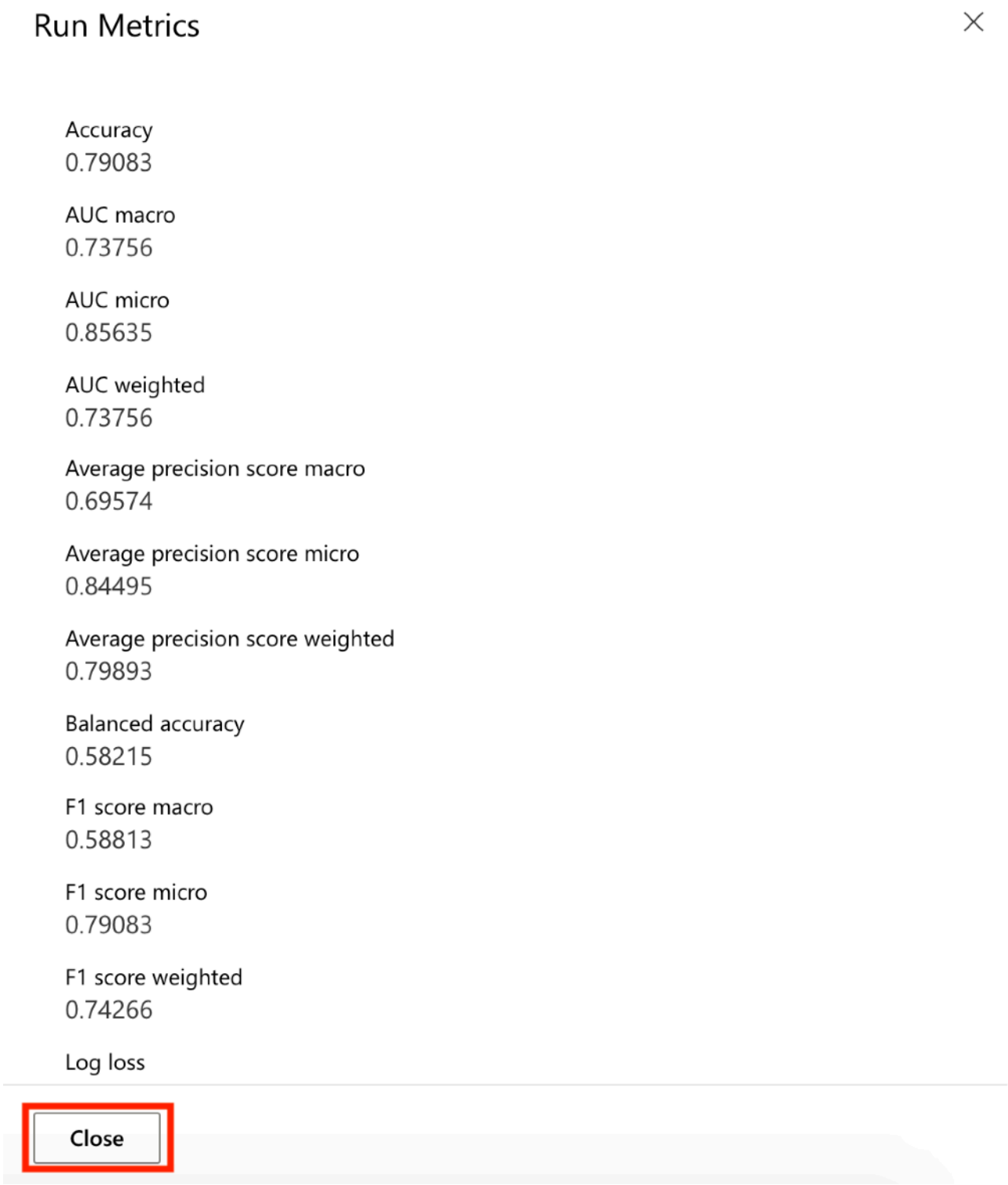
1. The `Details` tab shows the `Best model summary`. Next, select **Algorithm name** to review the model details.



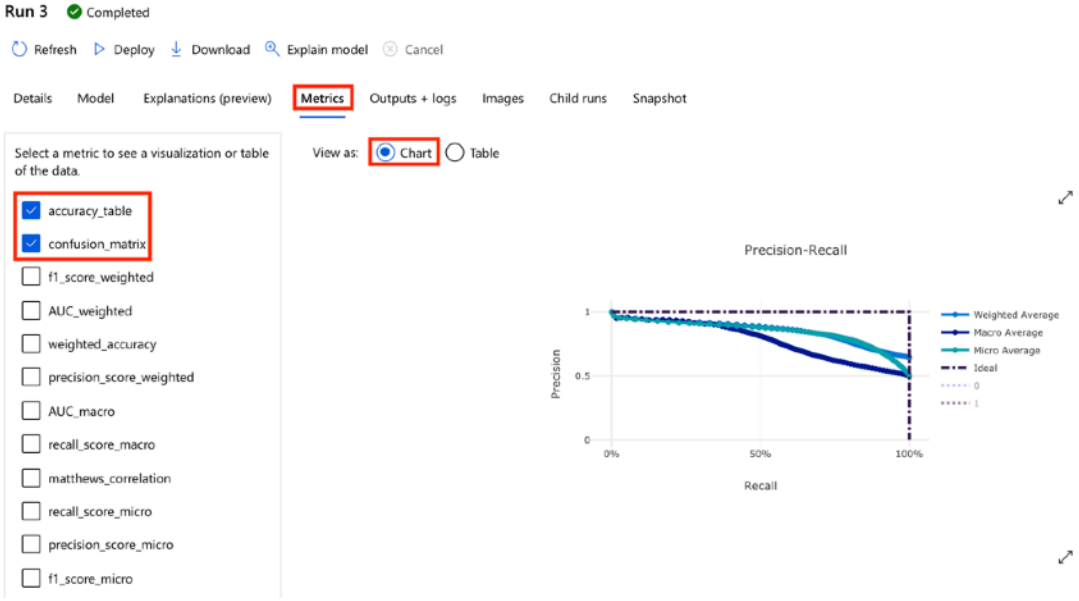
2. From the `Model details` tab, to view the various metrics to evaluate the best model performance, select **View all other metrics**.



3. Review the model performance metrics and then select **Close**.



4. Next, select **Metrics** to review the various model performance curves, such as Precision-Recall, ROC, Calibration curve, Gain & Lift curves, and Confusion matrix.



Next Steps

Congratulations! You have trained and evaluated your first automated machine learning model. 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.

Lab 8 - Two-Class Classifiers Performance

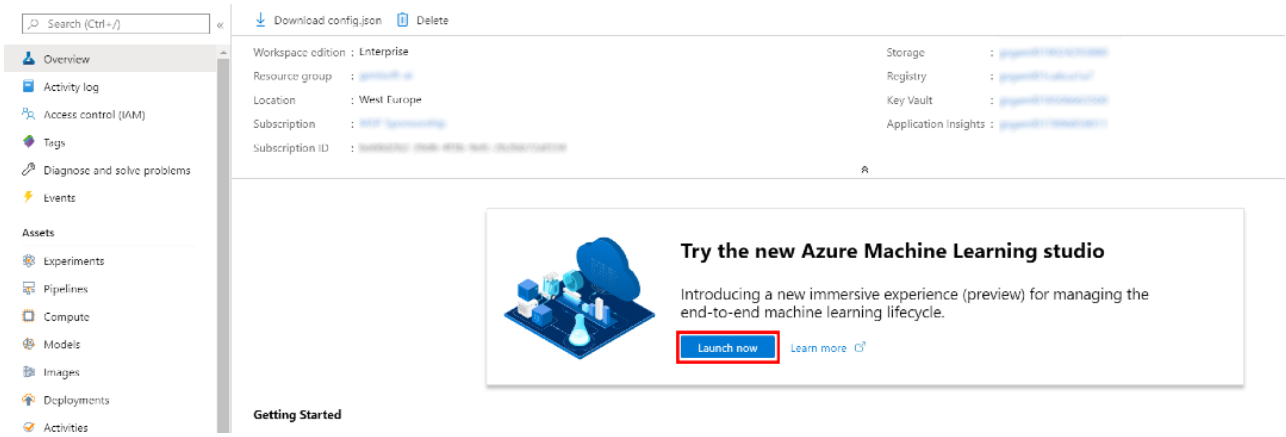
Overview

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.

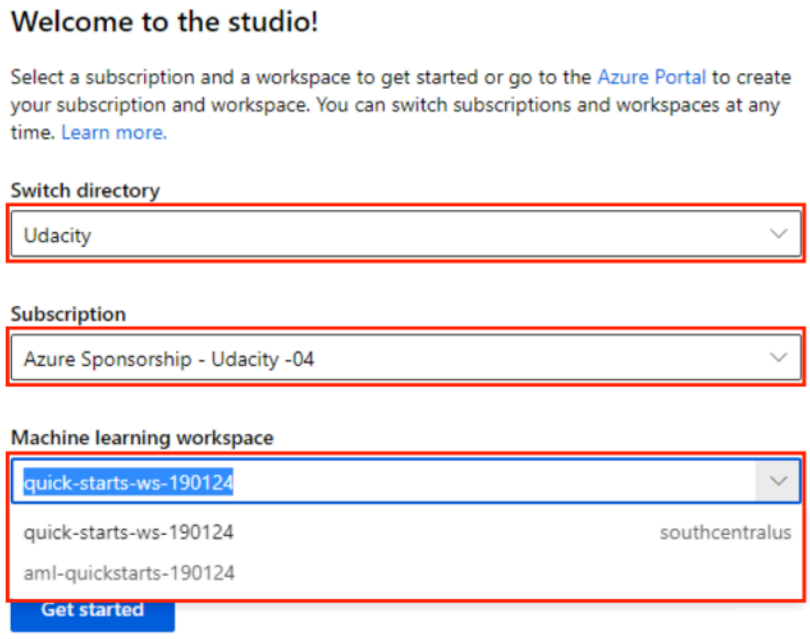
Exercise 1: Create Training Pipeline

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.

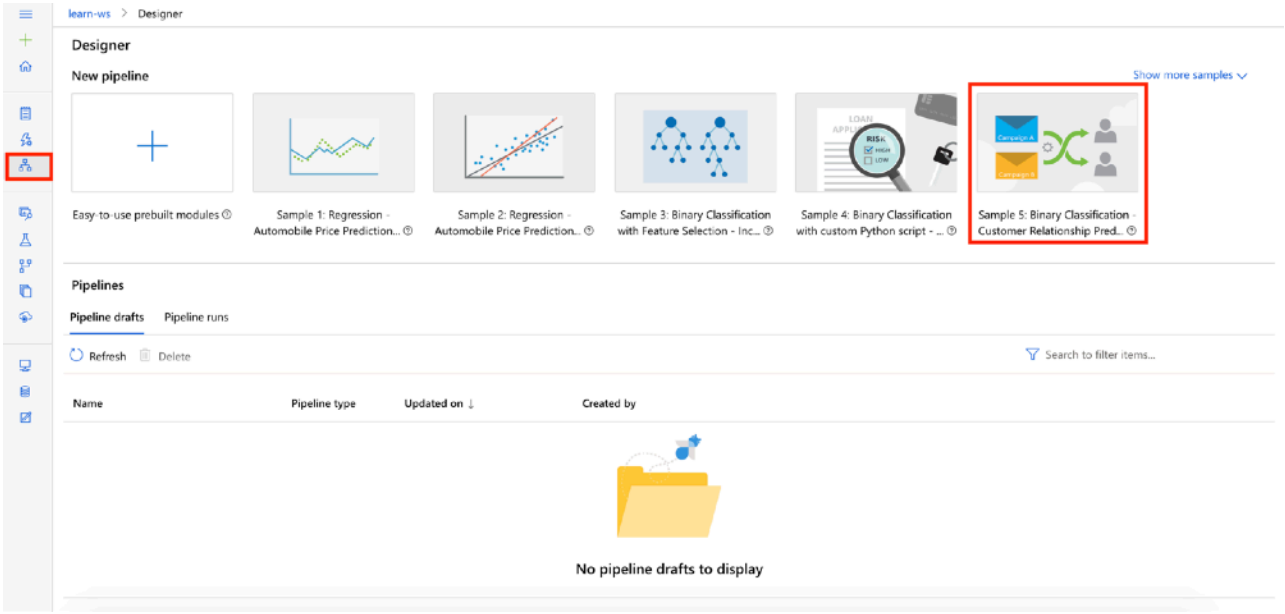


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



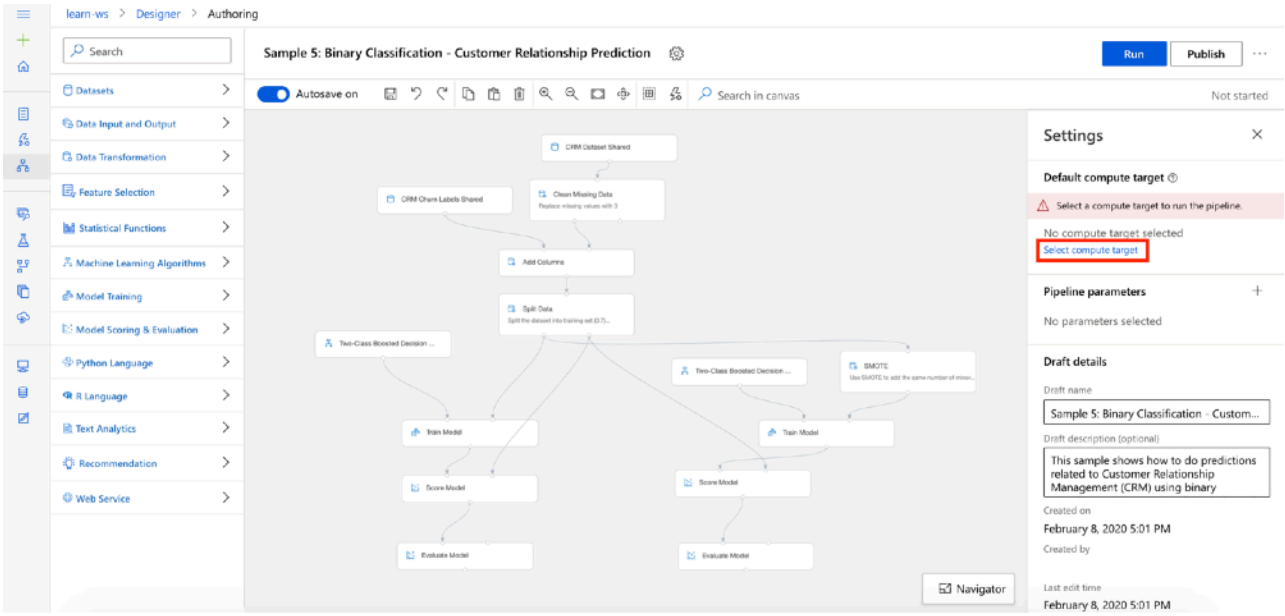
For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

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



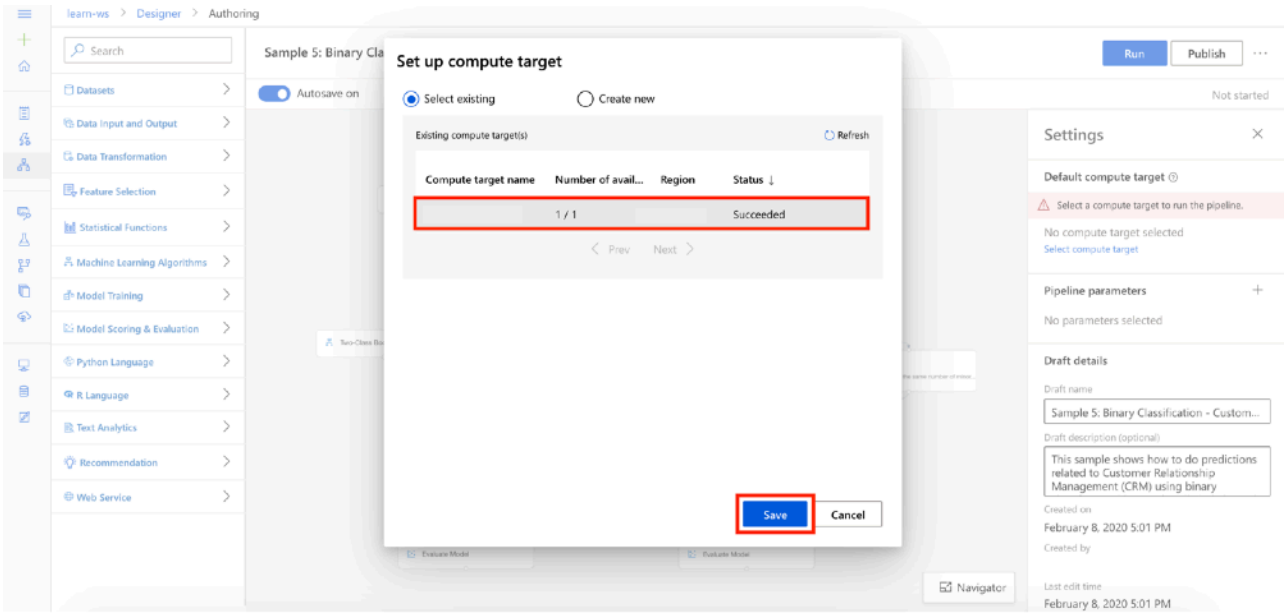
Task 2: Setup Compute Target

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



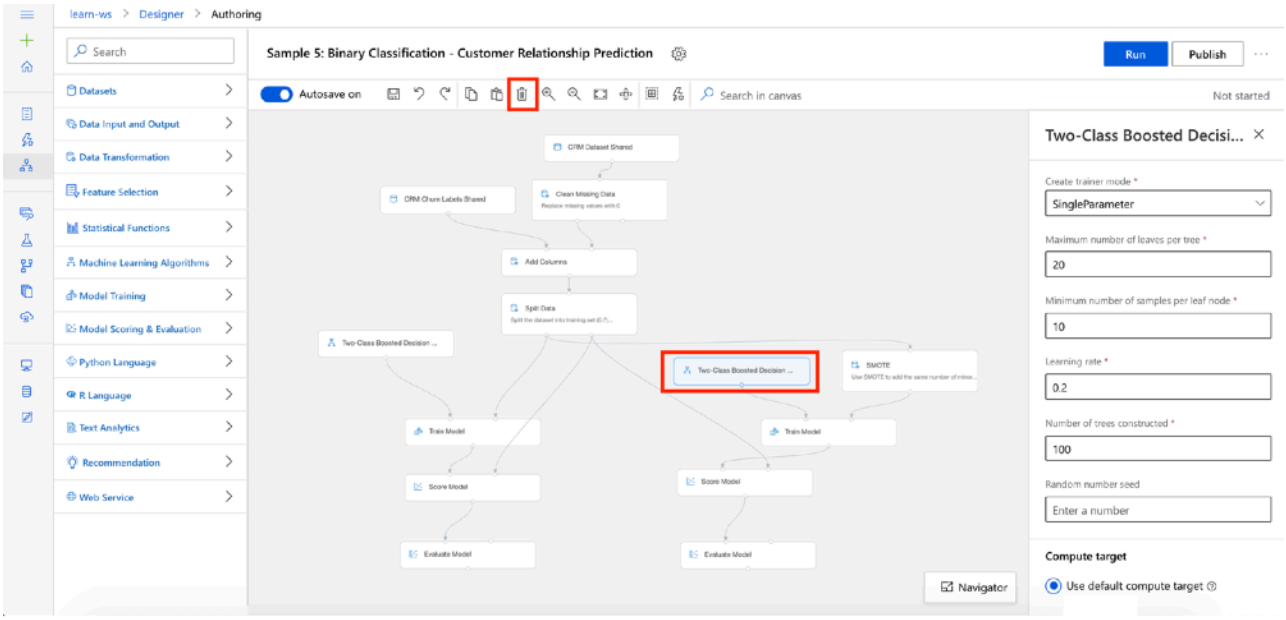
2. In the `Set up compute target` editor, select the available compute, and then select **Save**.

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.

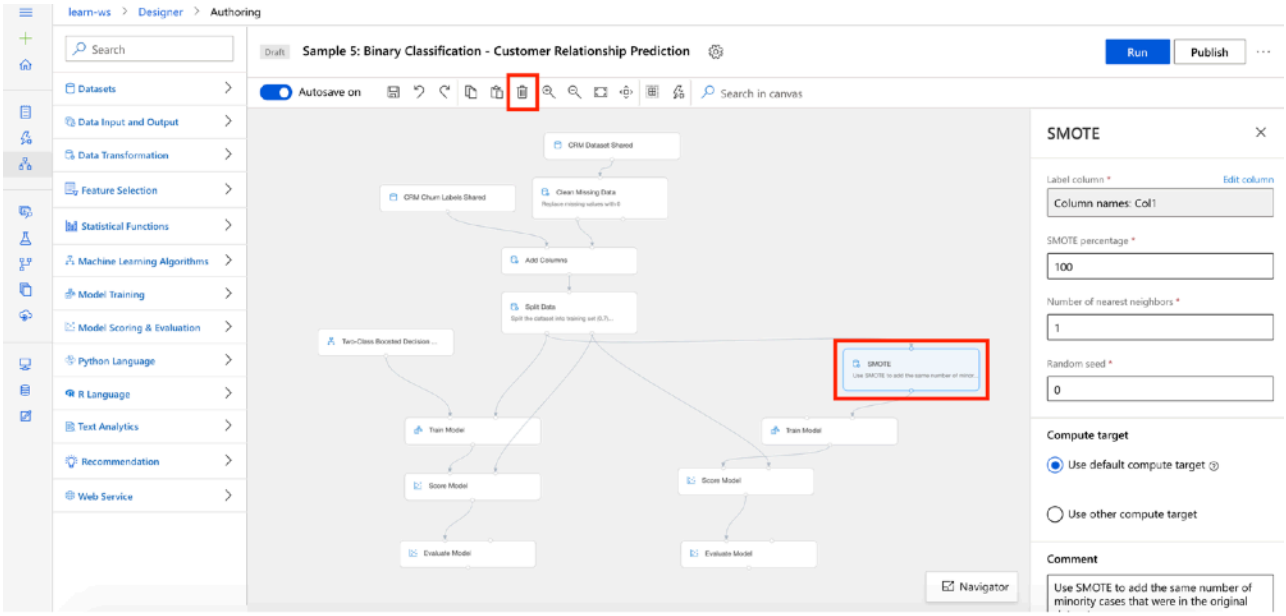


Task 3: Delete Pipeline Modules

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

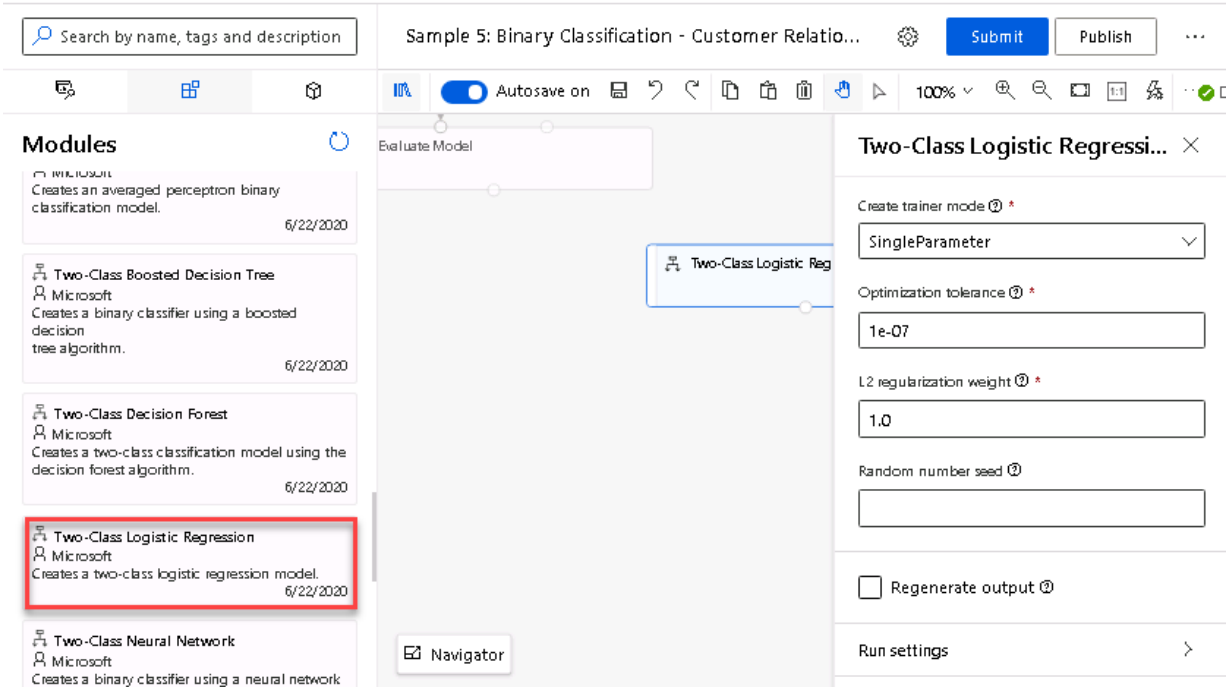


2. From the `right-hand-side` of the pipeline, select the **SMOTE module** and then select the **Delete Icon**.



Task 4: Setup the Two-Class Logistic Regression Module

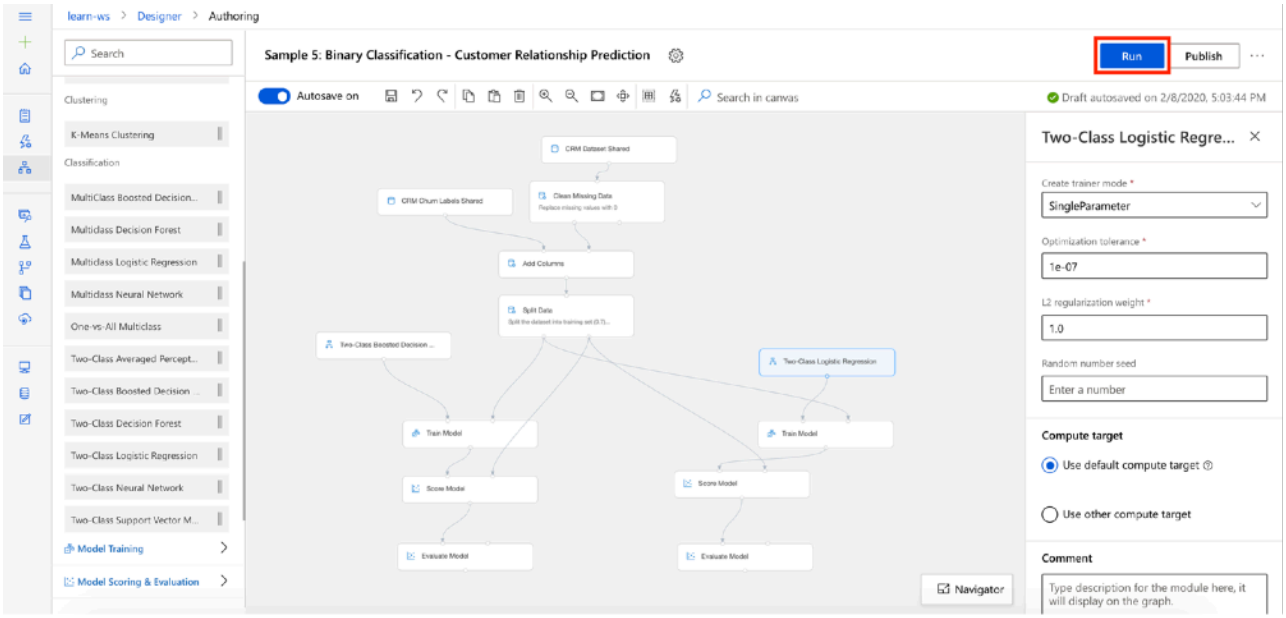
1. Select **Machine Learning Algorithms** section in the left navigation. Follow the steps outlined below:
    1. Select the **Two-Class Logistic Regression** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the `Two-Class Logistic Regression` module to the first 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

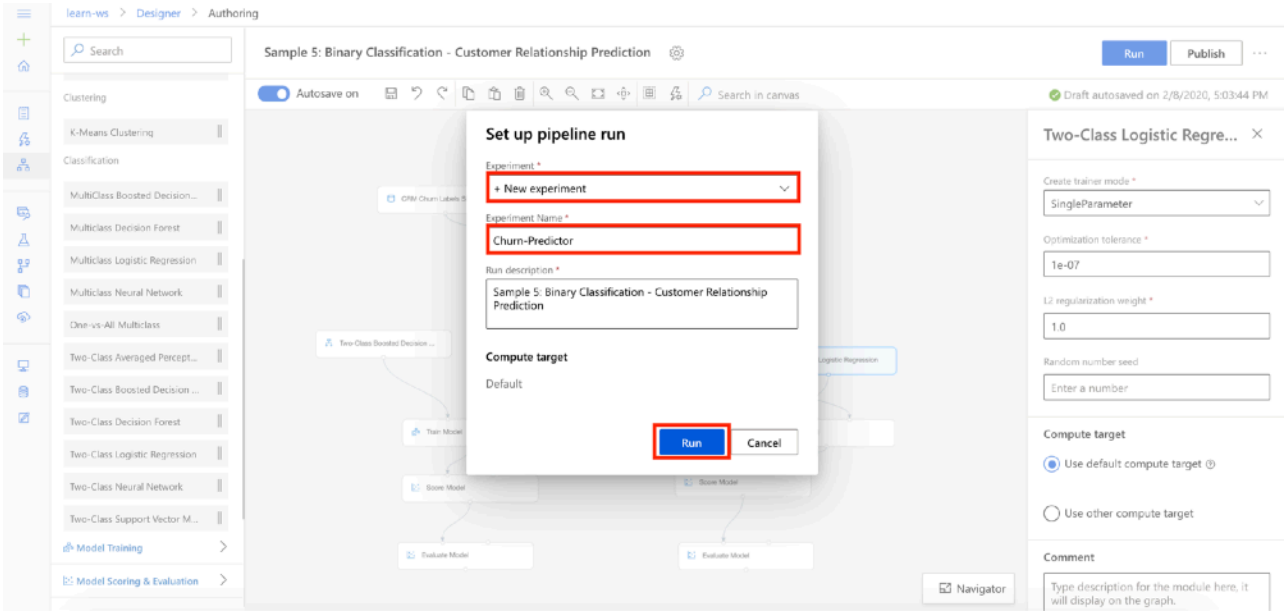
Task 1: Create Experiment and Submit Pipeline

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**.

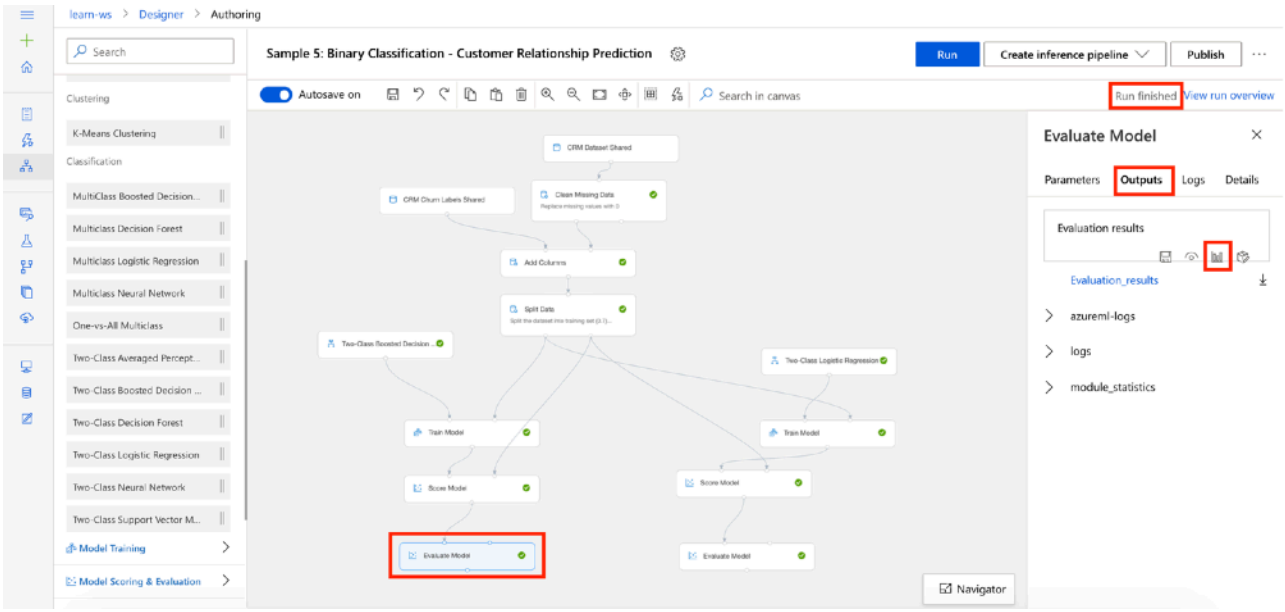


3. Wait for pipeline run to complete. It will take around **5 minutes** to complete the run.
4. While you wait for the model training to complete, you can learn more about the evaluation metrics for the classification algorithm used in this lab by selecting Metrics for classification models.

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.

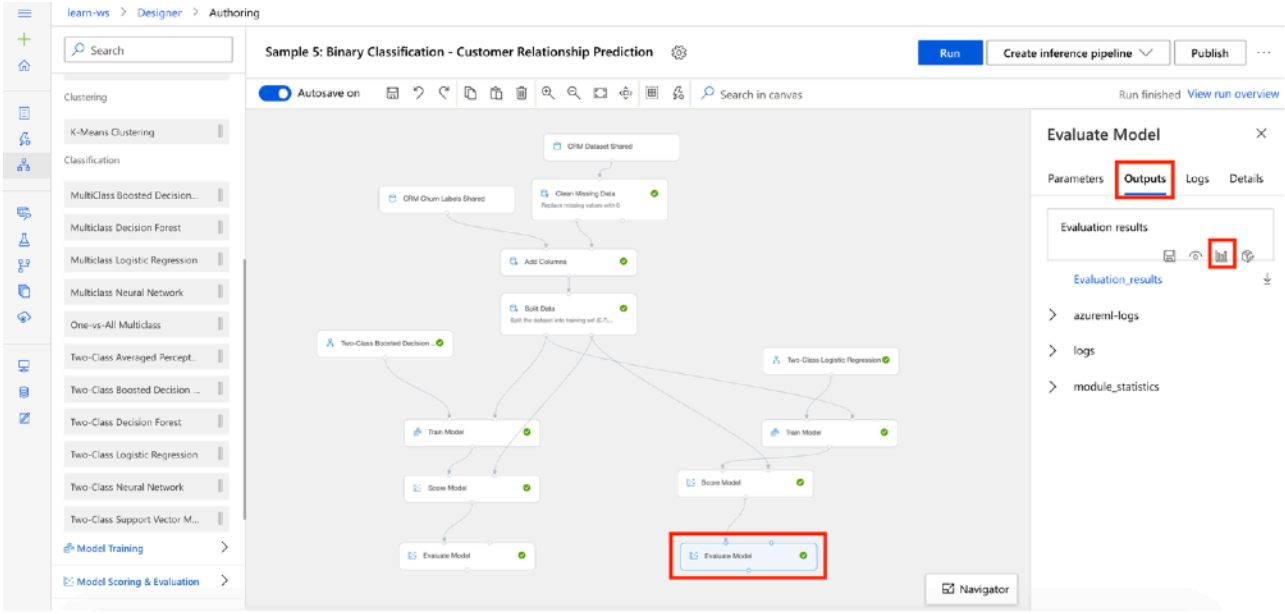


Task 2: Evaluate Two-Class Boosted Decision Tree Performance

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.

Lab 9 - Multi-Class Classifiers Performance

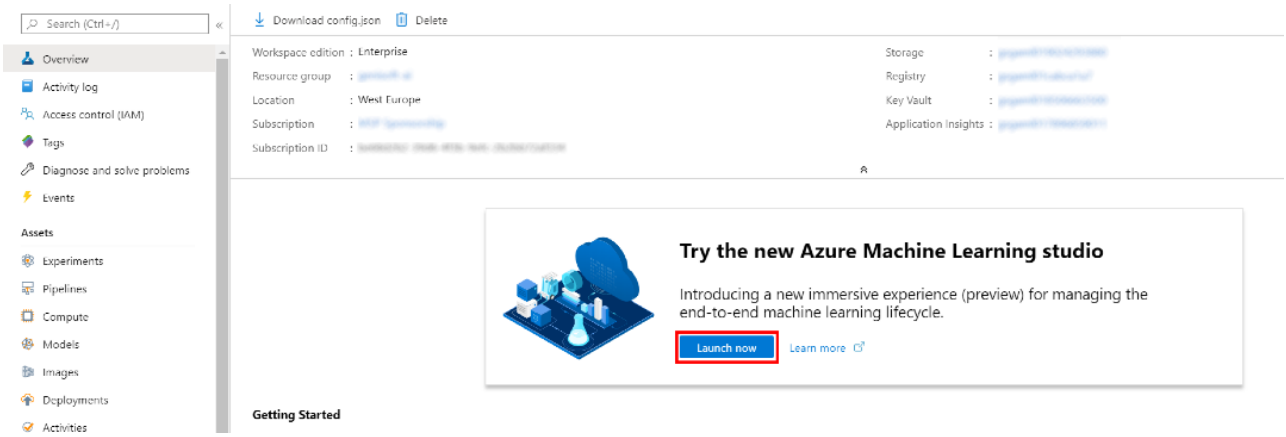
Overview

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.

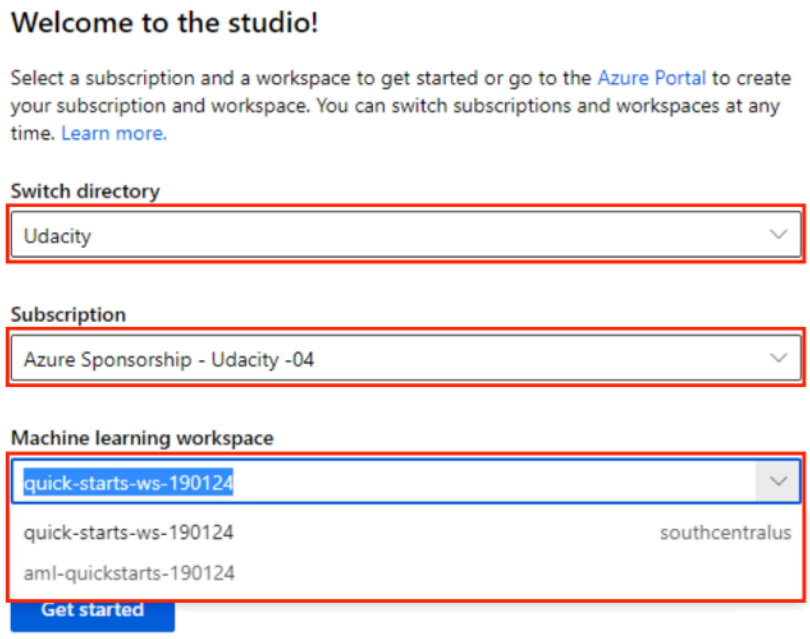
Exercise 1: Create Training Pipeline

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.

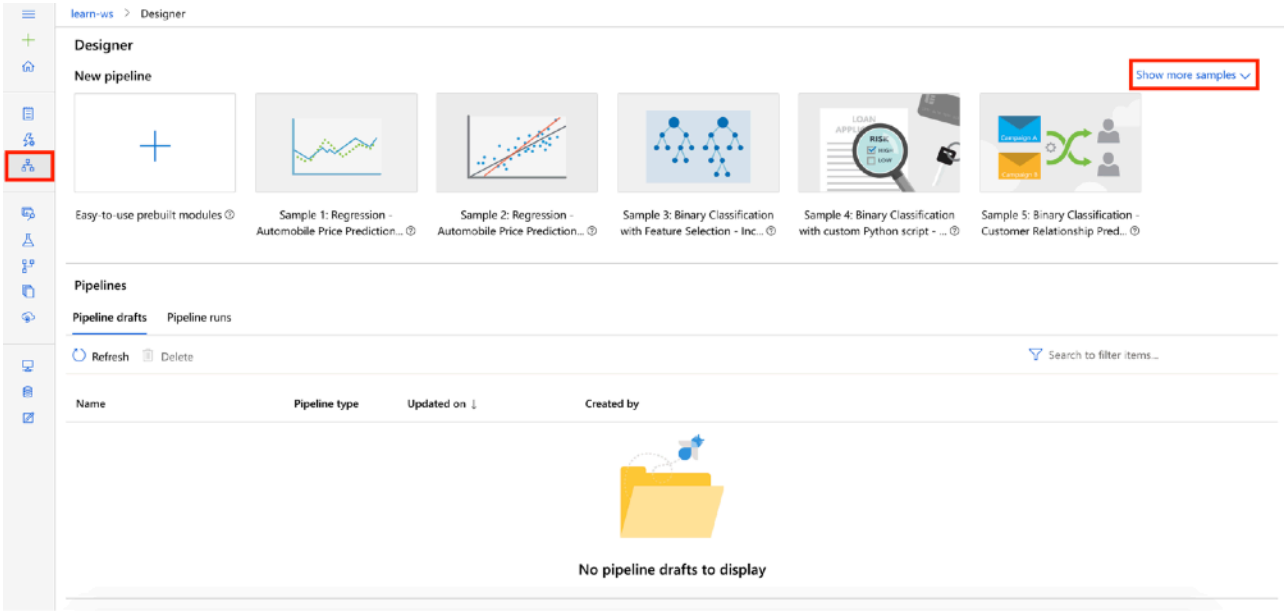


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:

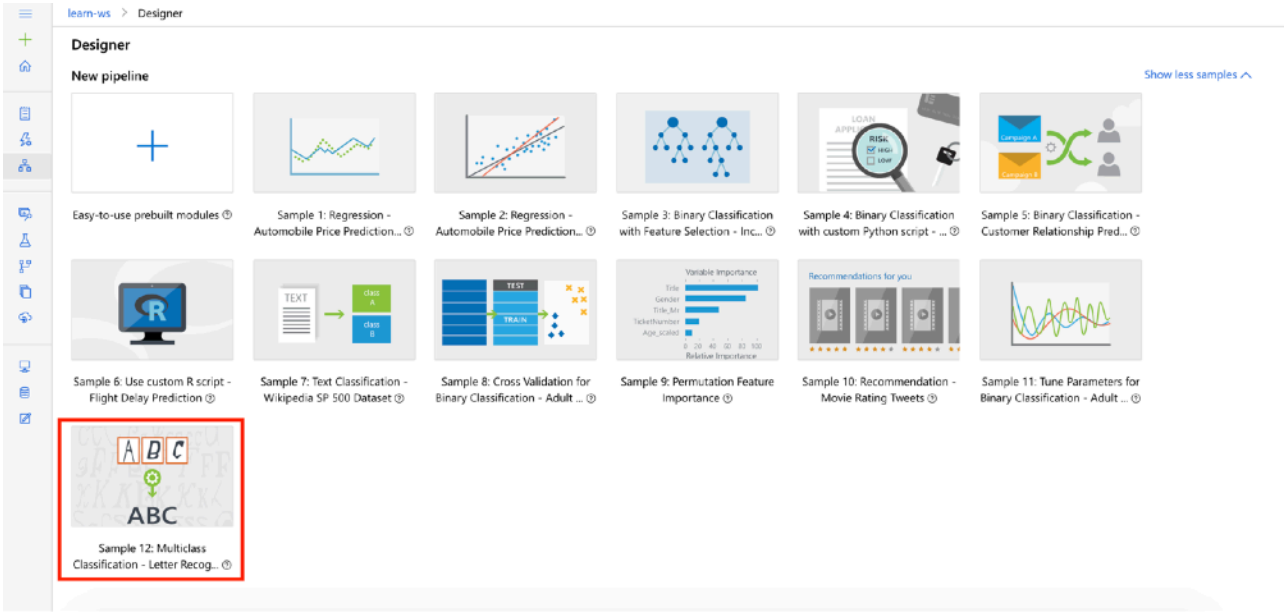


For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

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

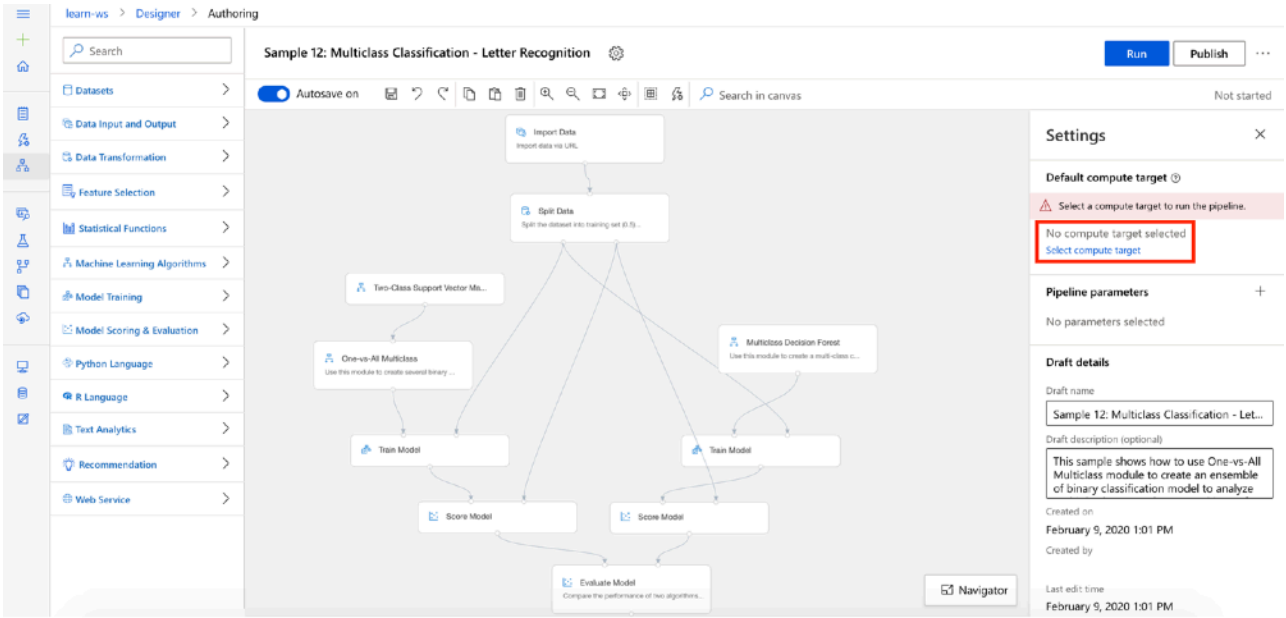


5. Select **Sample 12: Multiclass Classification - Letter Recognition**.



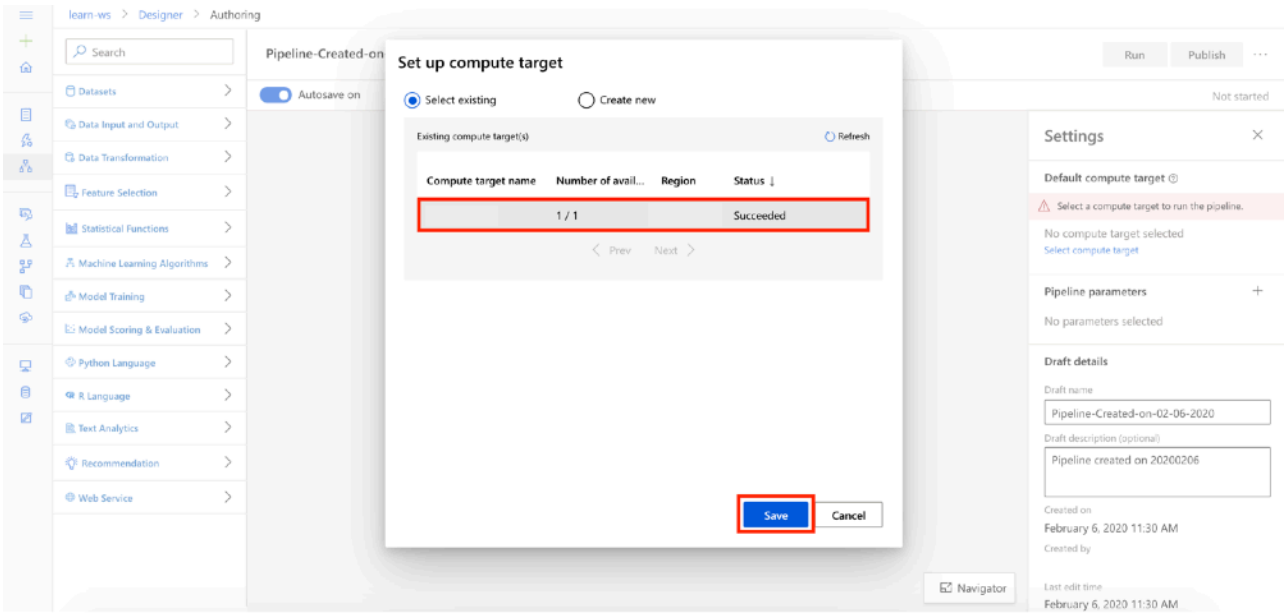
Task 2: Setup Compute Target

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**.

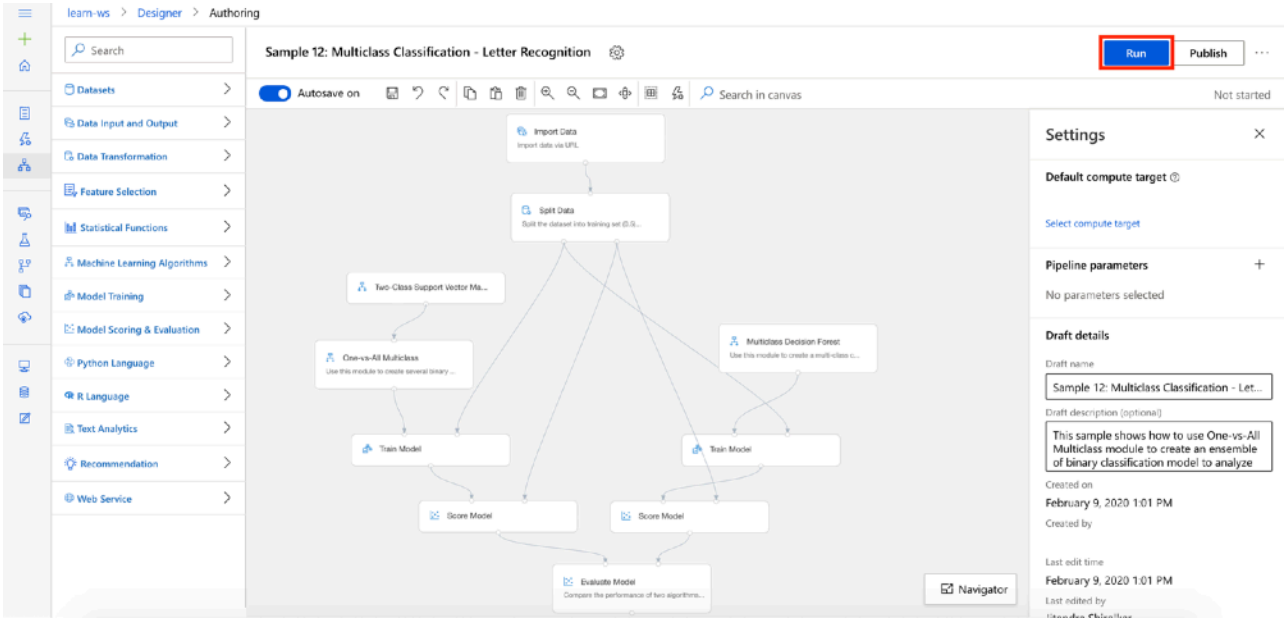
Note: If you are facing difficulties in accessing pop-up windows or buttons in the user interface, please refer to the Help section in the lab environment.



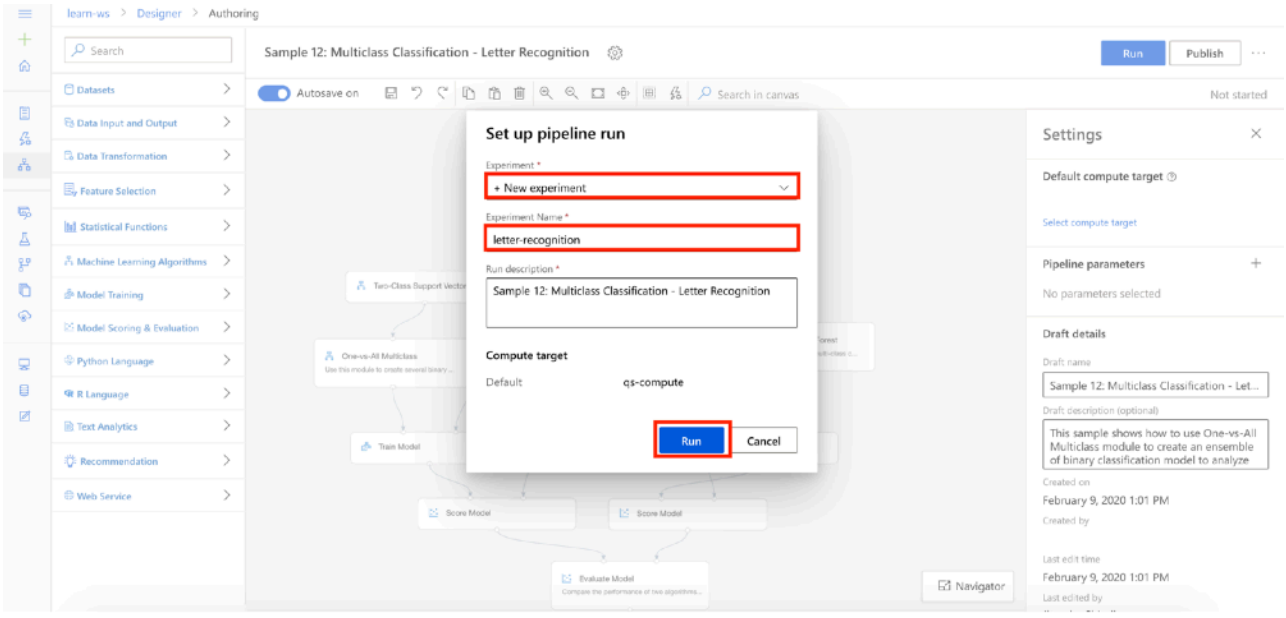
Exercise 2: Submit Training Pipeline

Task 1: Create Experiment and Submit Pipeline

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



2. Please note that the button name in the UI is changed from **Run** to **Submit**.
3. In the `Setup pipeline run editor`, select **Experiment, Create new** and provide `New experiment name:` **letter-recognition**, and then select **Submit**.

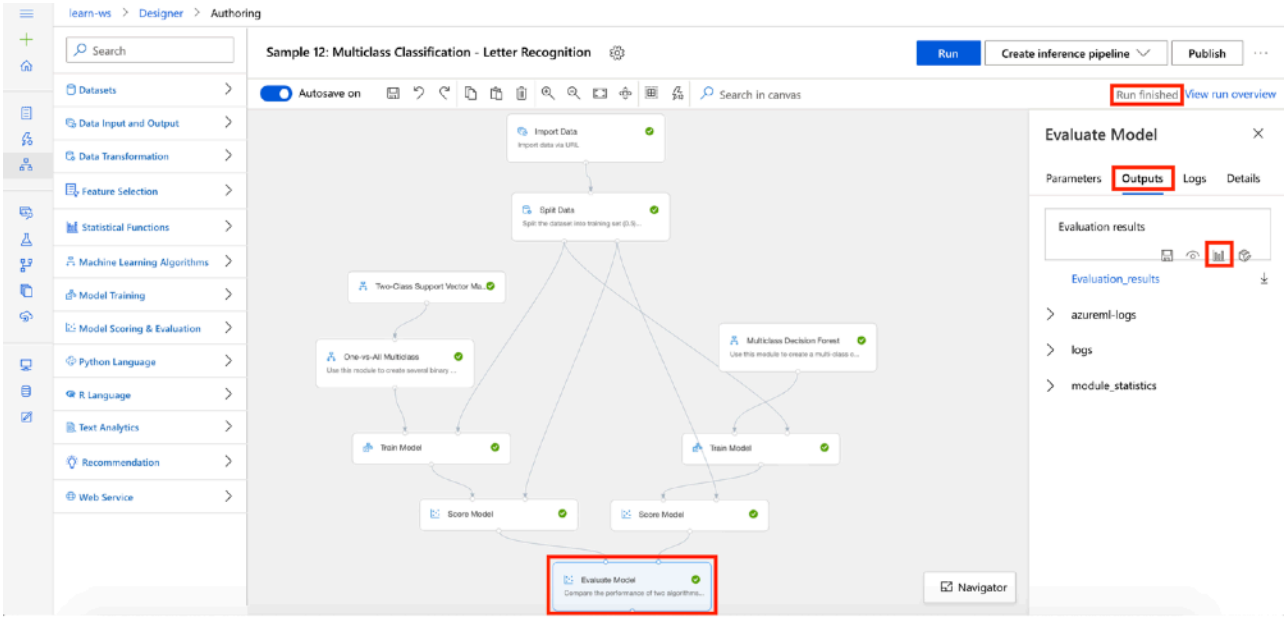


4. Wait for pipeline run to complete. It will take around **10 minutes** to complete the run.
5. While you wait for the model training to complete, you can learn more about the `One-vs-All Multiclass` module used in this lab by selecting One-vs-All Multiclass.

Exercise 3: Compare Model Performance

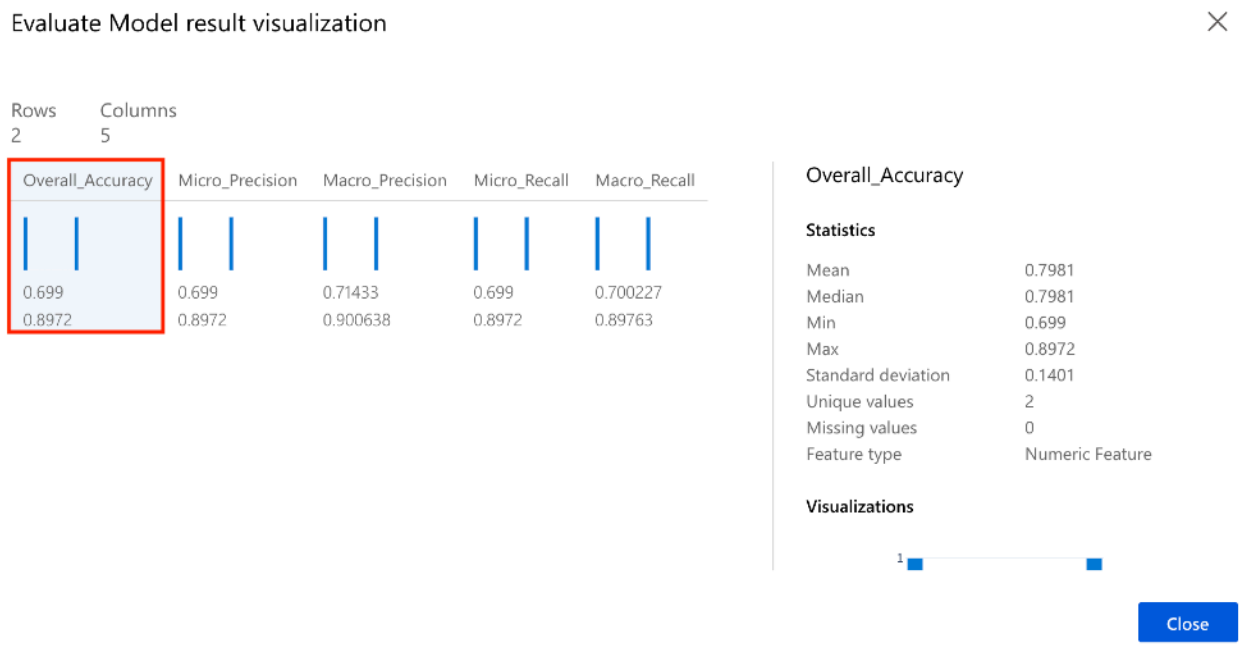
Task 1: Open Evaluation Results

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.

Lab 10 - Train a Classifier Using Automated Machine Learning

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:

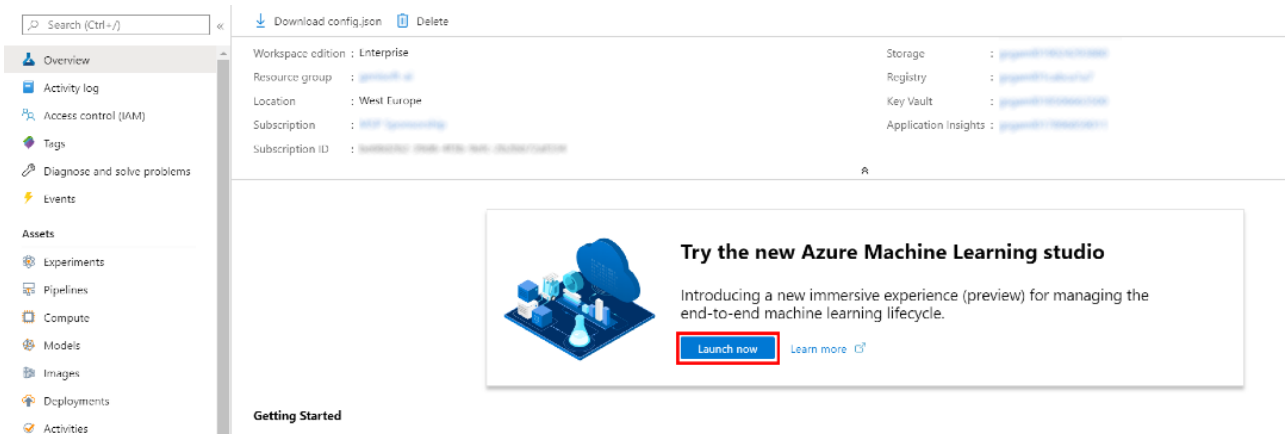
- Select your experiment type: Classification, Regression or Time Series Forecasting
- Data source, formats, and fetch data
- Choose your compute target
- Automated machine learning experiment settings
- Run an automated machine learning experiment
- Explore model metrics
- Register and deploy model

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.

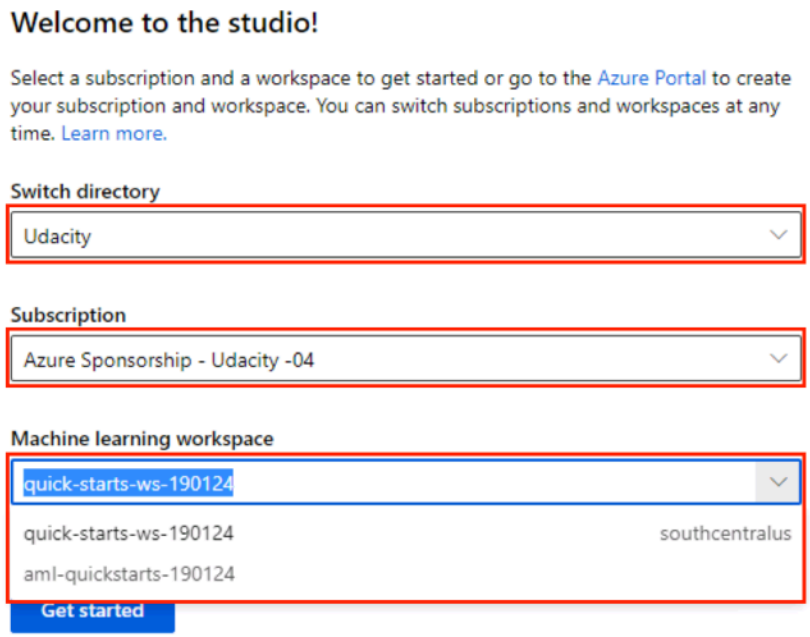
Exercise 1: Register Dataset with Azure Machine Learning studio

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.

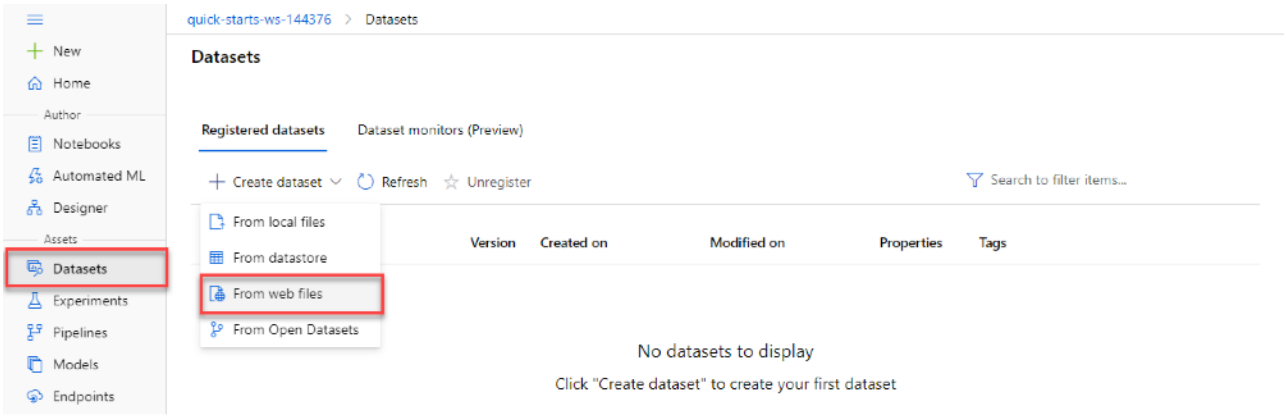


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



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 doesn't matter which) and then click **Get started**.

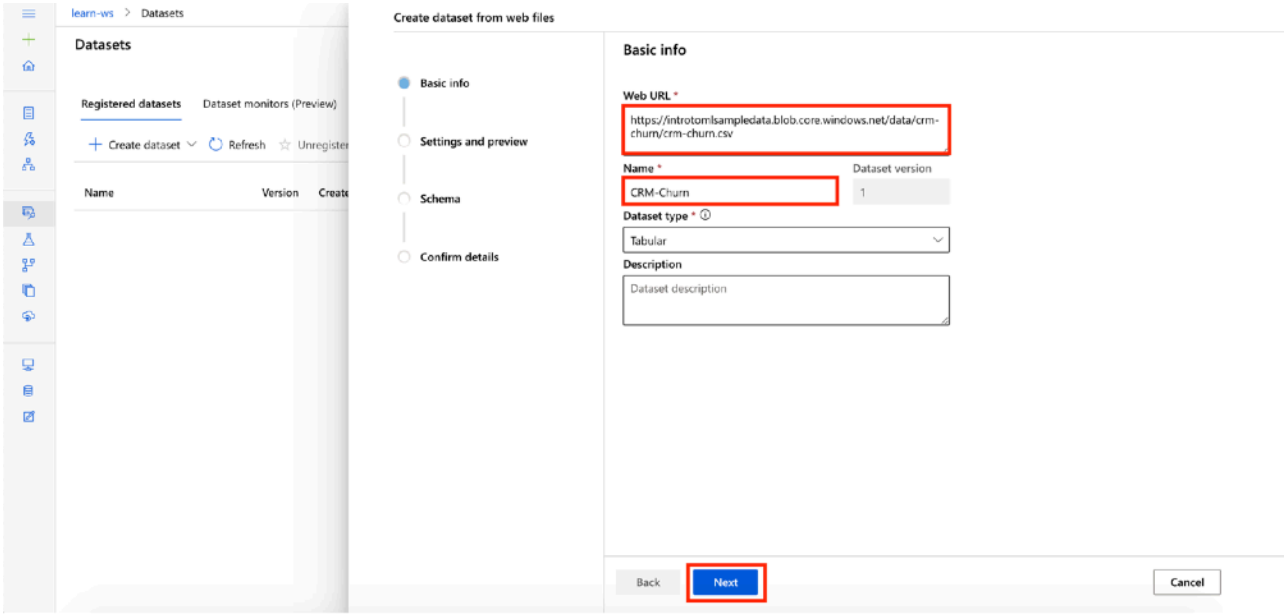
4. From the studio, select **Datasets, + Create dataset, From web files**. This will open the `Create dataset from web files` dialog on the right.



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

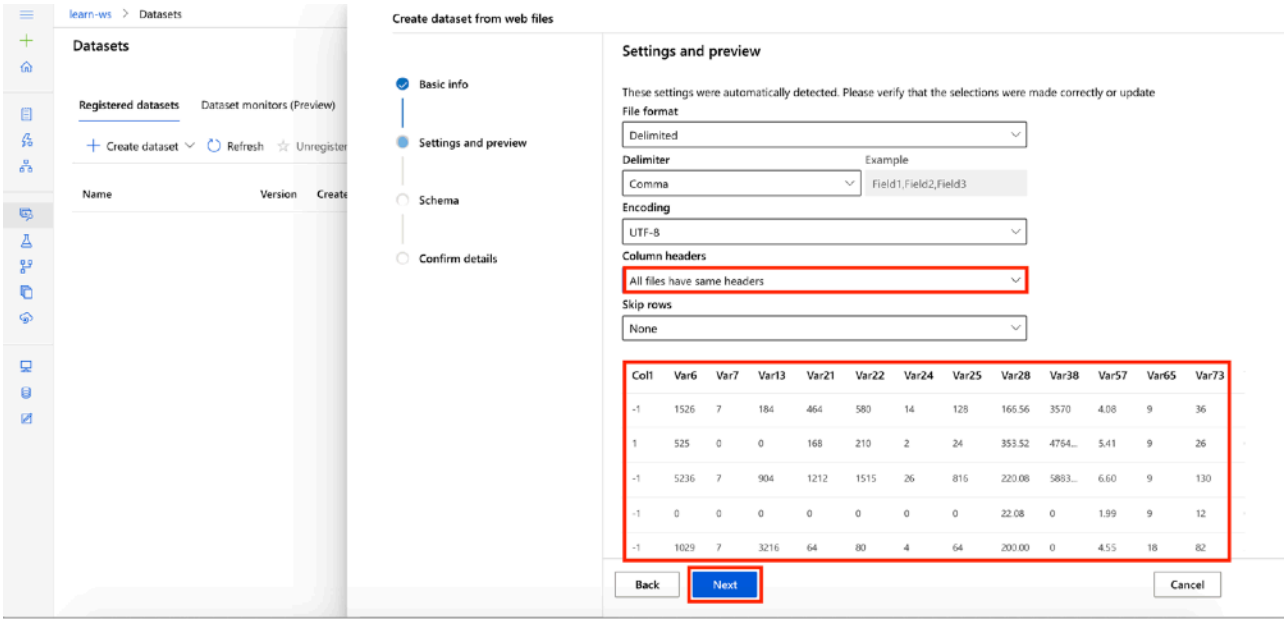
   `https://introtomlsampledata.blob.core.windows.net/data/crm-churn/crm-churn.csv`

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



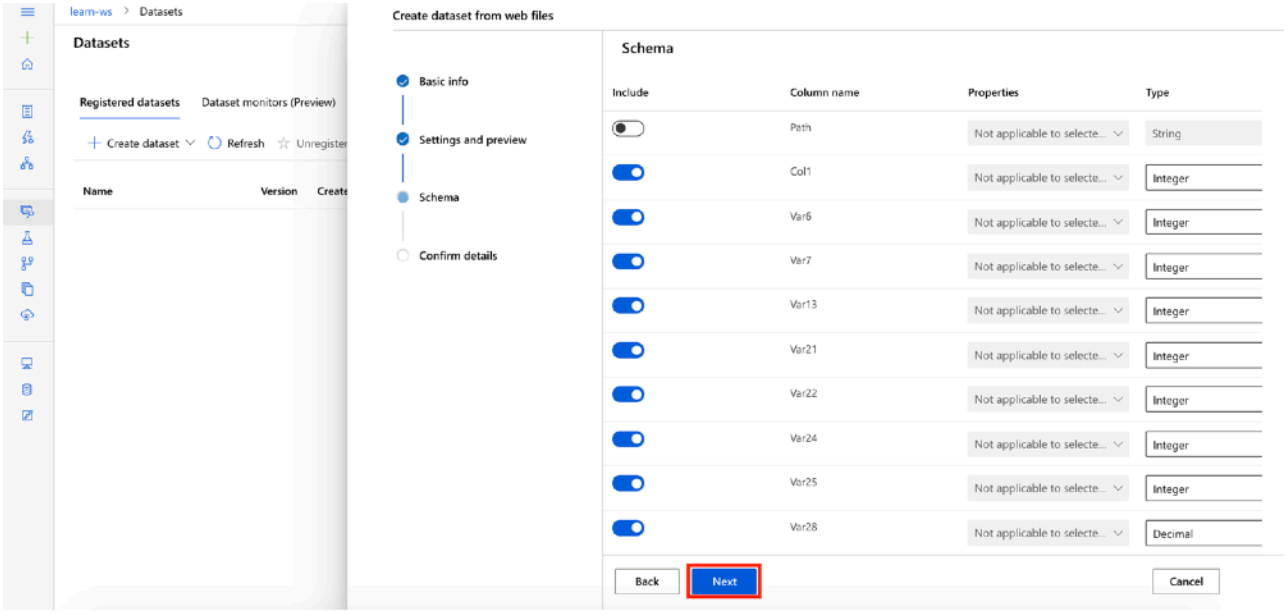
Task 2: Preview Dataset

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**



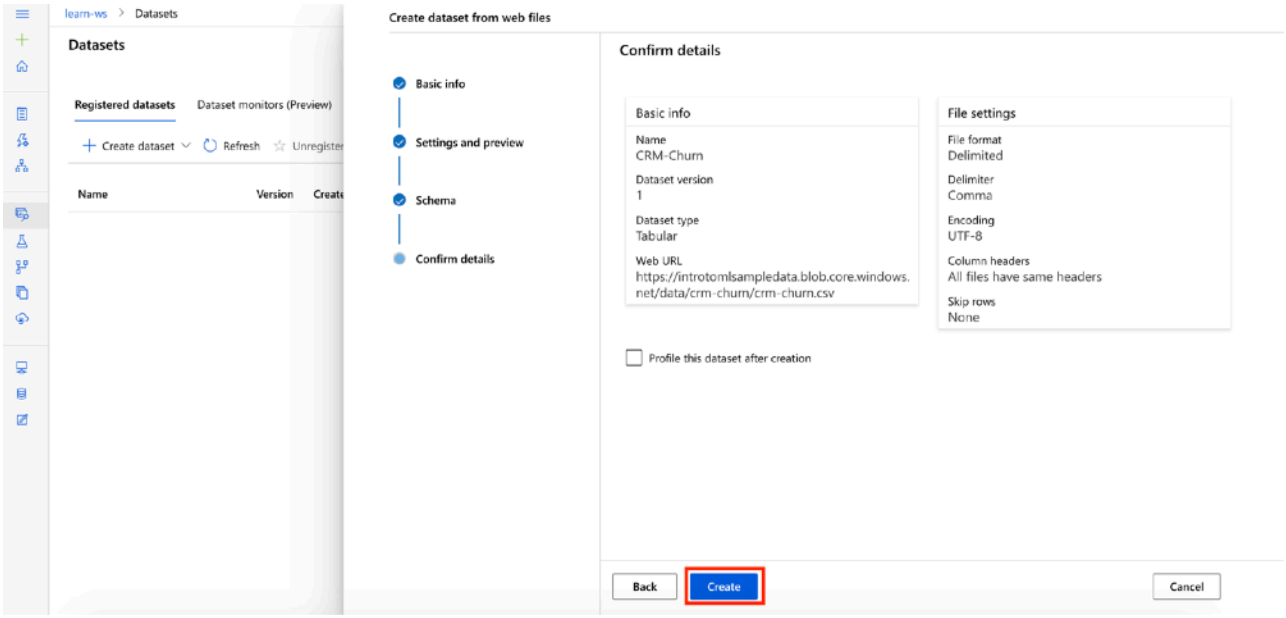
Task 3: Select Columns

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



Task 4: Create Dataset

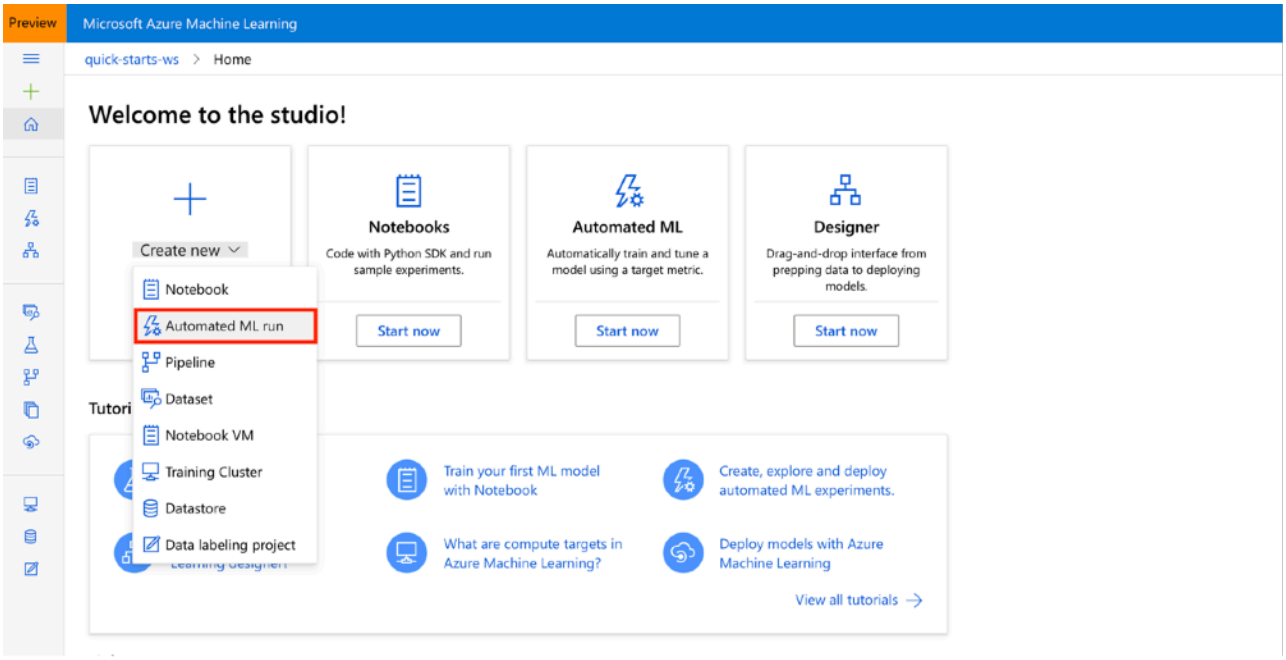
1. Confirm the dataset details and select **Create**



Exercise 2: Setup New Automated Machine Learning Experiment

Task 1: Create New Automated Machine Learning Experiment

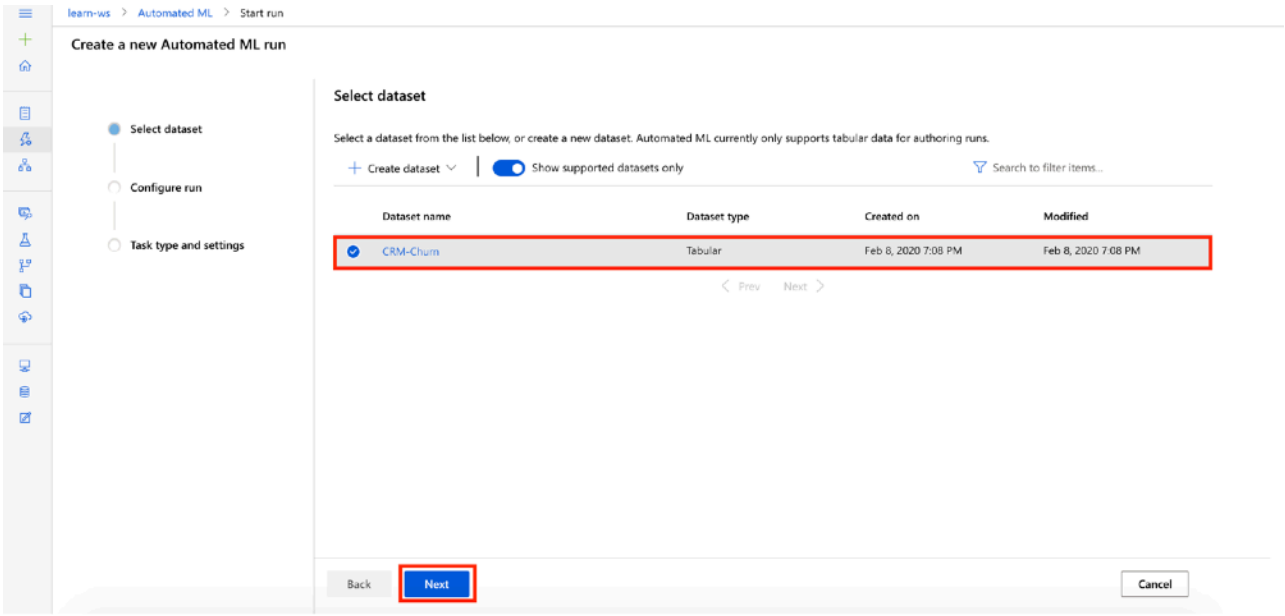
1. From the studio home, select **Create new, Automated ML run**



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

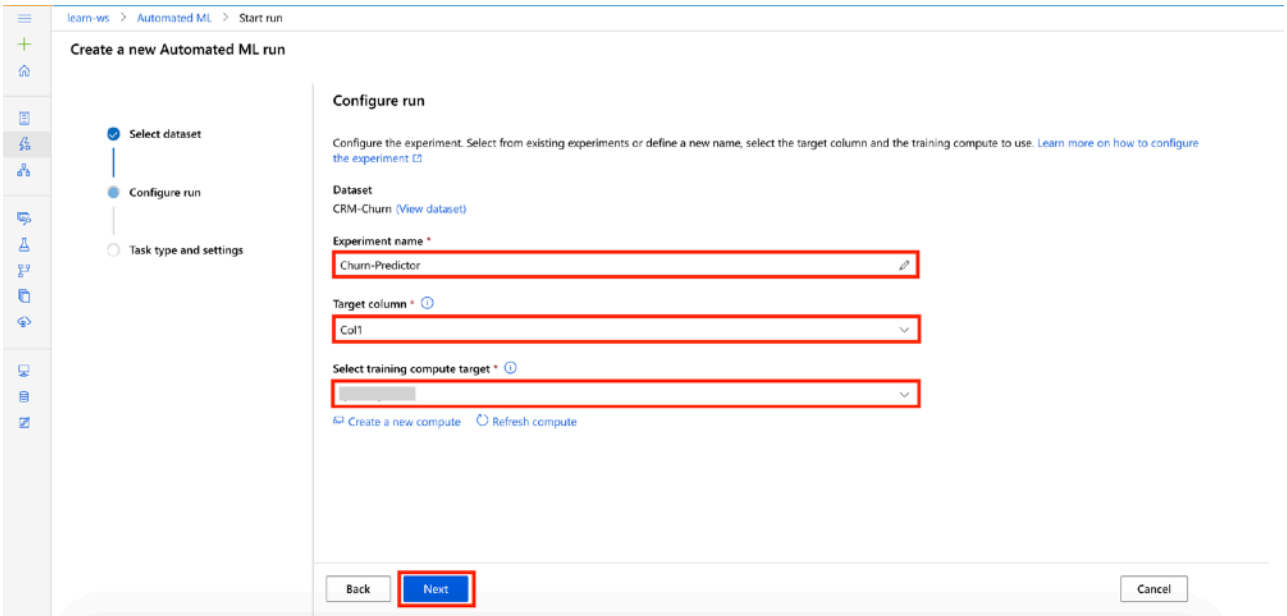
Task 2: Select Training Data

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



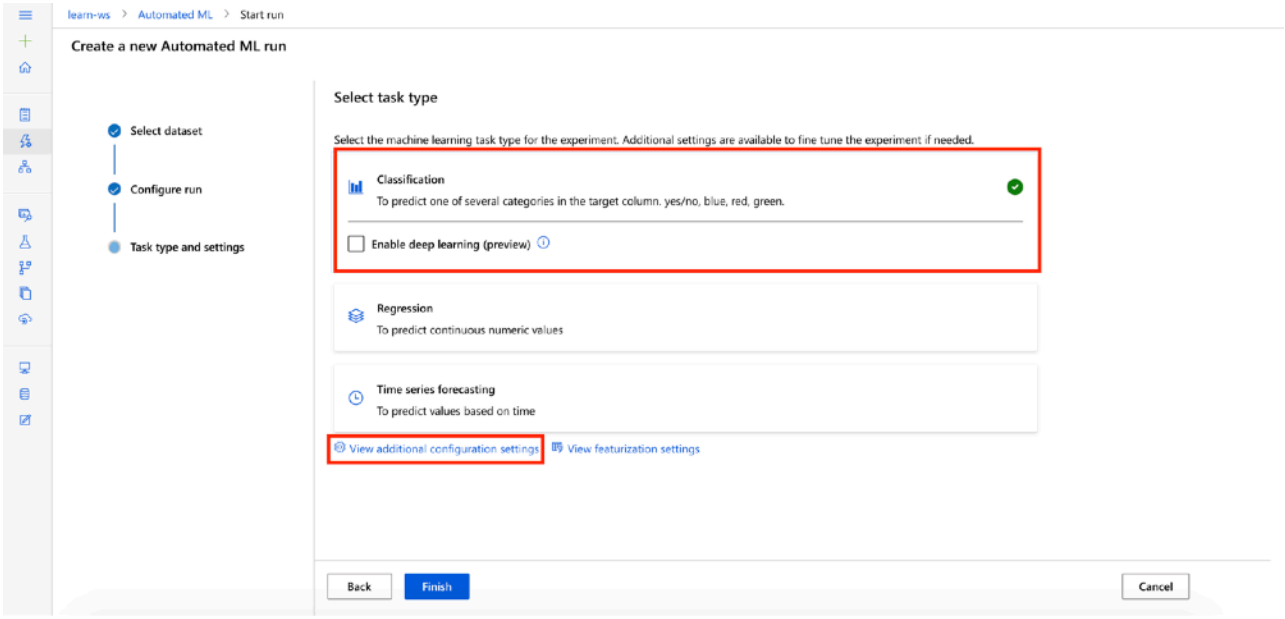
Task 3: Create a new Automated ML run

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

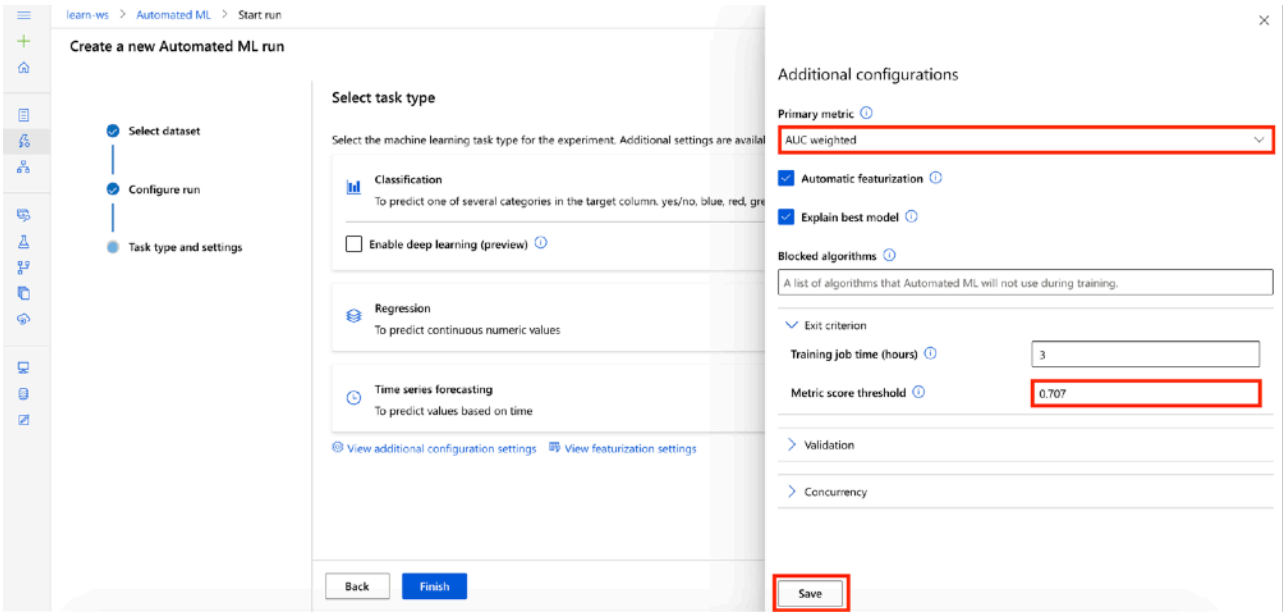


Task 4: Setup Task type and Settings

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`

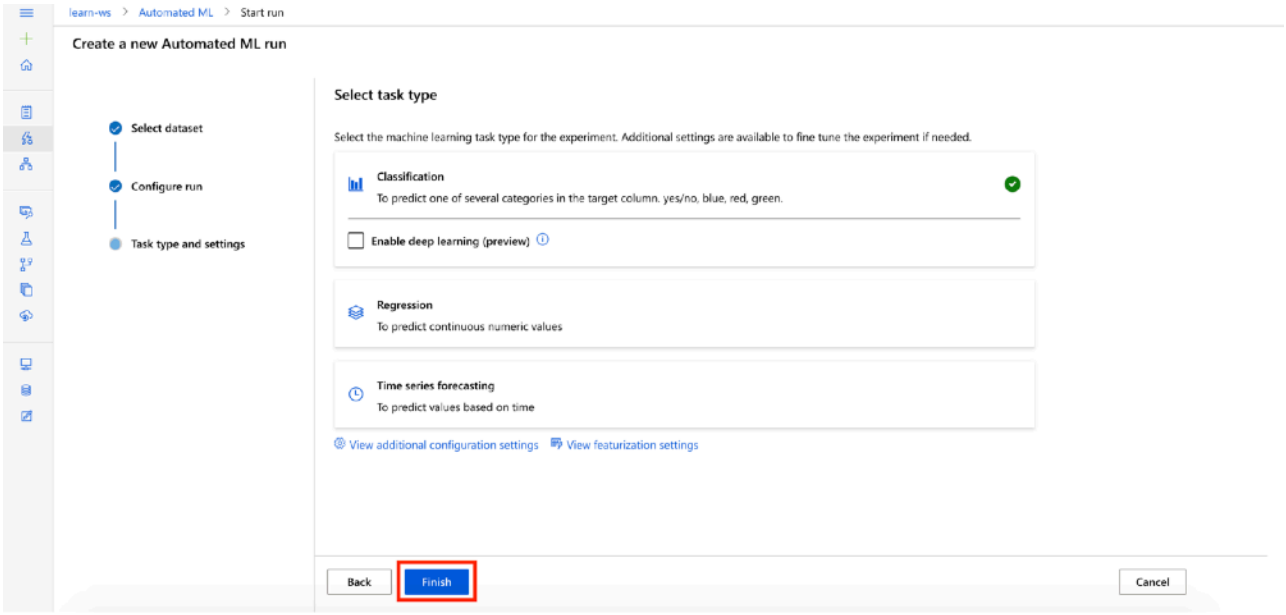


4. Note that we are setting a metric score threshold to limit the training time. In practice, for initial experiments, you will typically only set the training job time to allow AutoML to discover the best algorithm to use for your specific data.

Exercise 3: Start and Monitor Experiment

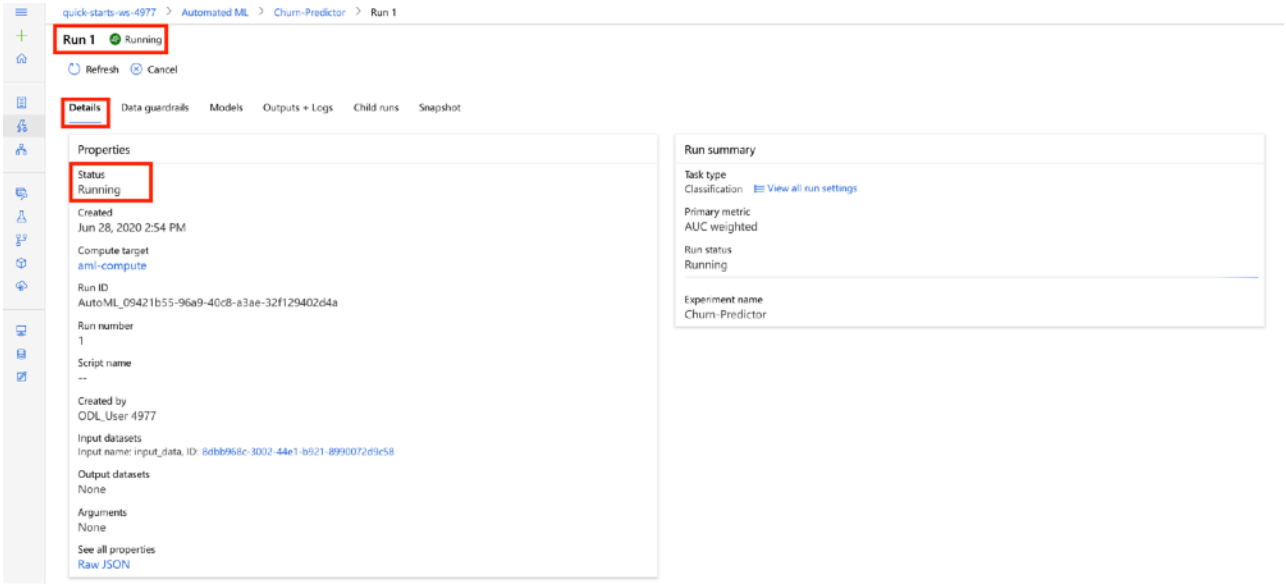
Task 1: Start Experiment

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

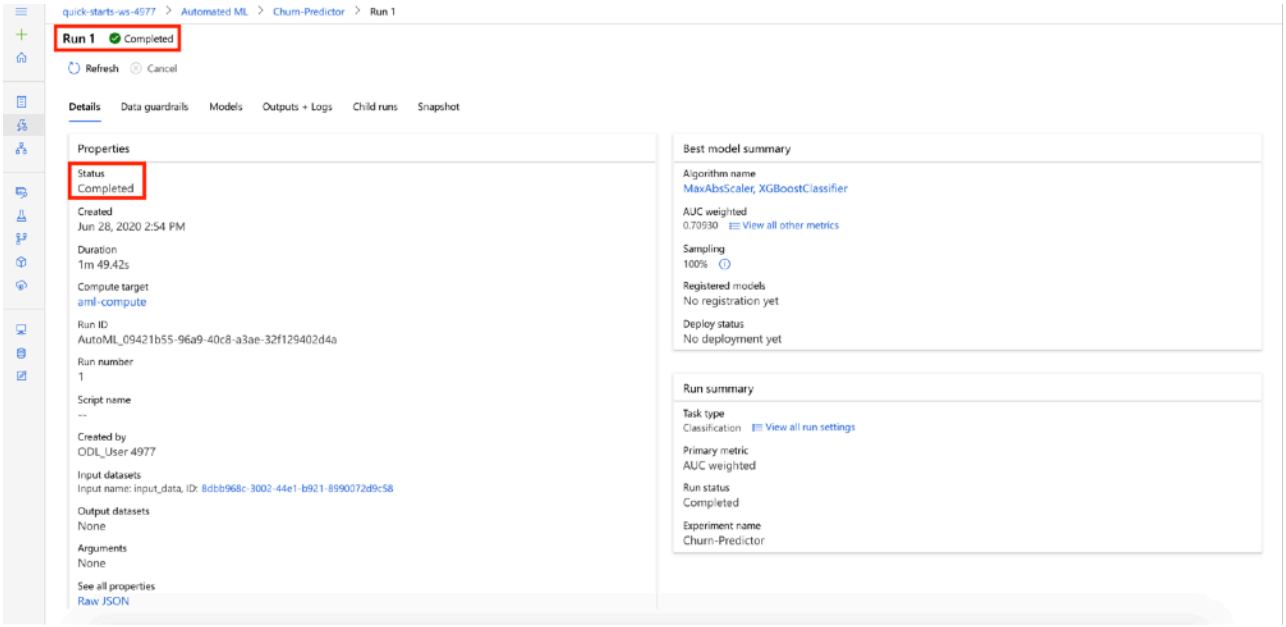


Task 2: Monitor Experiment

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**.

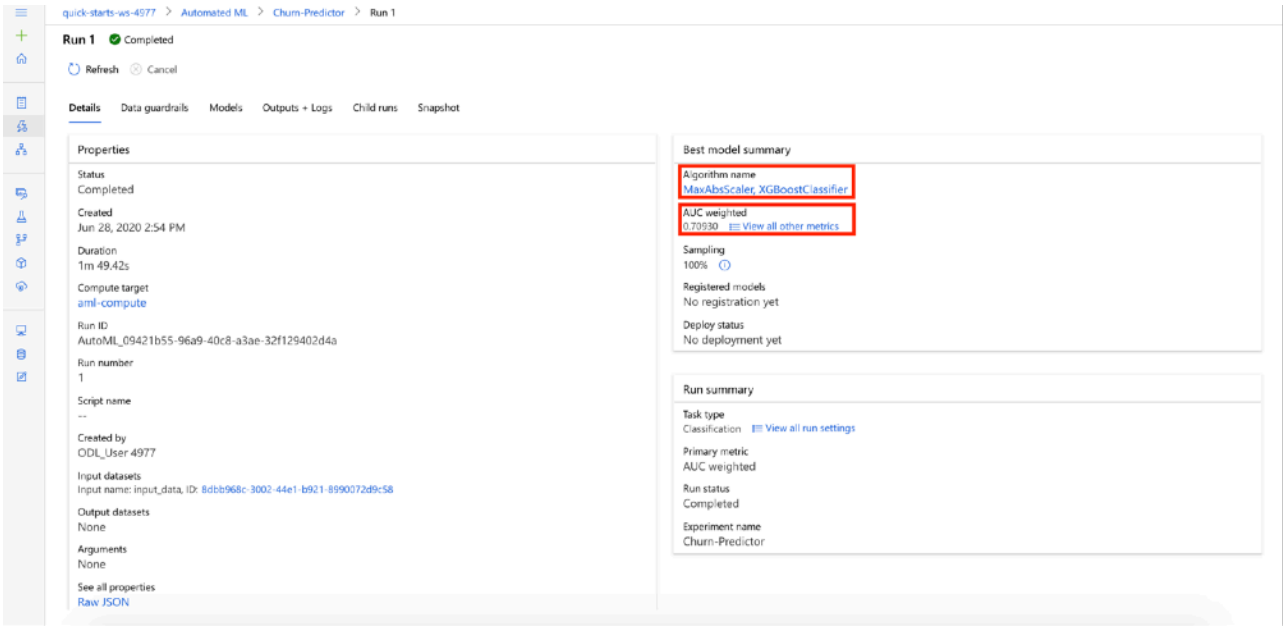


4. While you wait for the model training to complete, you can learn to view and understand the charts and metrics for your automated machine learning run by selecting Understand automated machine learning classification results.

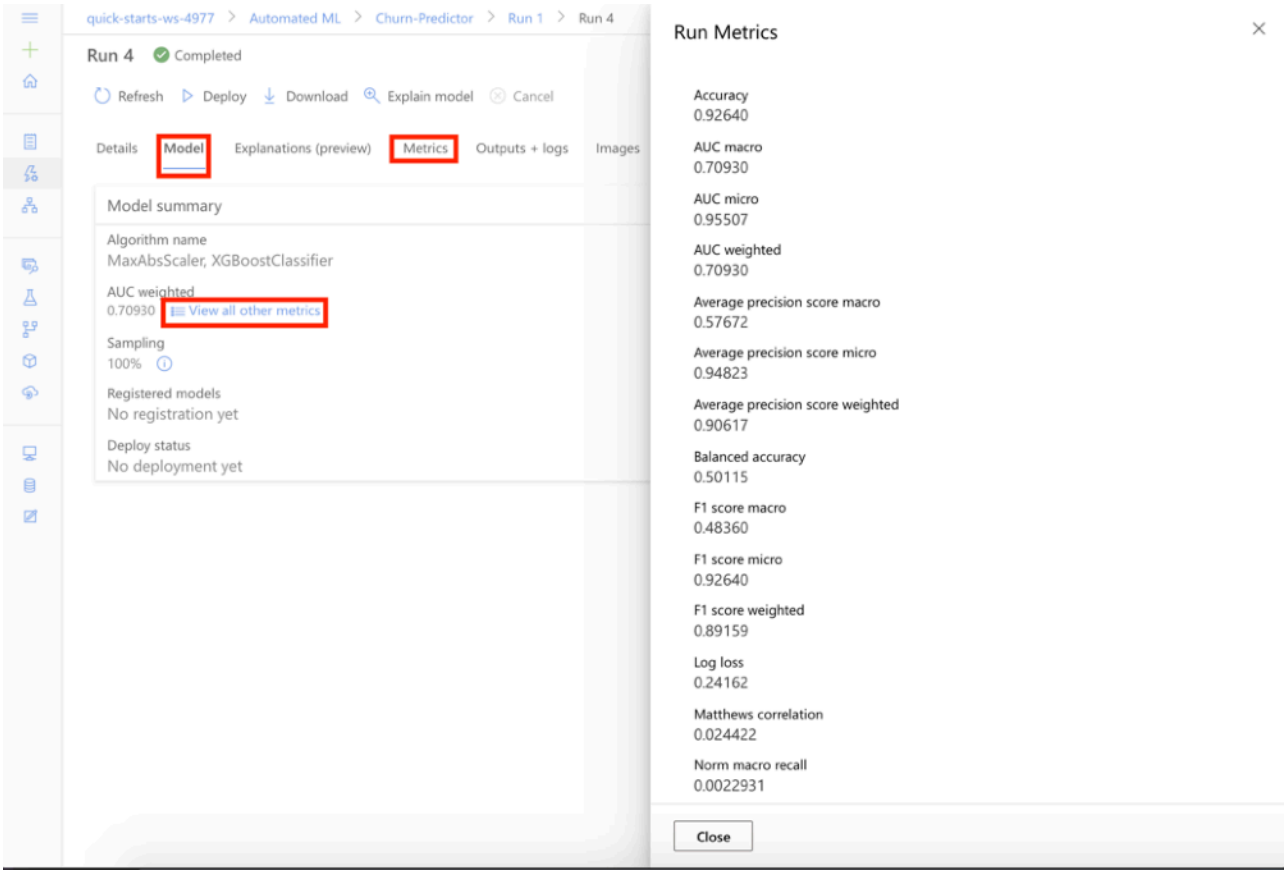
Exercise 4: Review Best Model's Performance

Task 1: Review Best Model Performance

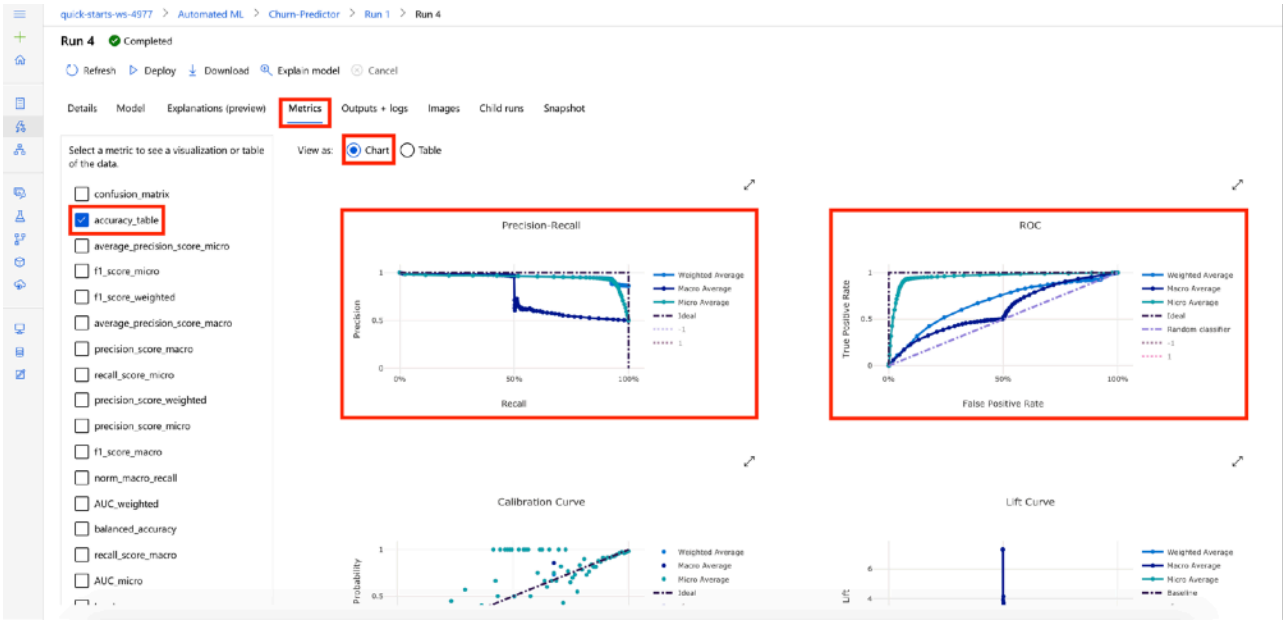
1. From the `Details` tab review the best model's `Algorithm name` and its corresponding `AUC weighted` score. Next, select the best model's **Algorithm name**



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.

Lab 11 - Regressors Performance

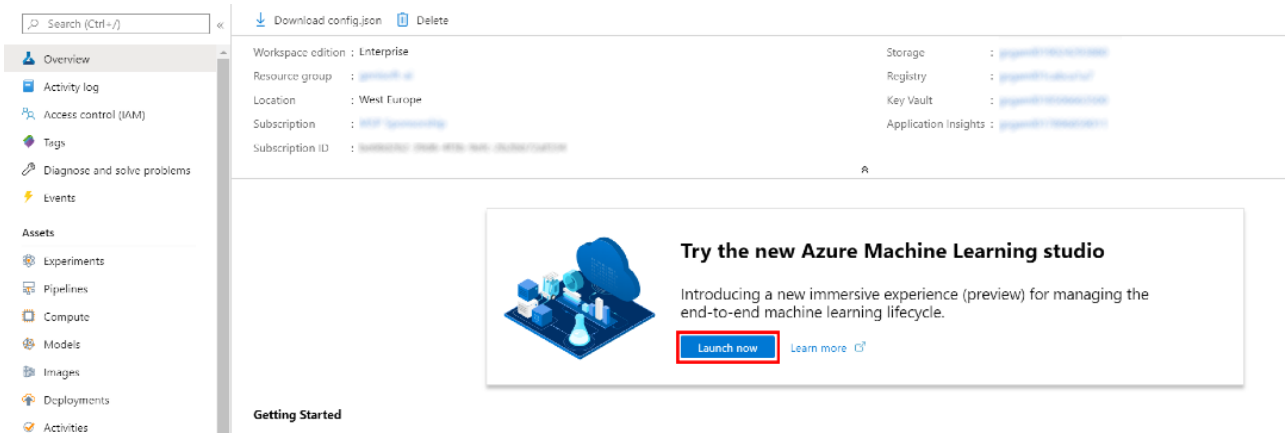
Overview

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.

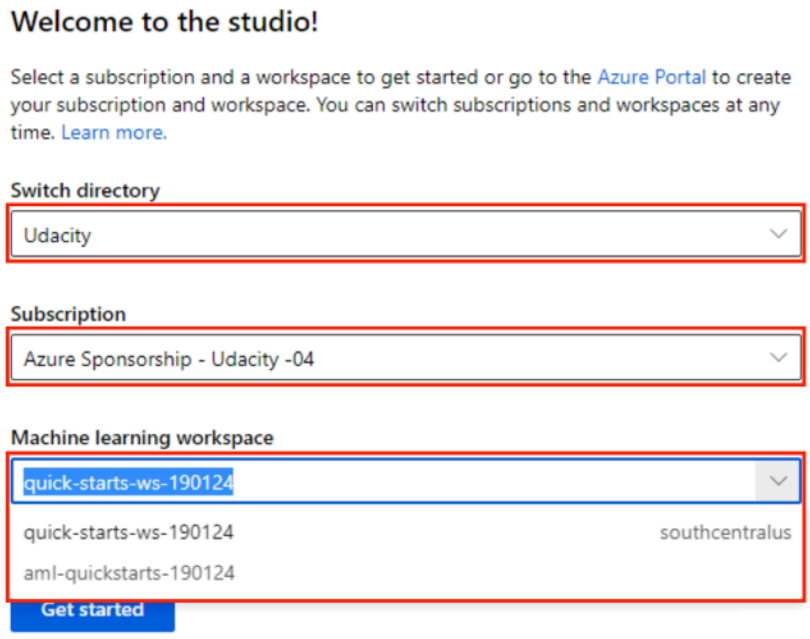
Exercise 1: Create Training Pipeline

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.

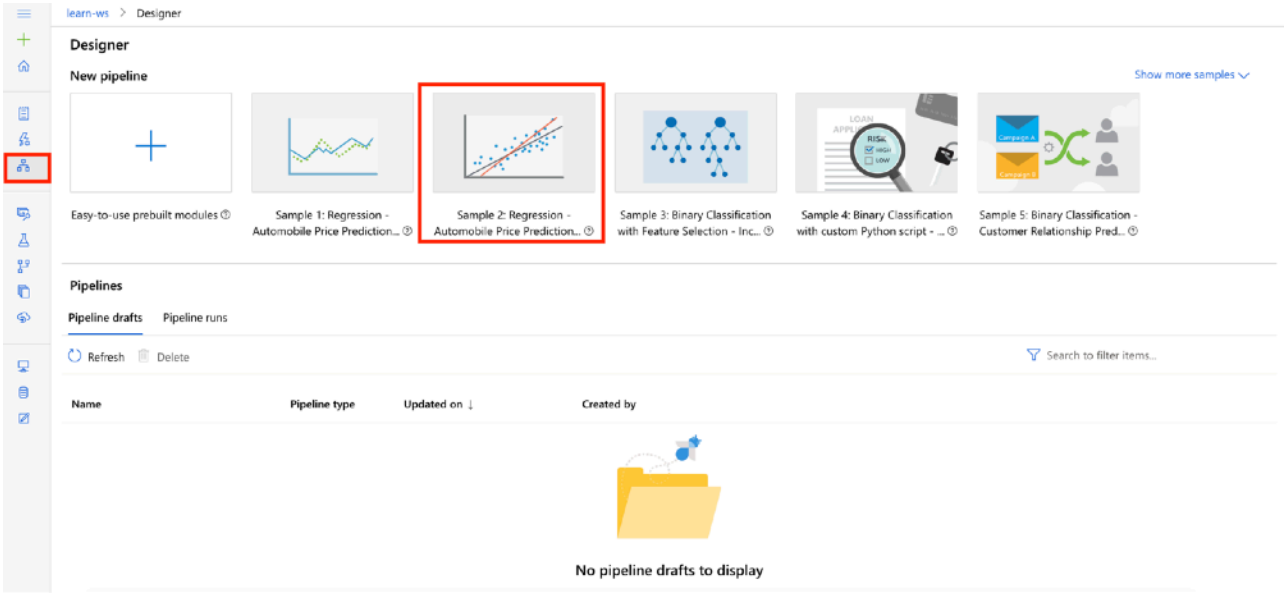


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



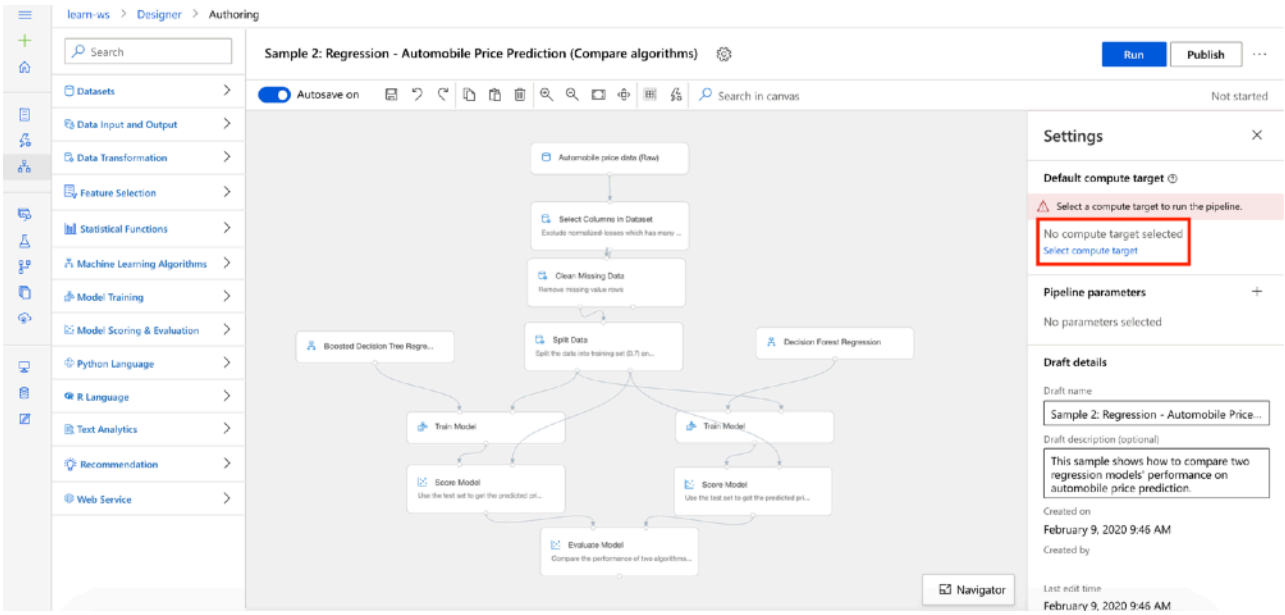
For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

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



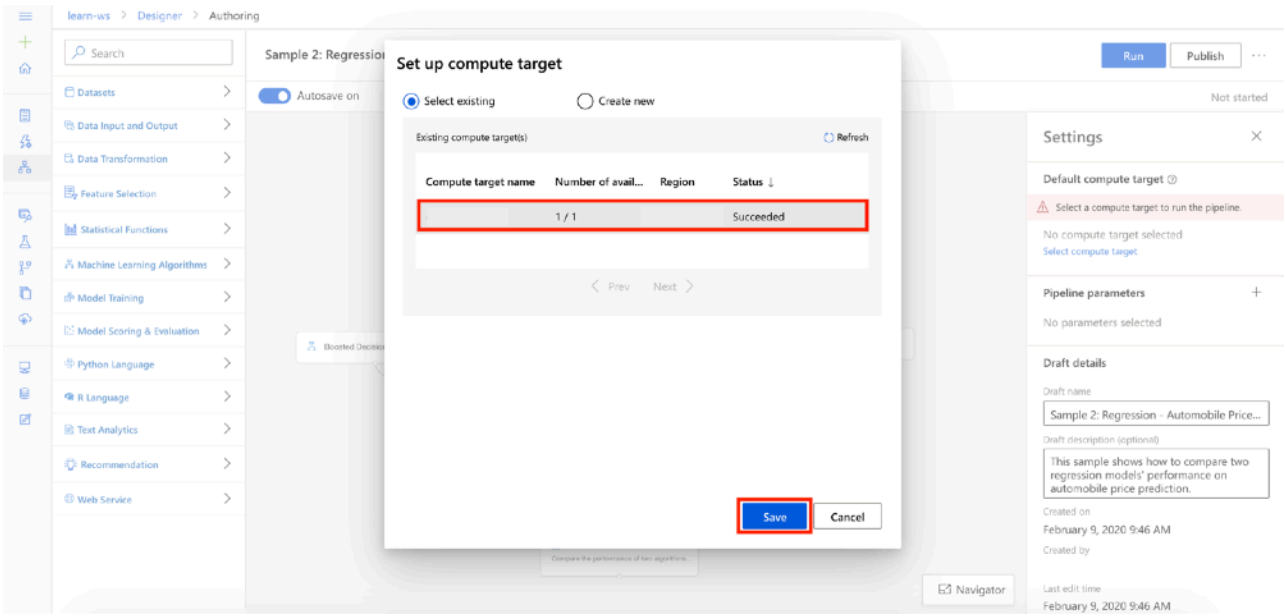
Task 2: Setup Compute Target

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



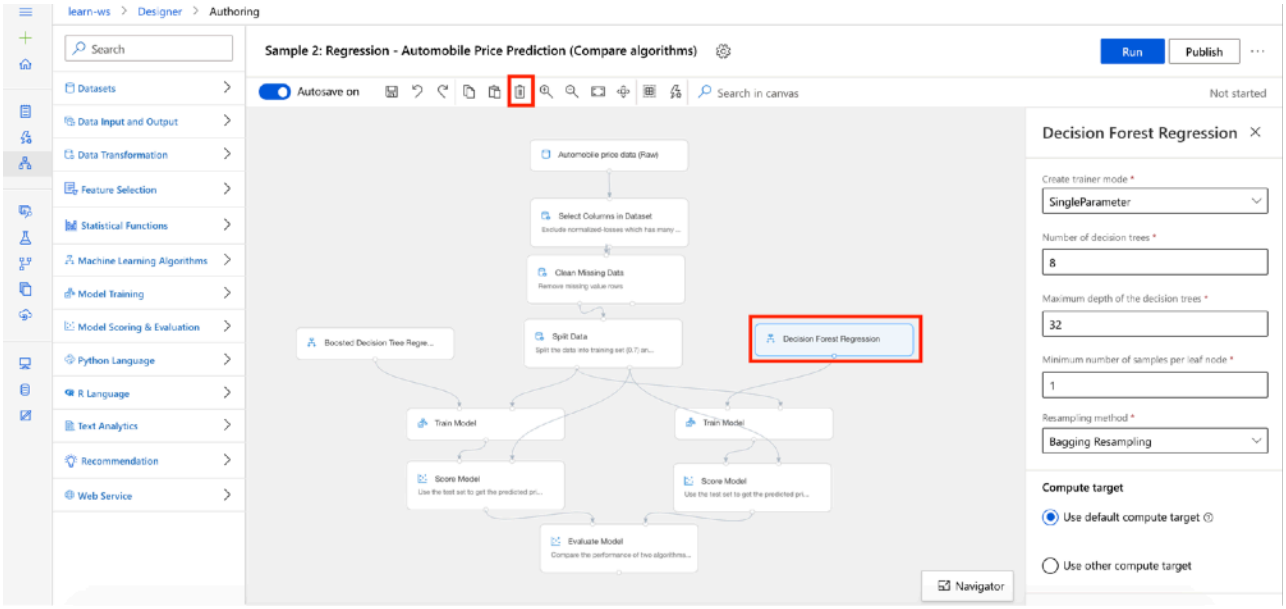
2. In the `Set up compute target` editor, select the available compute, and then select **Save**.

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

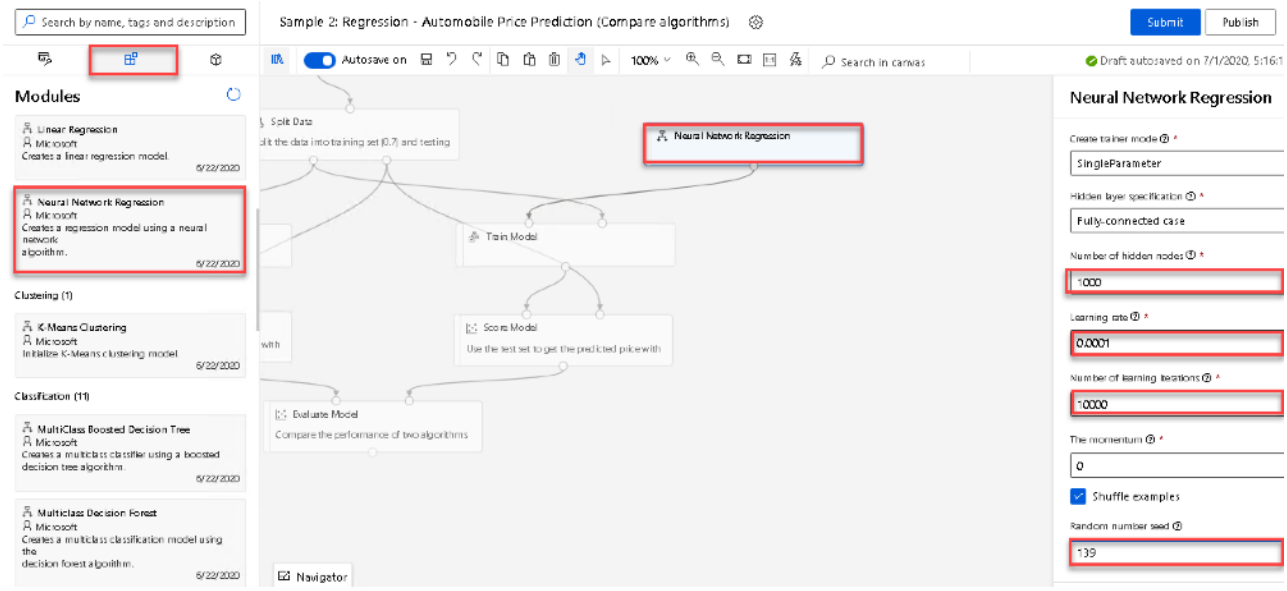
1. From the `right-hand-side` of the pipeline, select the **Decision Forest Regression module** and then select the **Delete Icon**.



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**

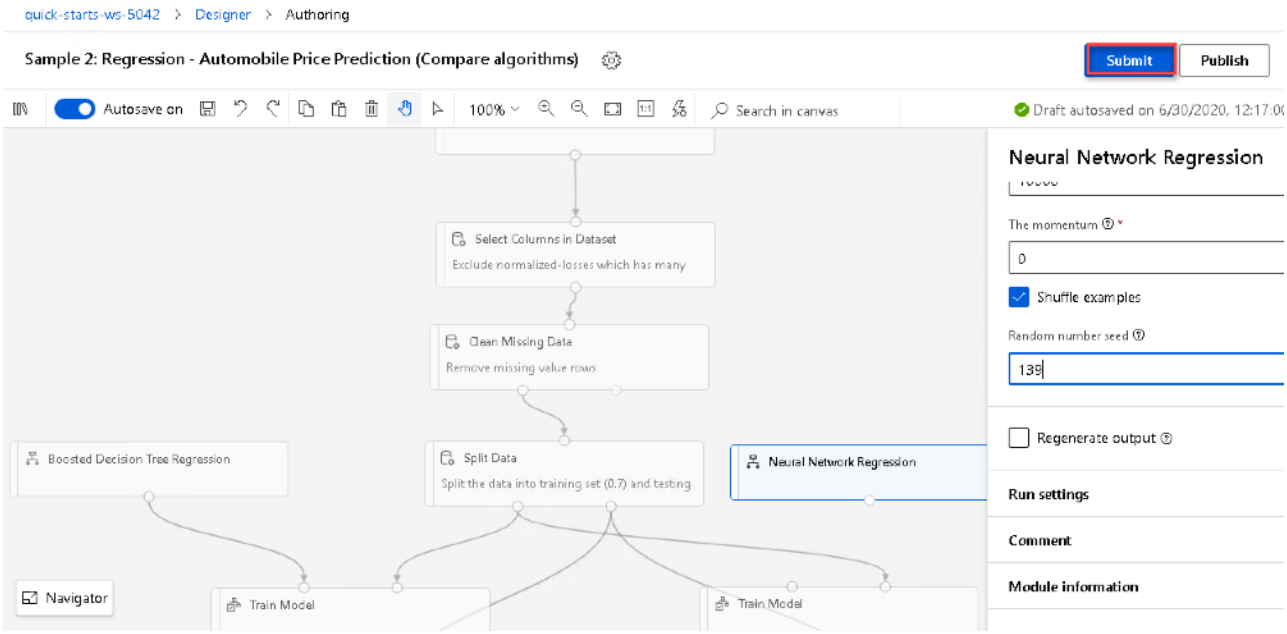
7. Connect the `Neural Net Regression` module to the first input of the `Train Model` module



Exercise 2: Submit Training Pipeline

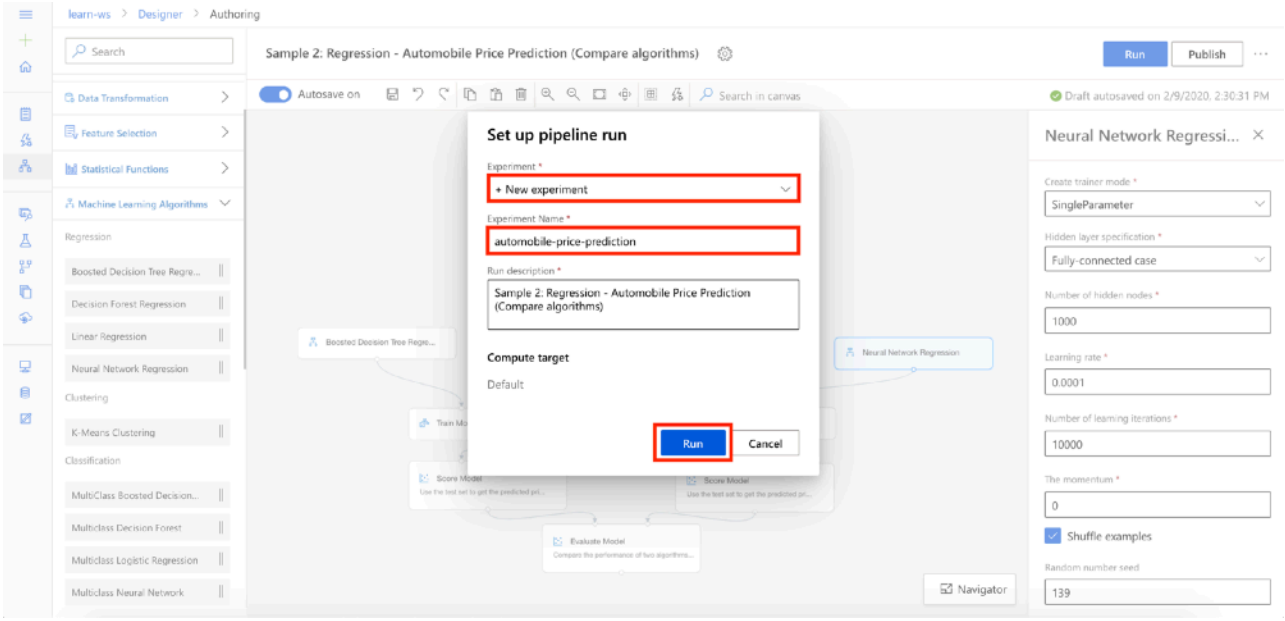
Task 1: Create Experiment and Submit Pipeline

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



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

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

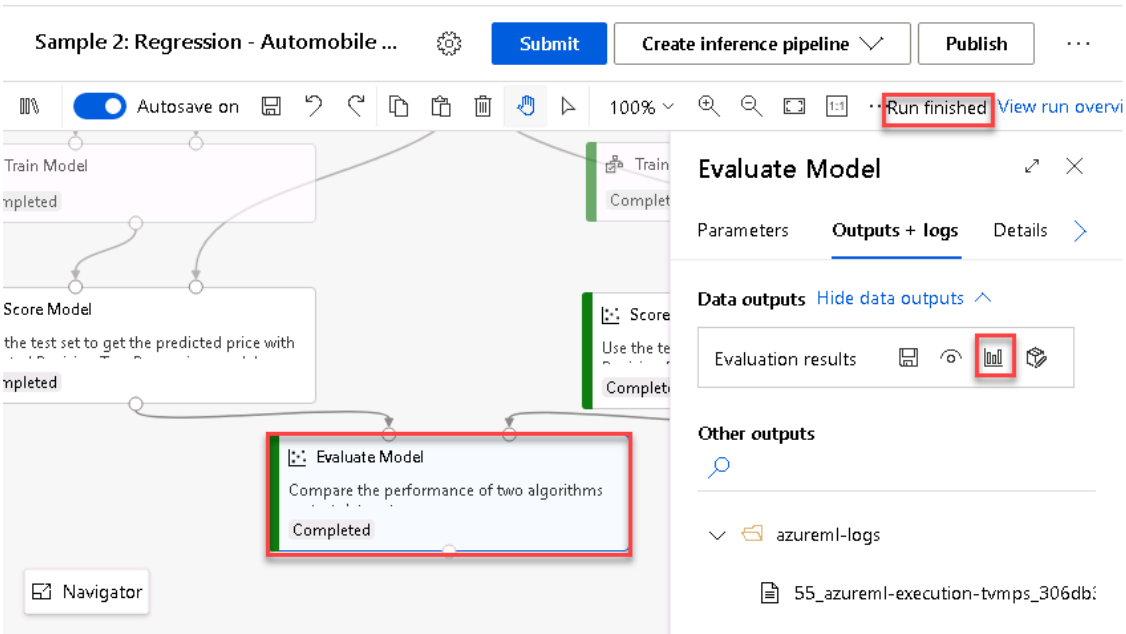


4. Wait for pipeline run to complete. It will take around **10 minutes** to complete the run.
5. While you wait for the model training to complete, you can learn more about the evaluation metrics for the regression algorithm used in this lab by selecting Metrics for regression models.

Exercise 3: Compare Model Performance

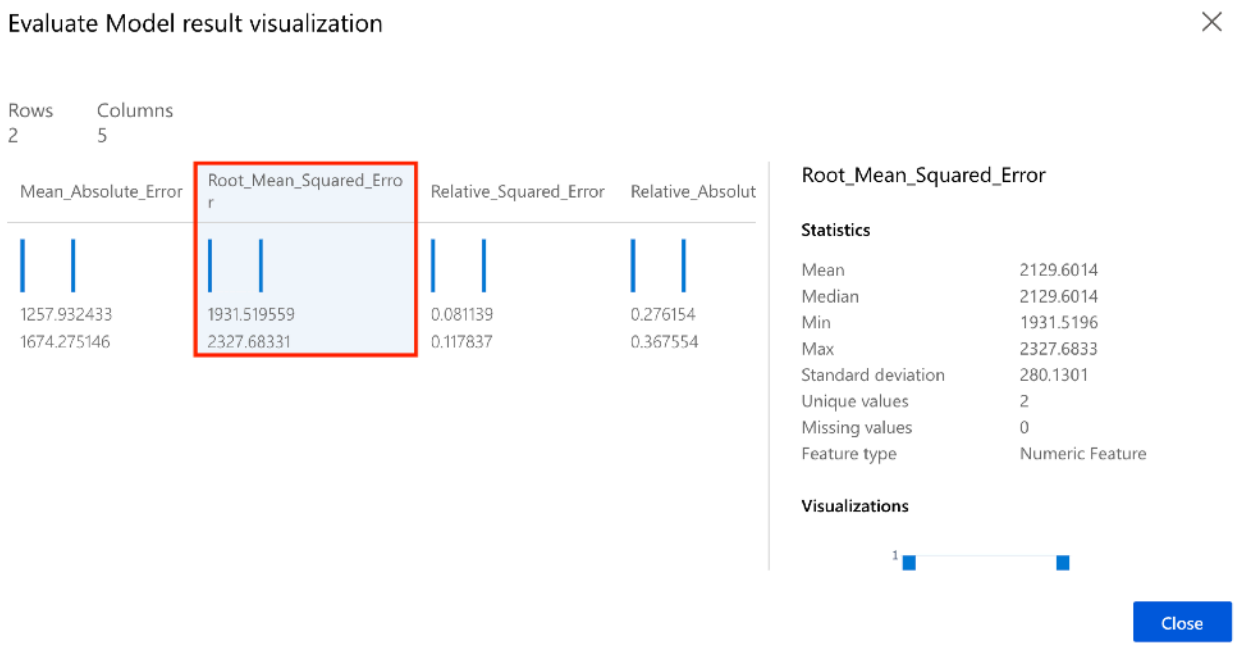
Task 1: Open Evaluation Results

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.

Lab 12 - Train a Regressor using Automated Machine Learning

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:

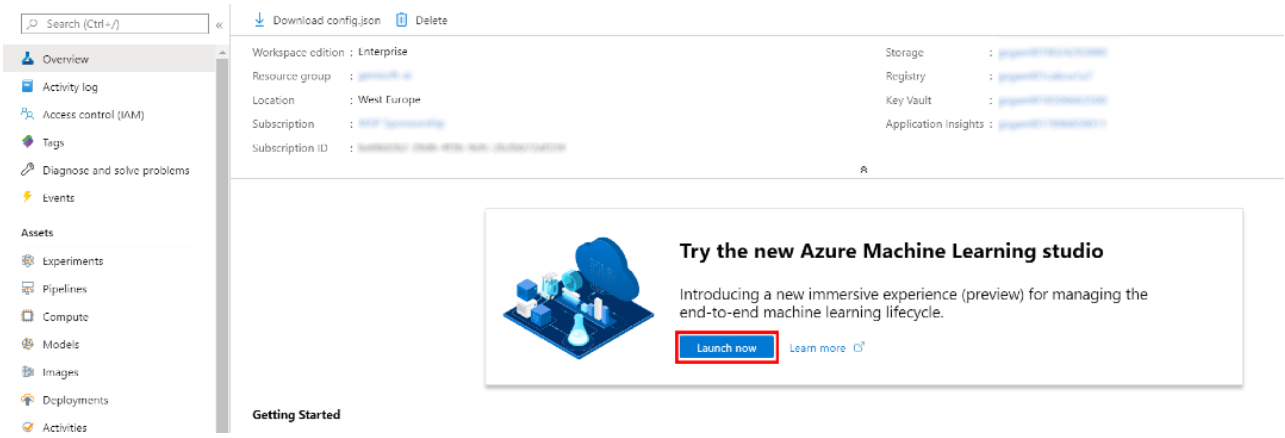
- Select your experiment type: Classification, Regression or Time Series Forecasting
- Data source, formats, and fetch data
- Choose your compute target
- Automated machine learning experiment settings
- Run an automated machine learning experiment
- Explore model metrics
- Register and deploy model

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.

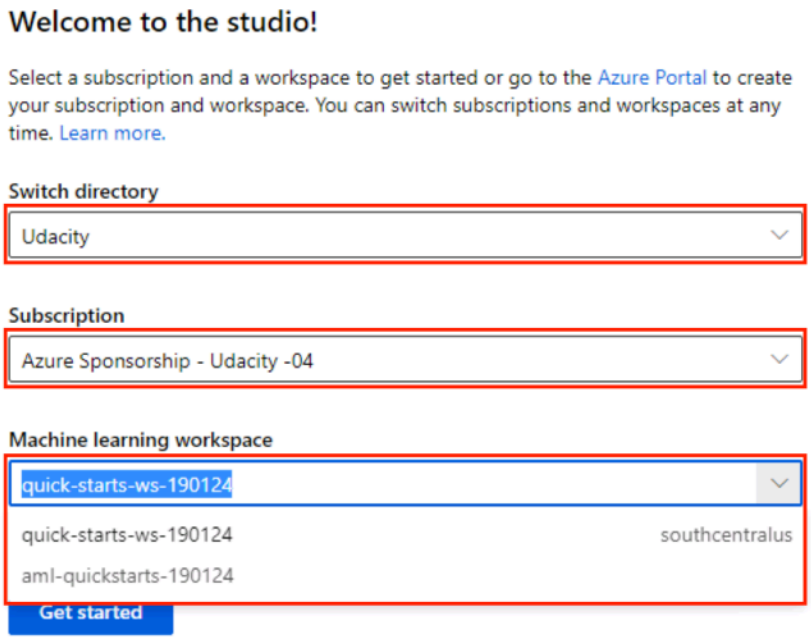
Exercise 1: Register Dataset with Azure Machine Learning studio

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.



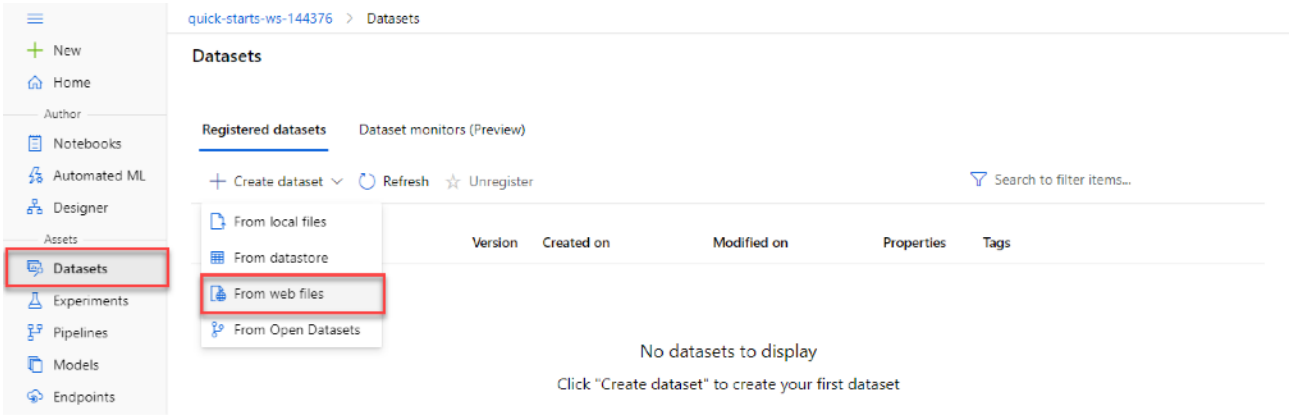
3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



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**.

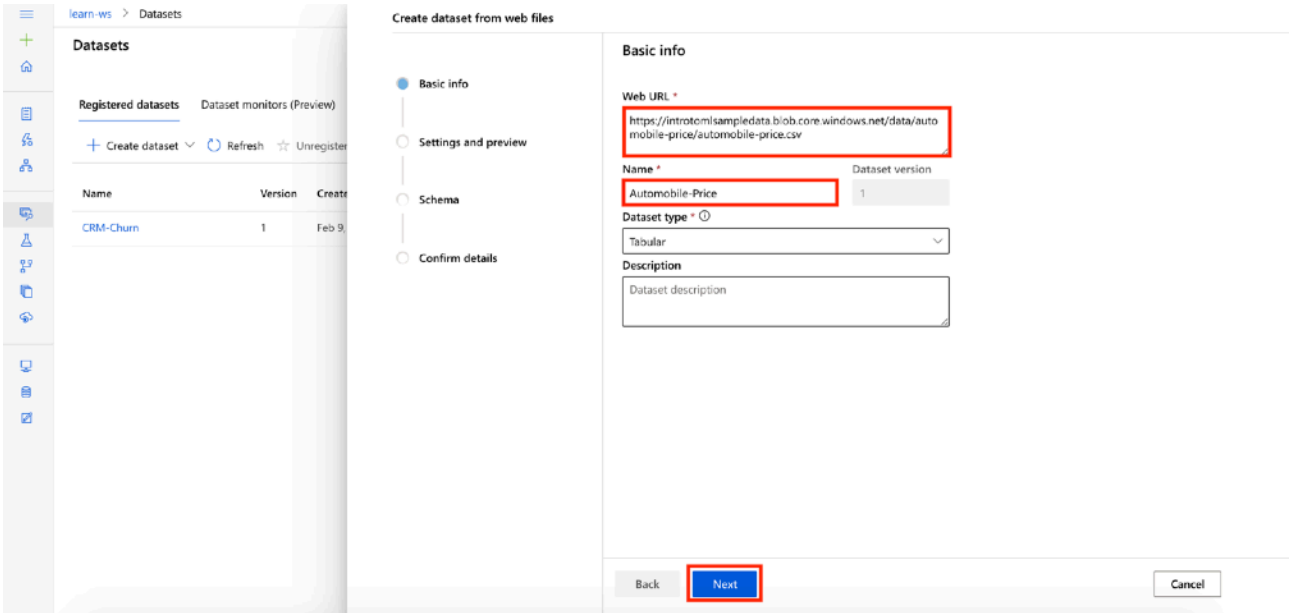
4. From the studio, select **Datasets, + Create dataset, From web files**. This will open the `Create dataset from web files` dialog on the right.



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

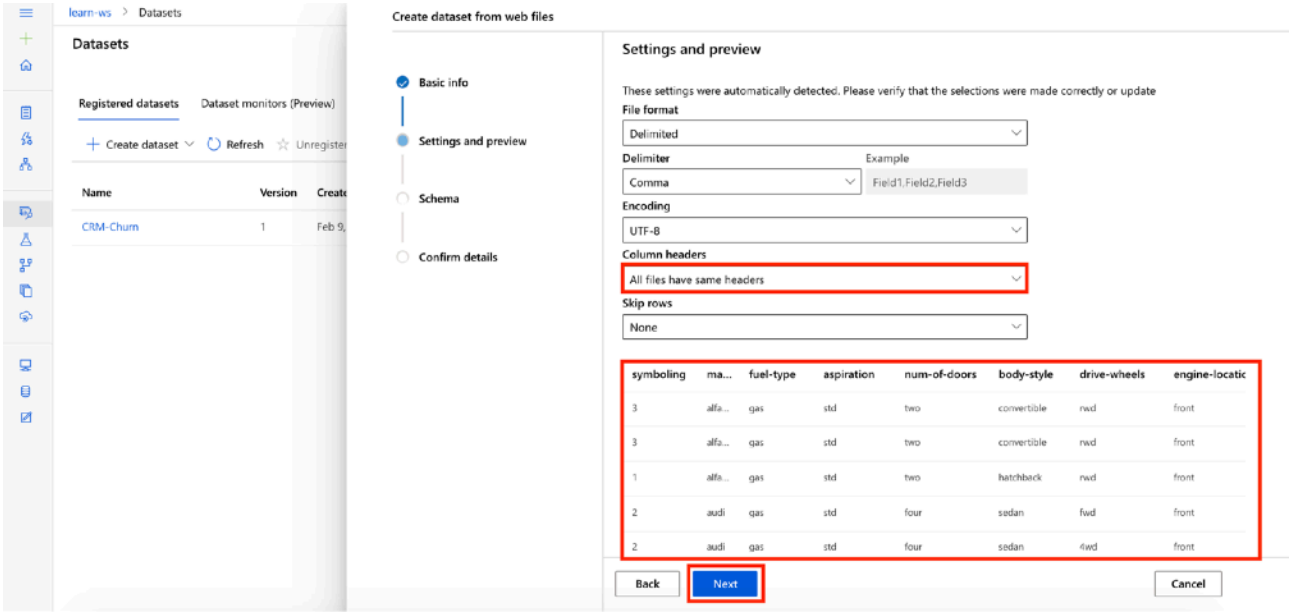
```
https://introtomlsampledata.blob.core.windows.net/data/automobile-price/automobile-price.csv
```

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



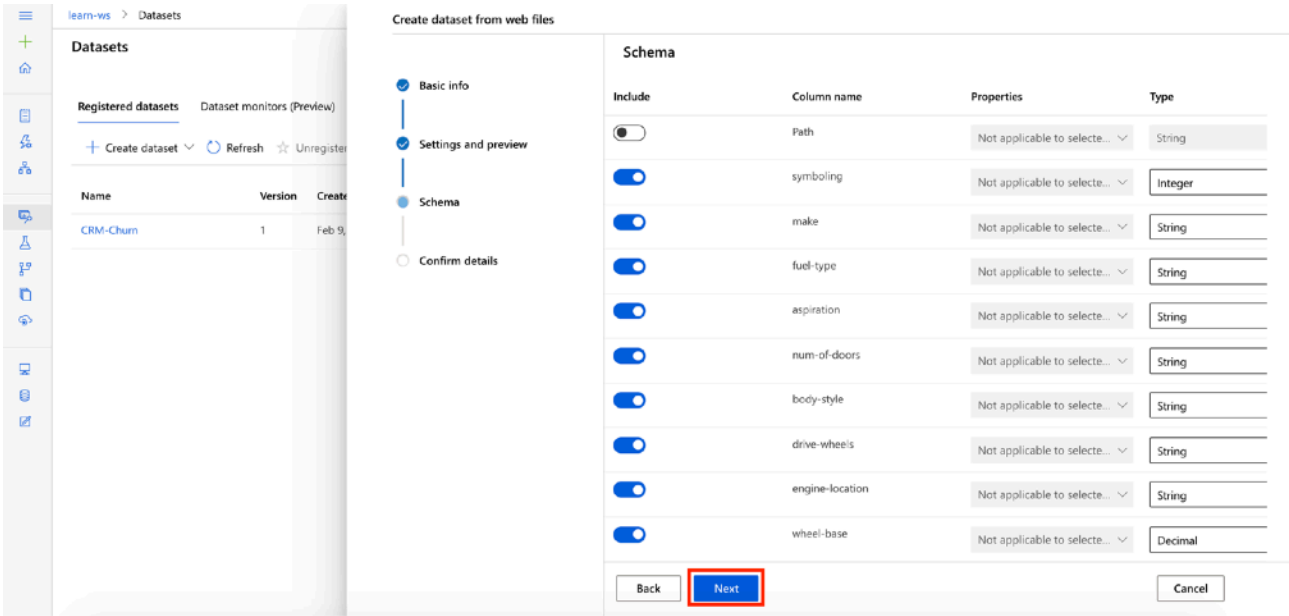
Task 2: Preview Dataset

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**



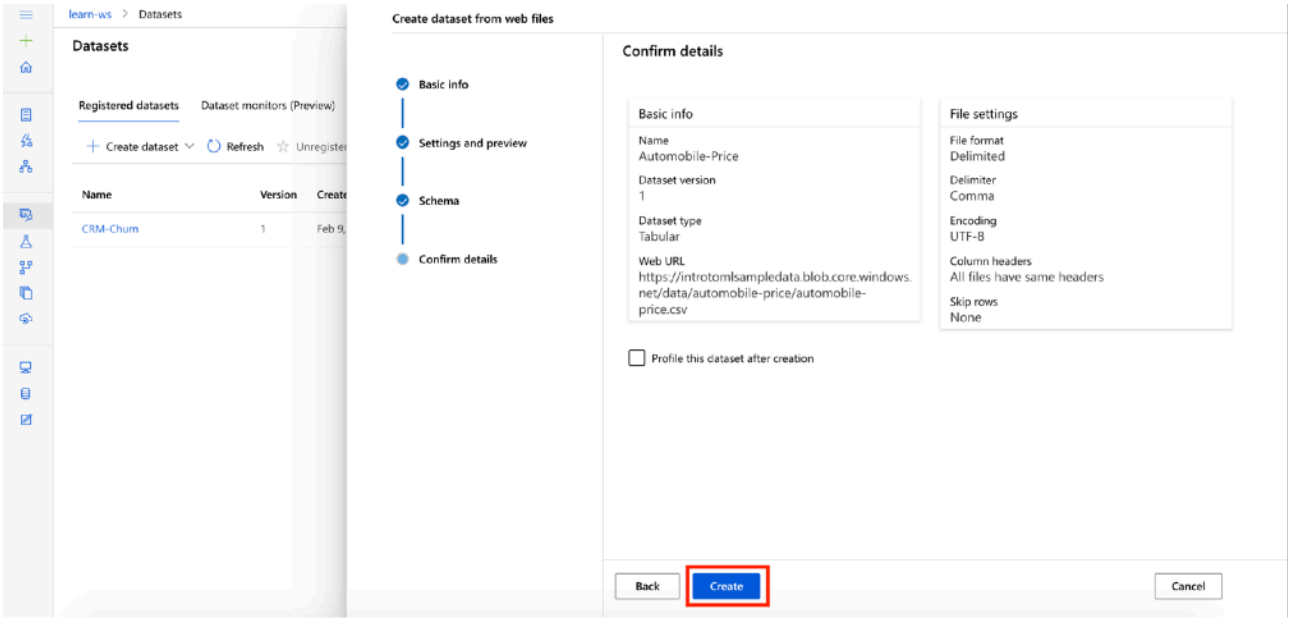
Task 3: Select Columns

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



Task 4: Create Dataset

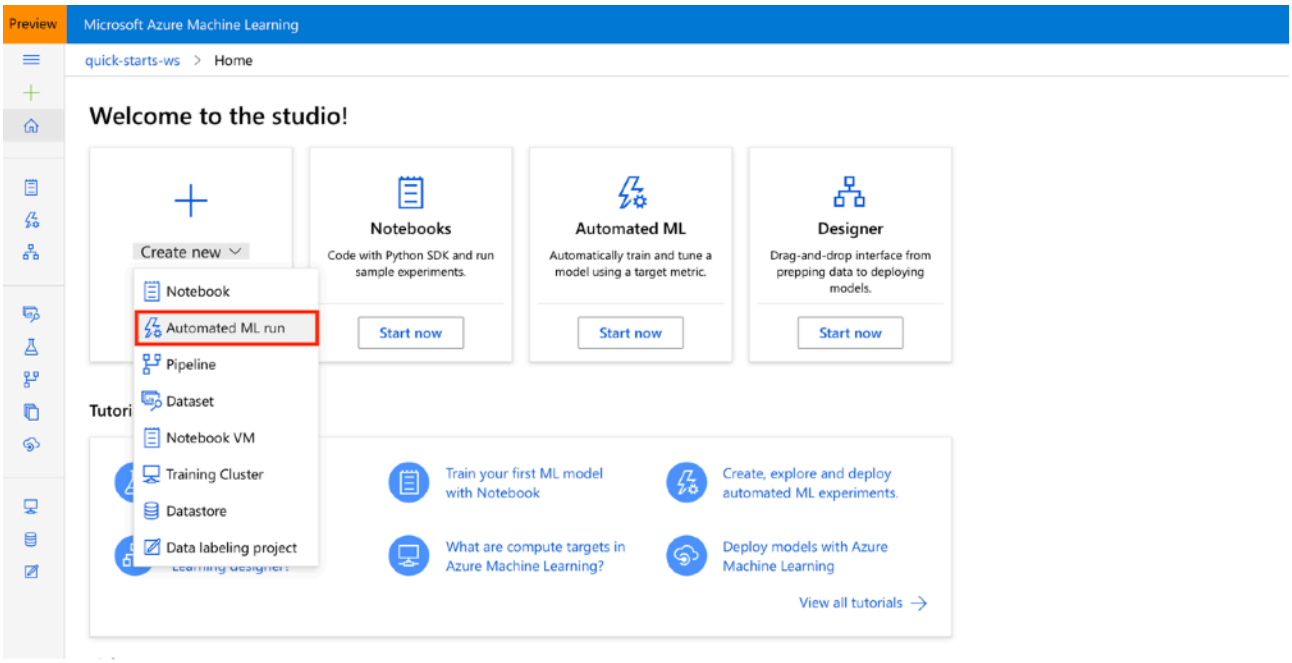
1. Confirm the dataset details and select **Create**



Exercise 2: Setup New Automated Machine Learning Experiment

Task 1: Create New Automated Machine Learning Experiment

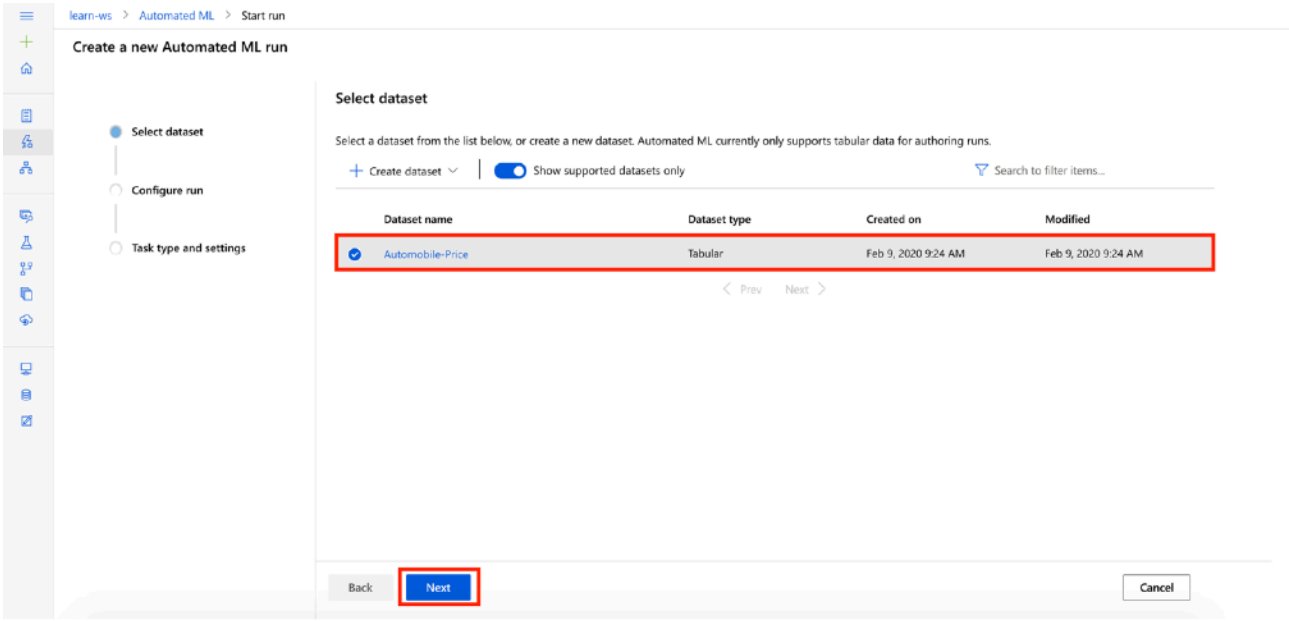
1. From the studio home, select **Create new, Automated ML run**



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

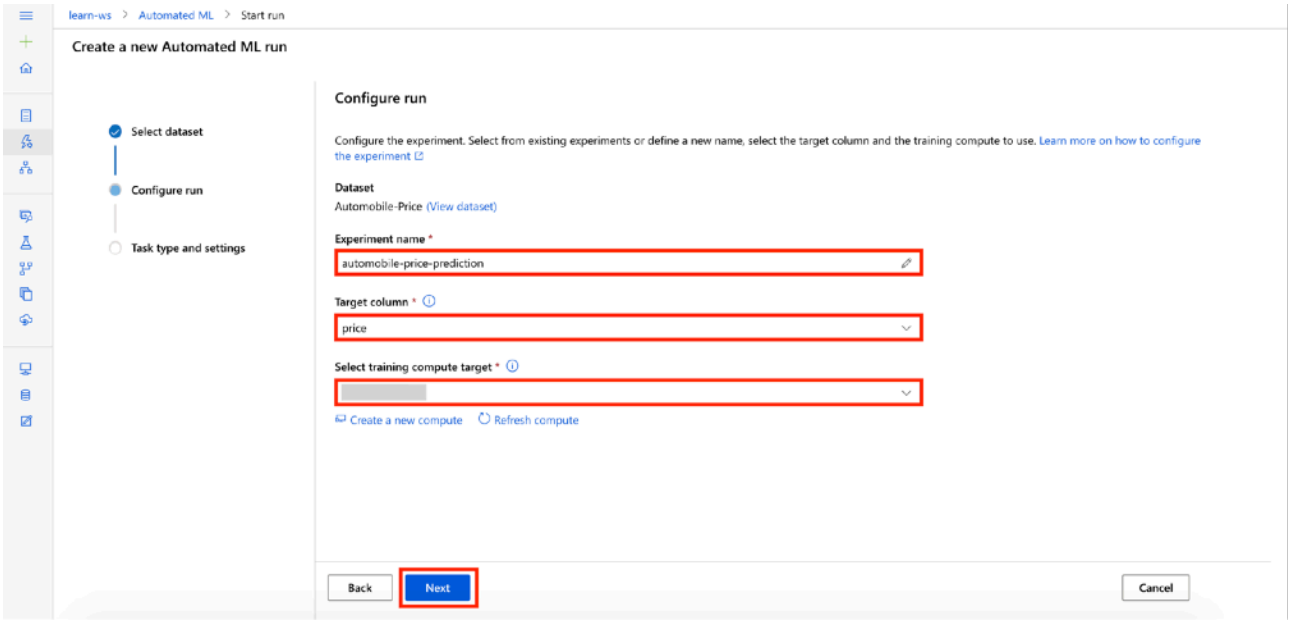
Task 2: Select Training Data

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



Task 3: Create a new Automated ML run

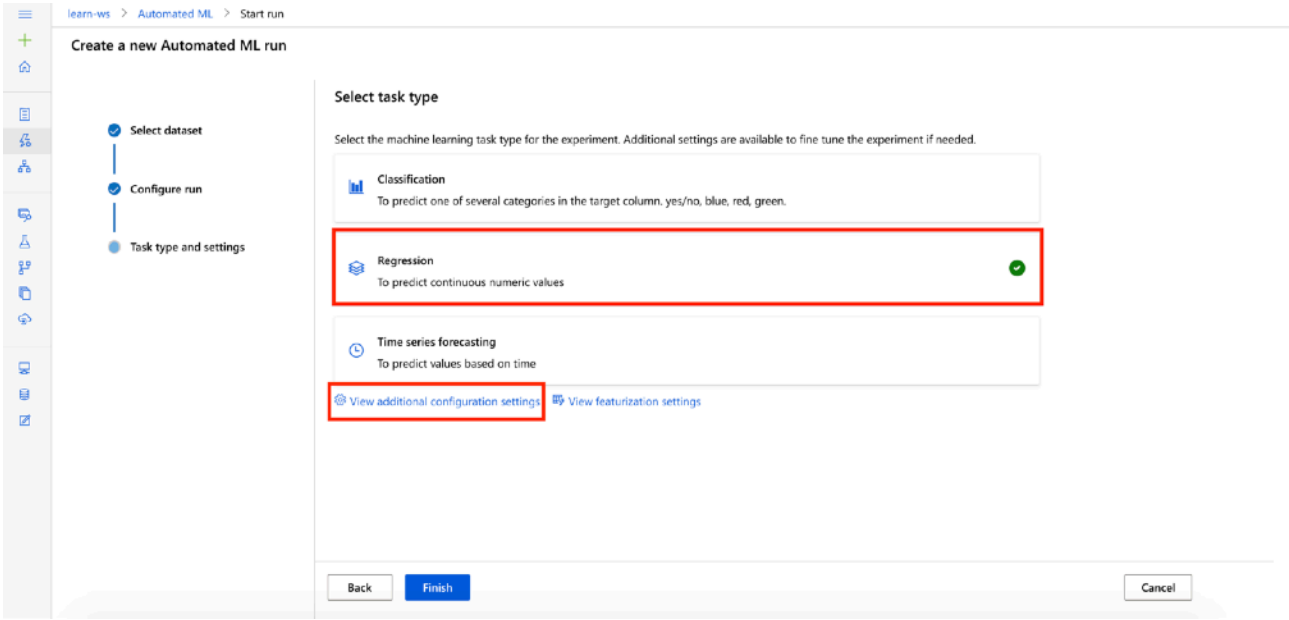
1. Provide an experiment name: **automobile-price-prediction**
2. Select target column: **price**
3. Select compute target: **select the available compute**



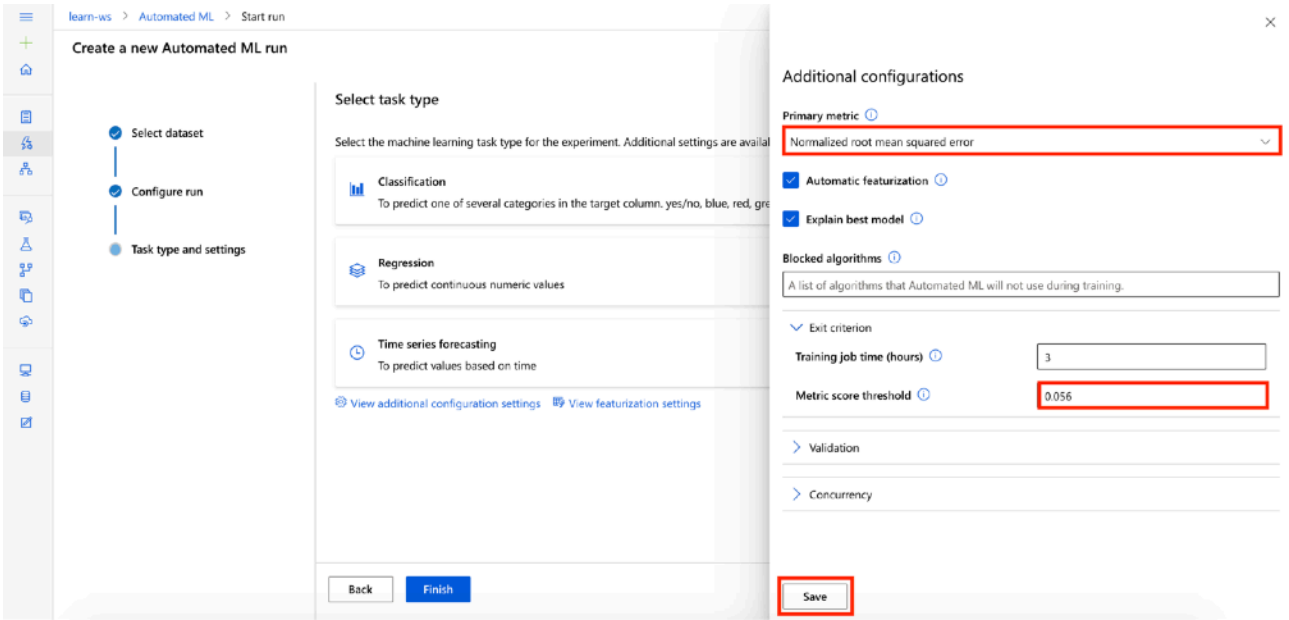
4. Select **Next**

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`

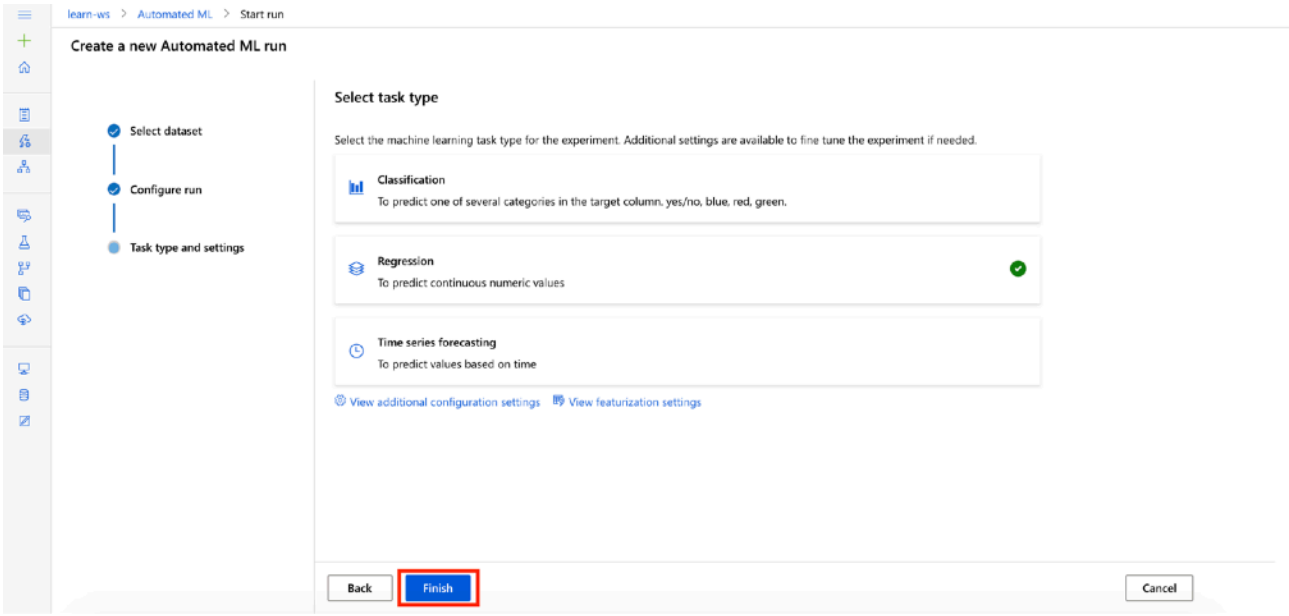


4. Note that we are setting a metric score threshold to limit the training time. In practice, for initial experiments, you will typically only set the training job time to allow AutoML to discover the best algorithm to use for your specific data.

Exercise 3: Start and Monitor Experiment

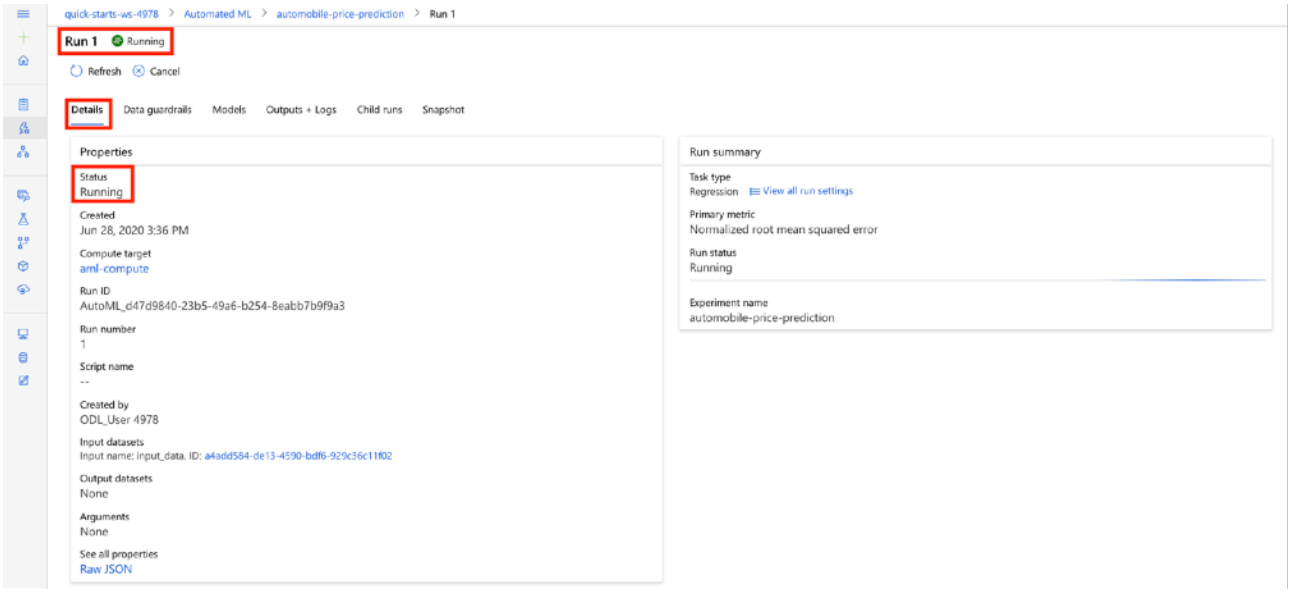
Task 1: Start Experiment

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



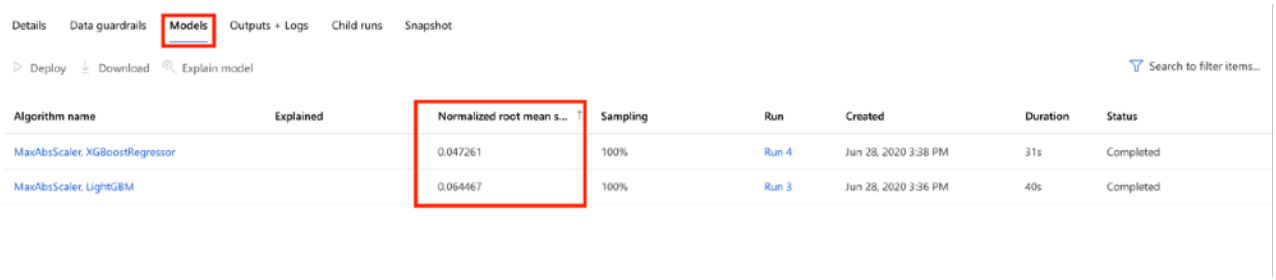
Task 2: Monitor Experiment

1. The experiment will run for about 10 min.

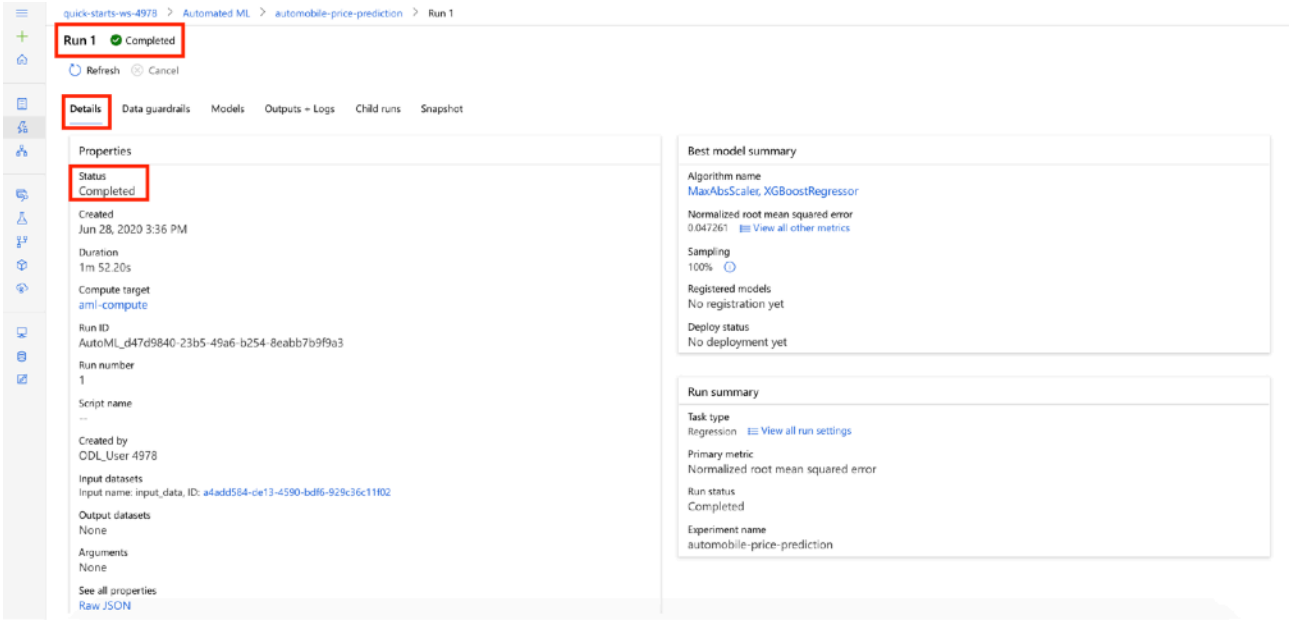


2. In the **Details** tab, observe the **run status** of the job.

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**.

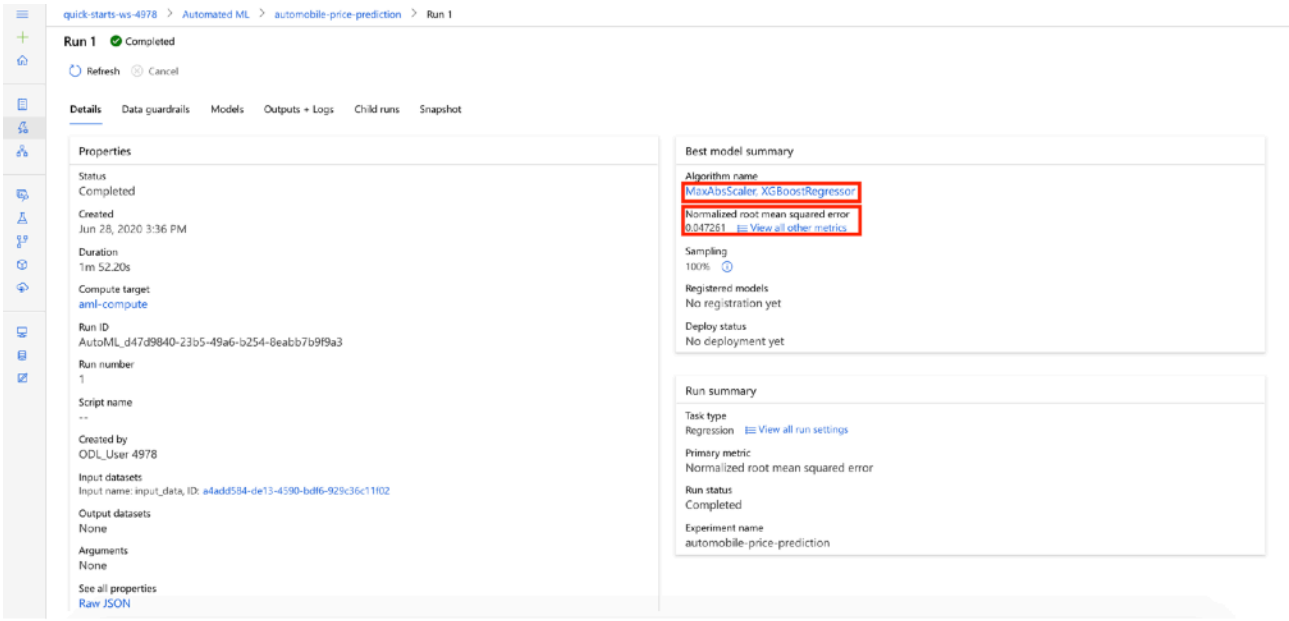


5. While you wait for the model training to complete, you can learn more about how Automated Machine Learning offers preprocessing and data guardrails automatically by selecting Automatic featurization.

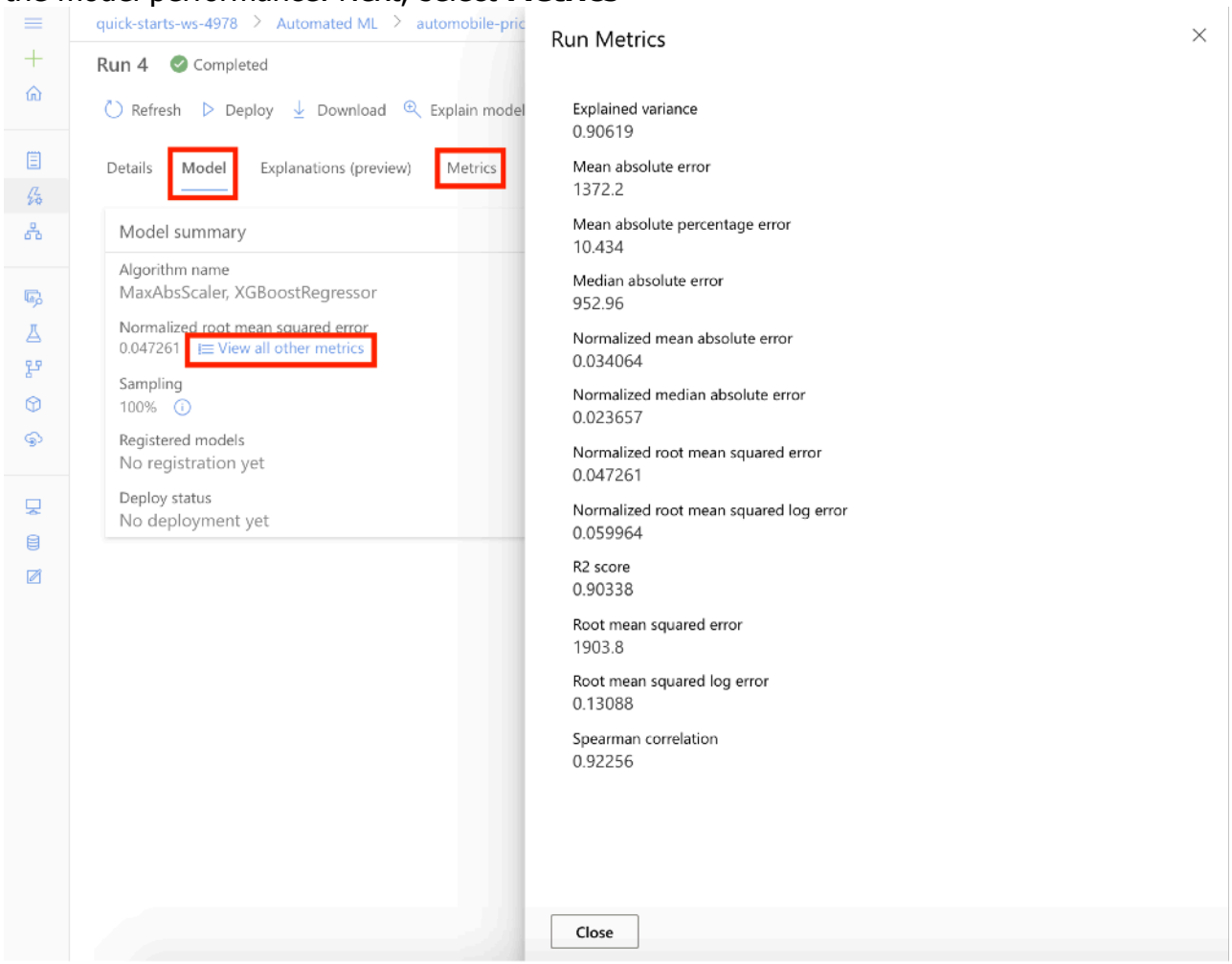
Exercise 4: Review Best Model's Performance

Task 1: Review Best Model Performance

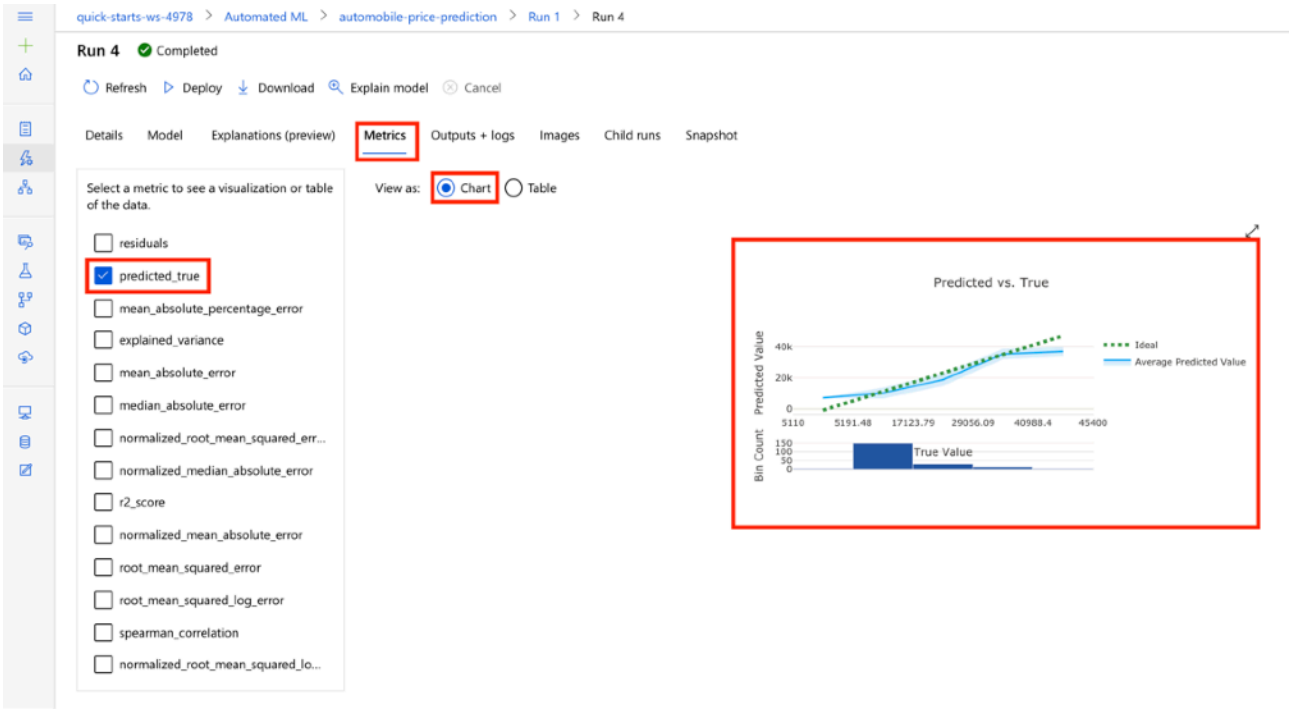
1. From the `Details` tab review the best model's `Algorithm name` and its corresponding `Normalized root mean squared error` score. Next, select the best model's **Algorithm name**



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.

Lab 13 - Train a Simple Clustering Model

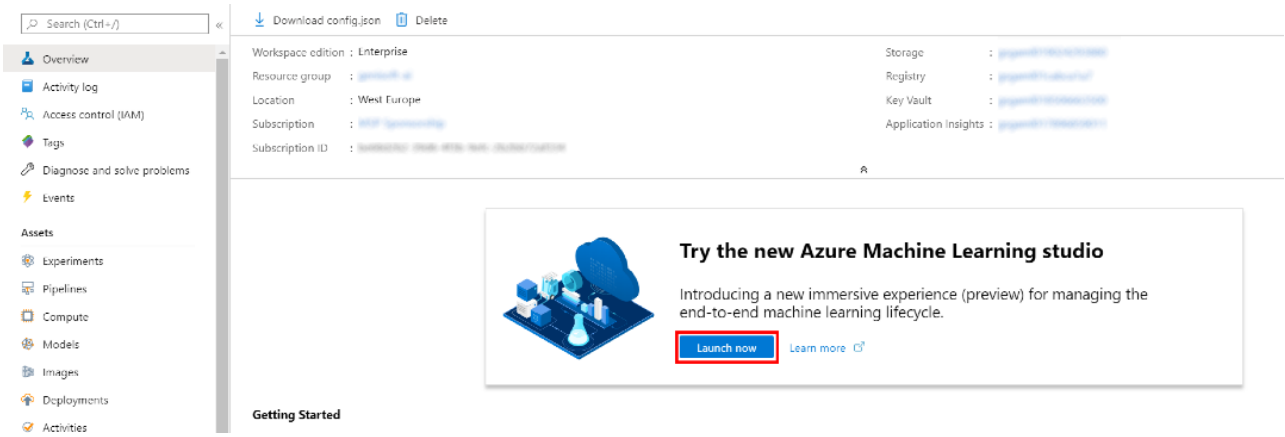
Overview

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.

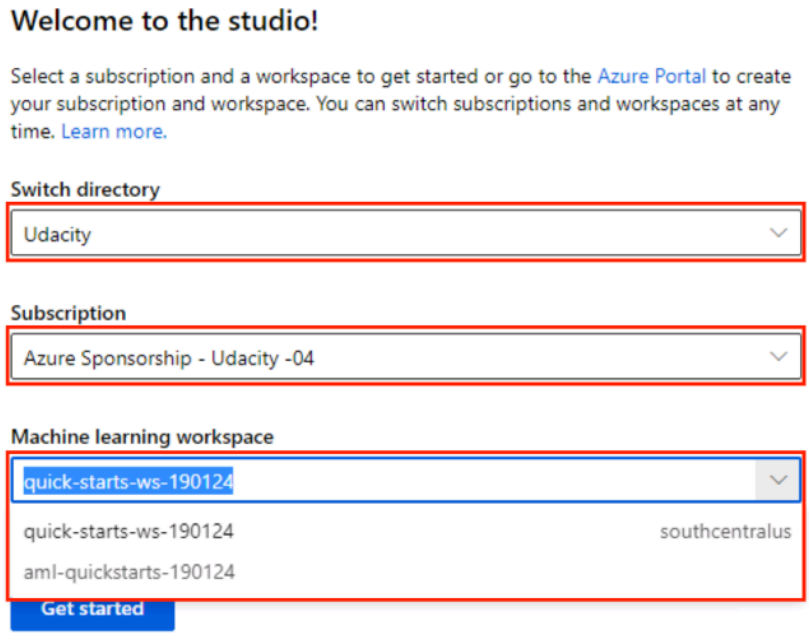
Exercise 1: Create New Training Pipeline

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.

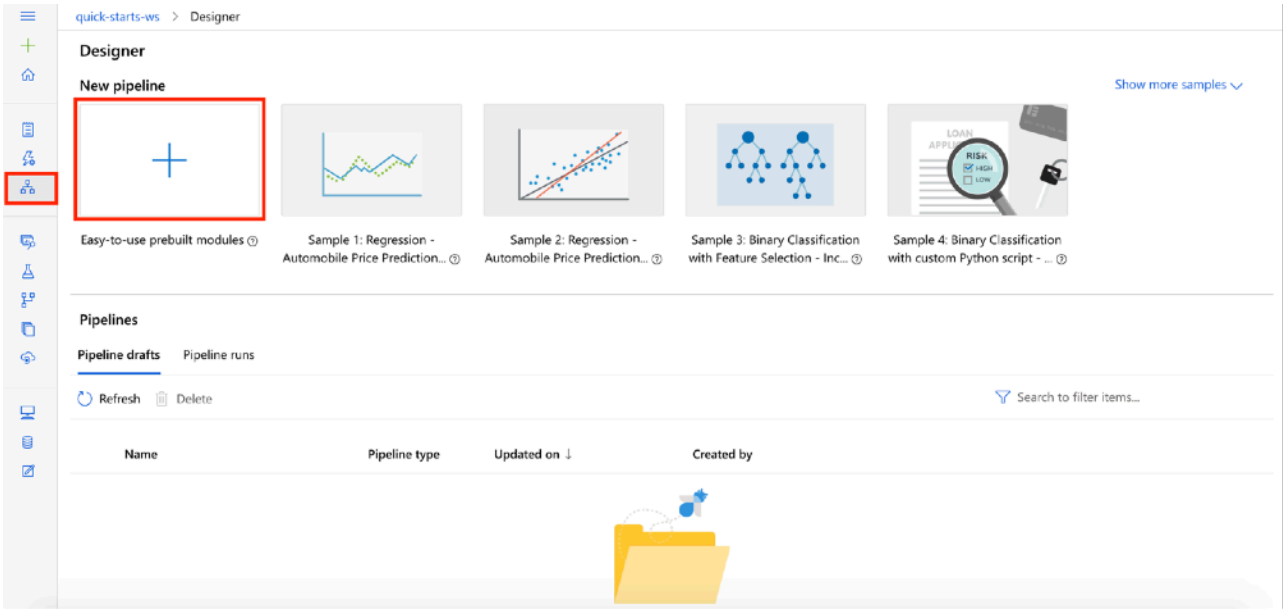


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



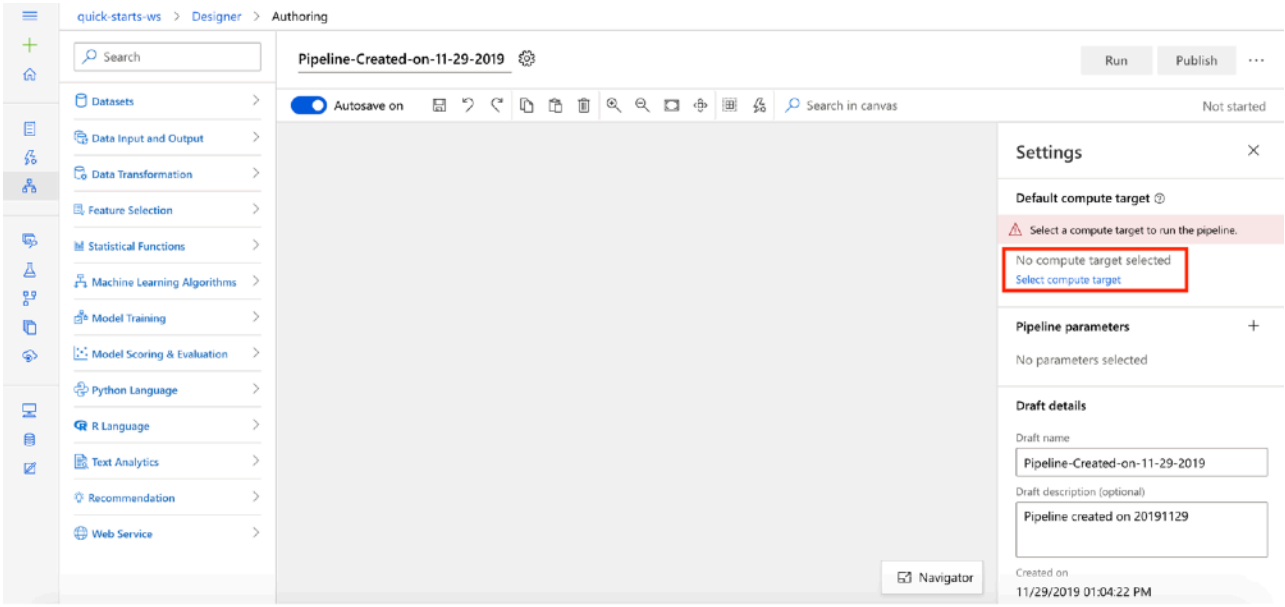
For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

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

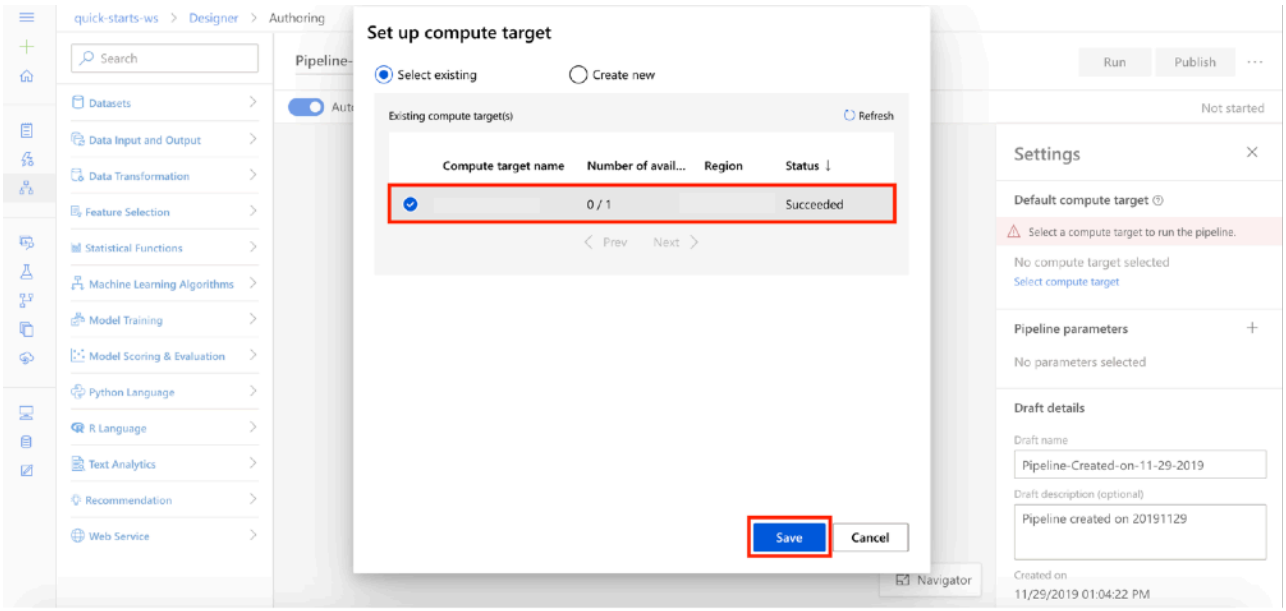


Task 2: Setup Compute Target

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



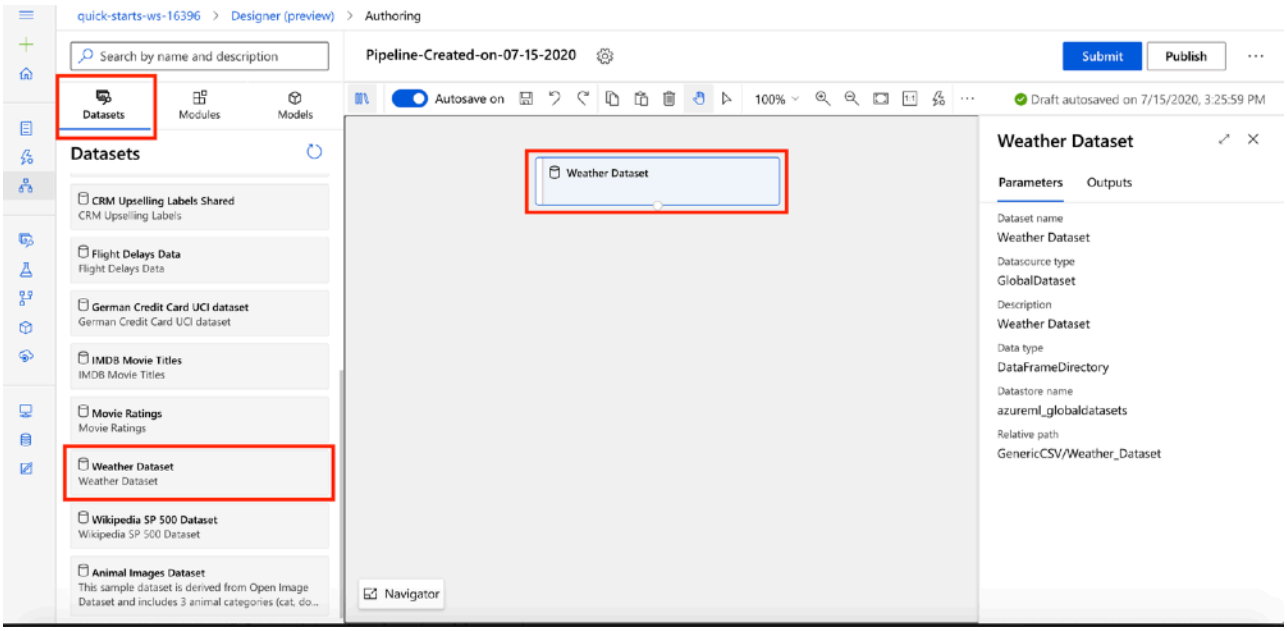
2. In the `Set up compute target` editor, select the available compute, and then select **Save**.



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

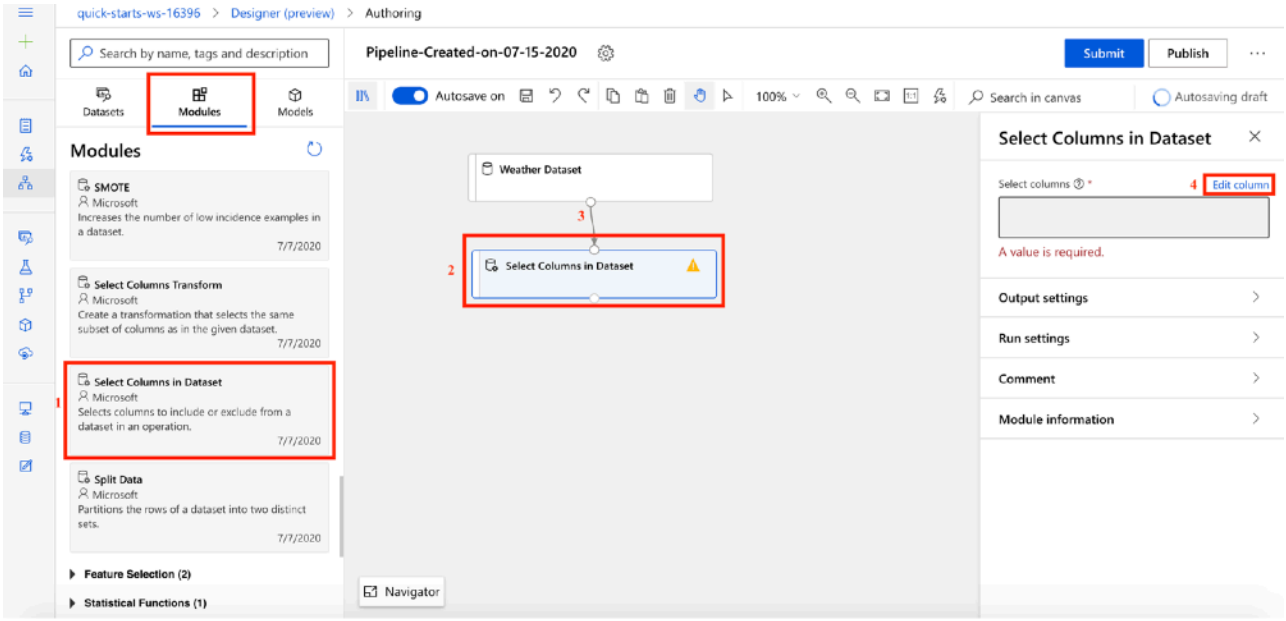
1. Select **Datasets** section in the left navigation. Next, select **Samples, Weather Dataset** and drag and drop the selected dataset on to the canvas.



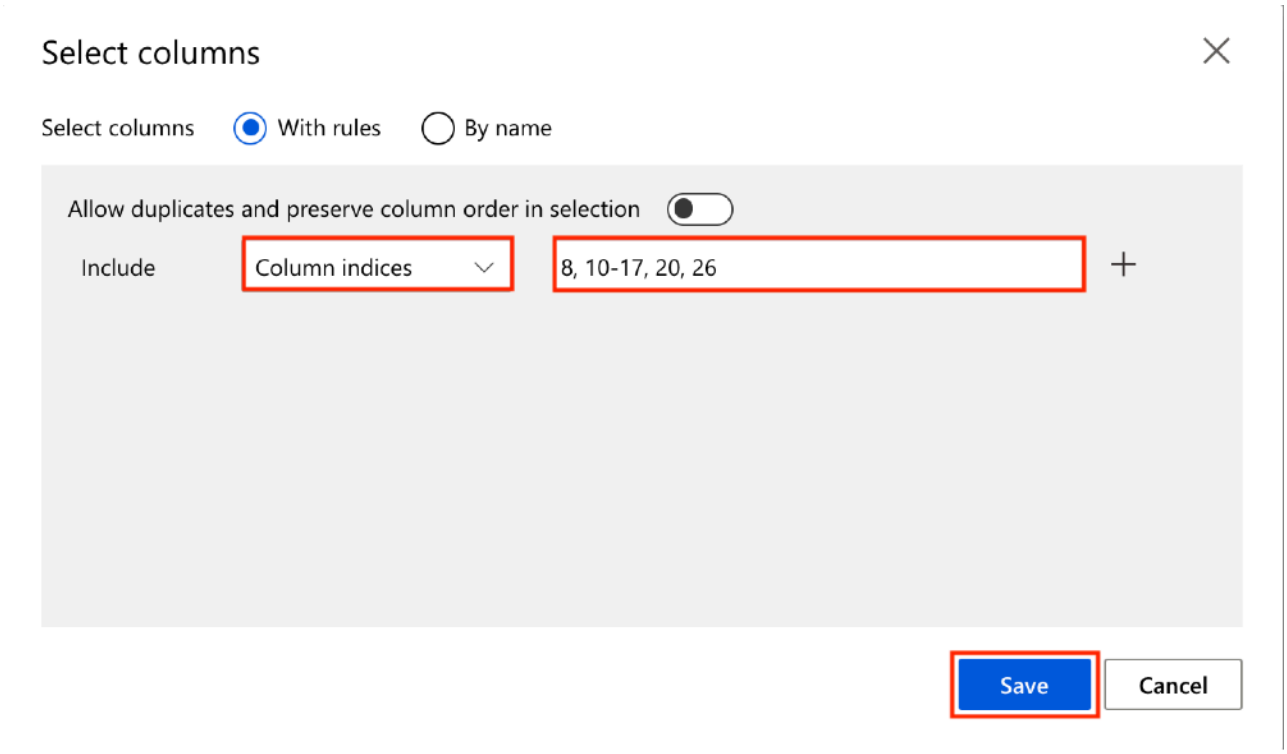
Task 4: Select Columns in Dataset

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

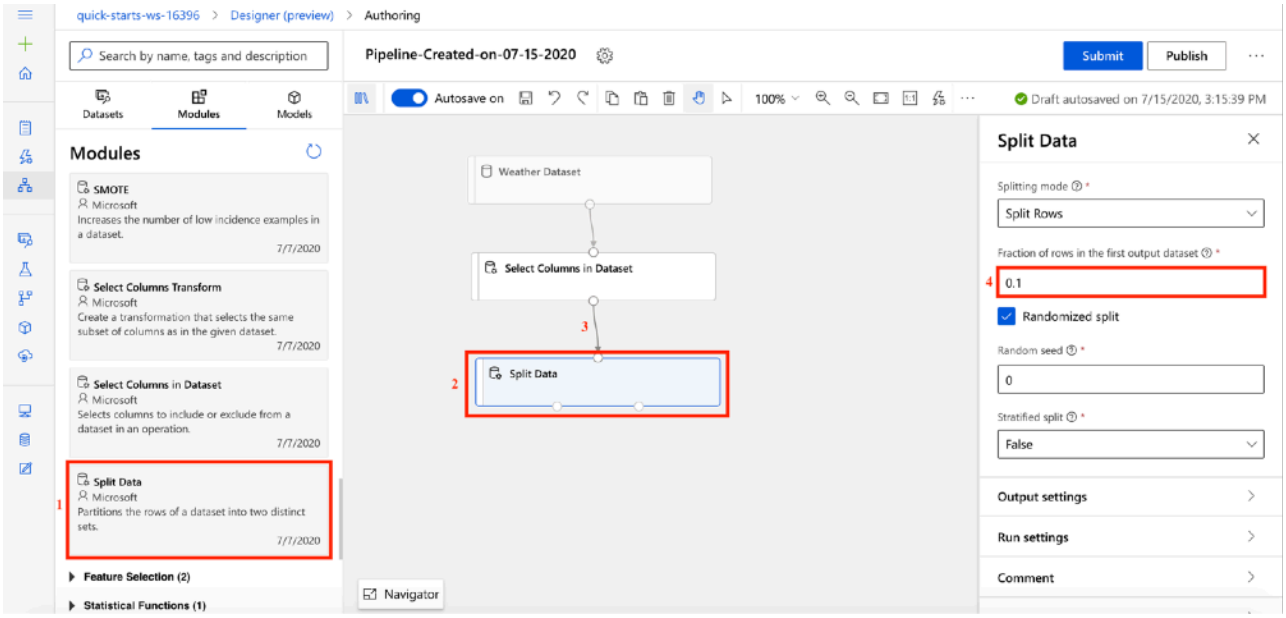


2. Note that you can submit the pipeline at any point to peek at the outputs and activities. Running pipeline also generates metadata that is available for downstream activities such selecting column names from a list in selection dialogs.
3. In the `Select columns` editor, follow the steps outlined below:
    1. Include: **Column indices**
    2. Provide column indices: **8, 10-17, 20, 26**
    3. Select **Save**



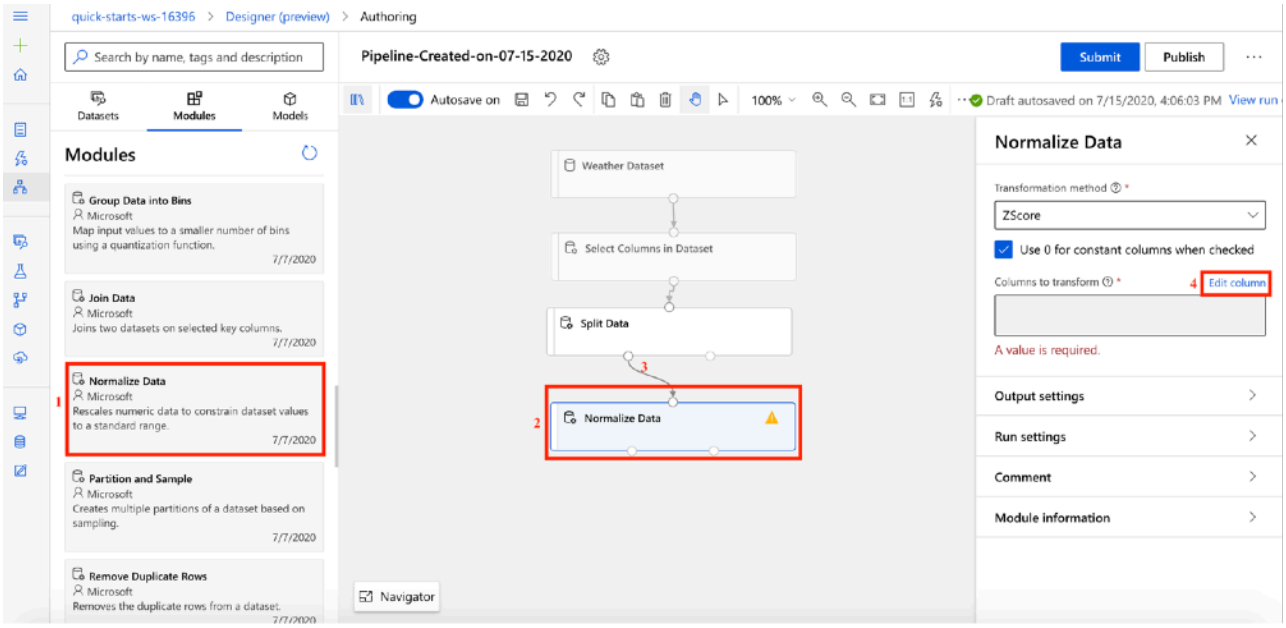
Task 5: Split Data

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**

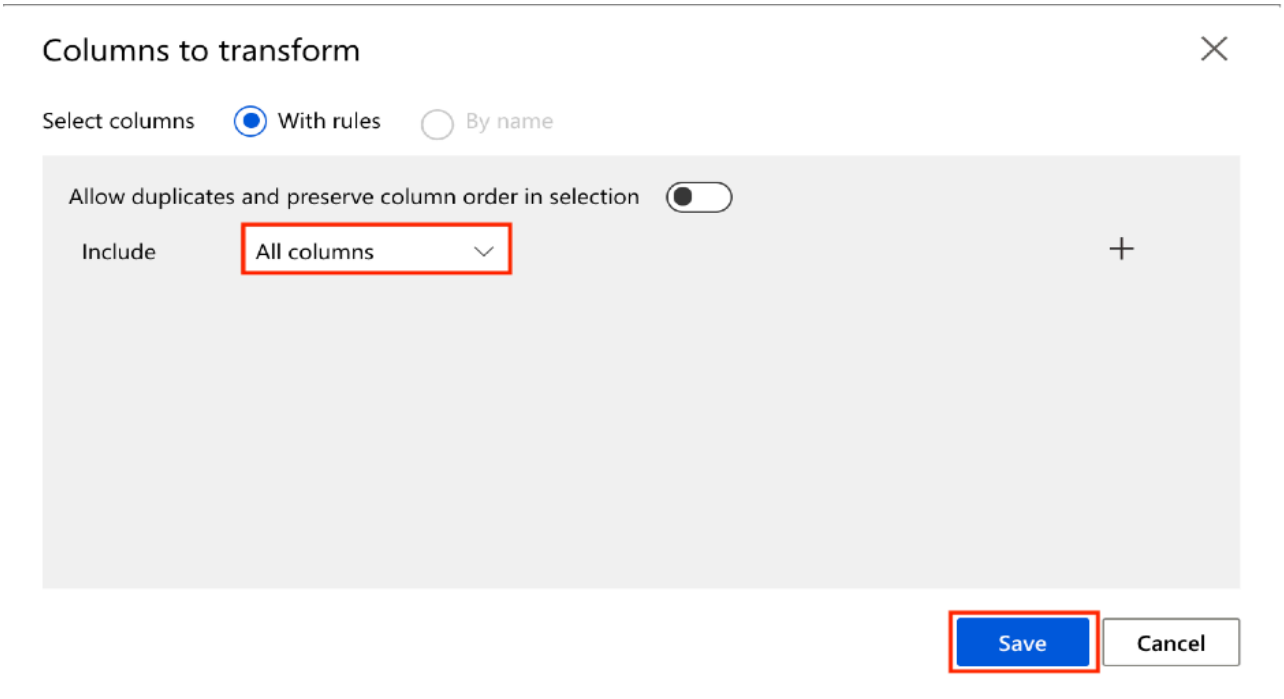


Task 6: Normalize Data

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

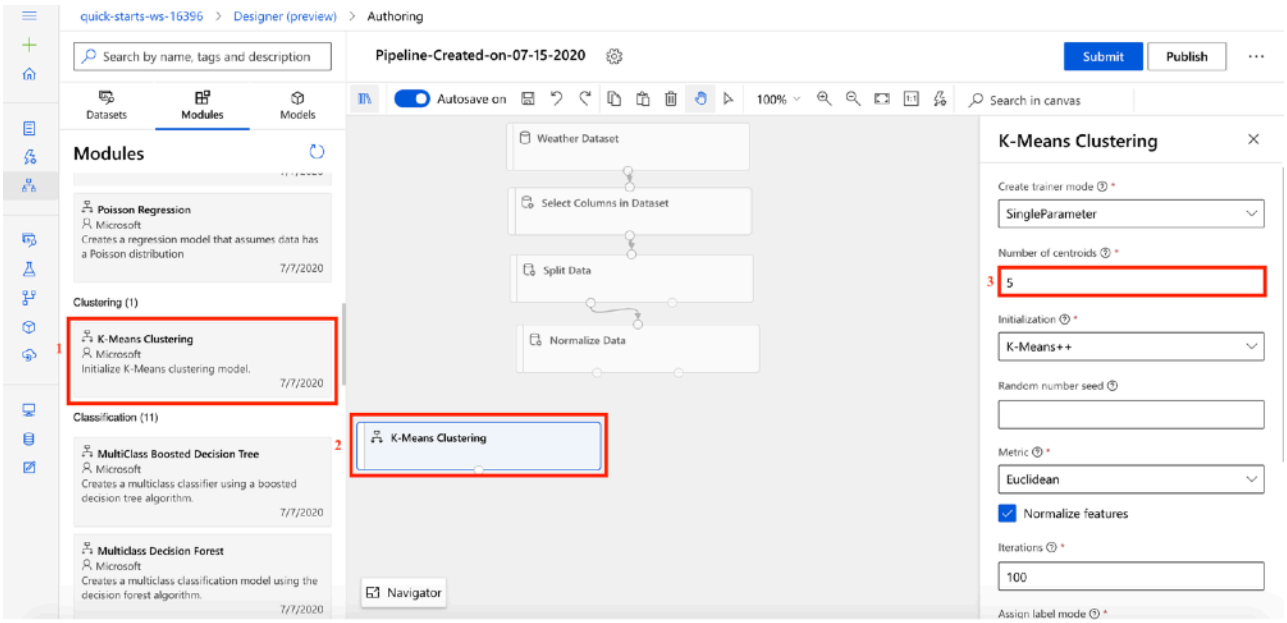


2. In the `Columns to transform` editor, follow the steps outlined below:
    1. Include: **All columns**
    2. Select **Save**



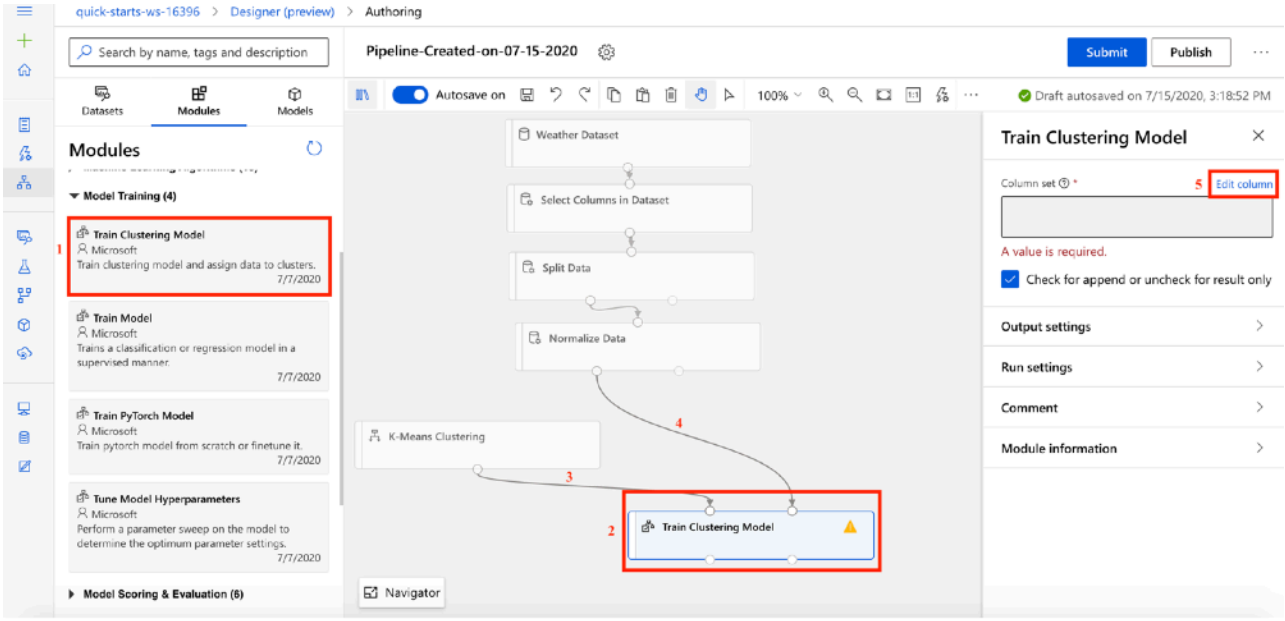
Task 7: Initialize K-Means Clustering Model

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**

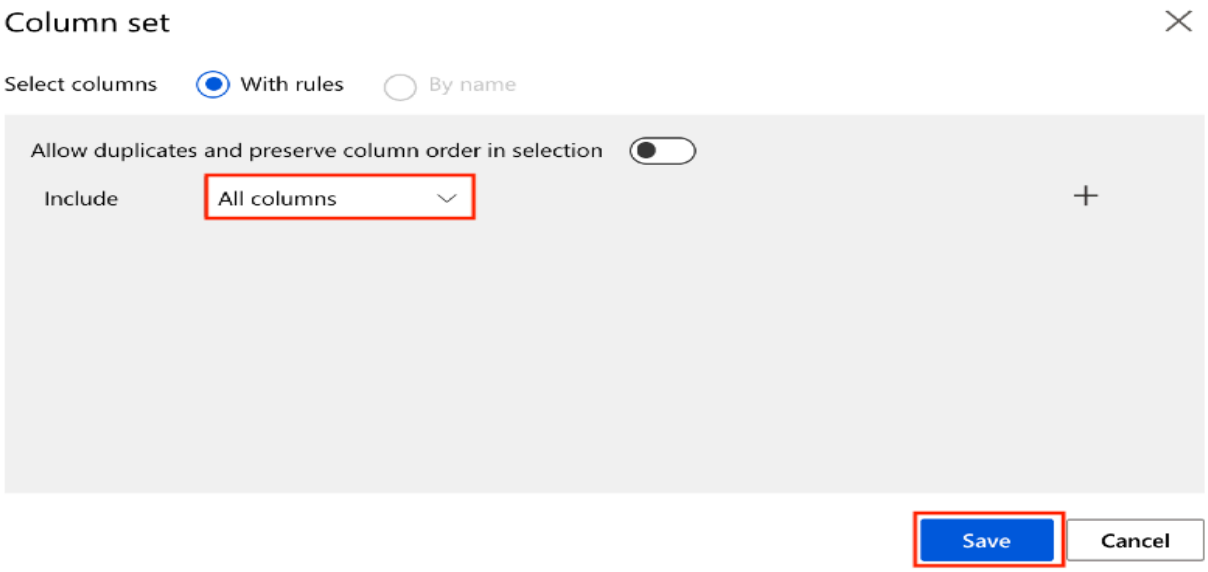


Task 8: Setup Train Clustering Model Module

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

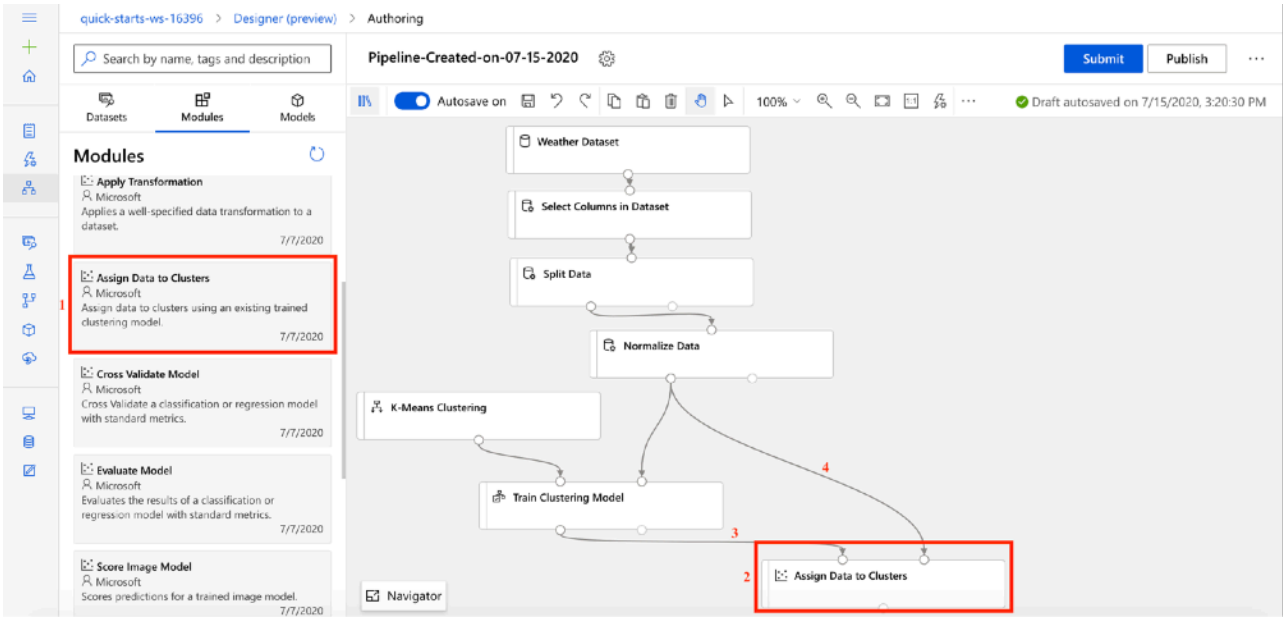


2. In the `Columns set` editor, follow the steps outlined below:
    1. Include: **All columns**
    2. Select **Save**



Task 9: Setup Assign Data to Clusters Module

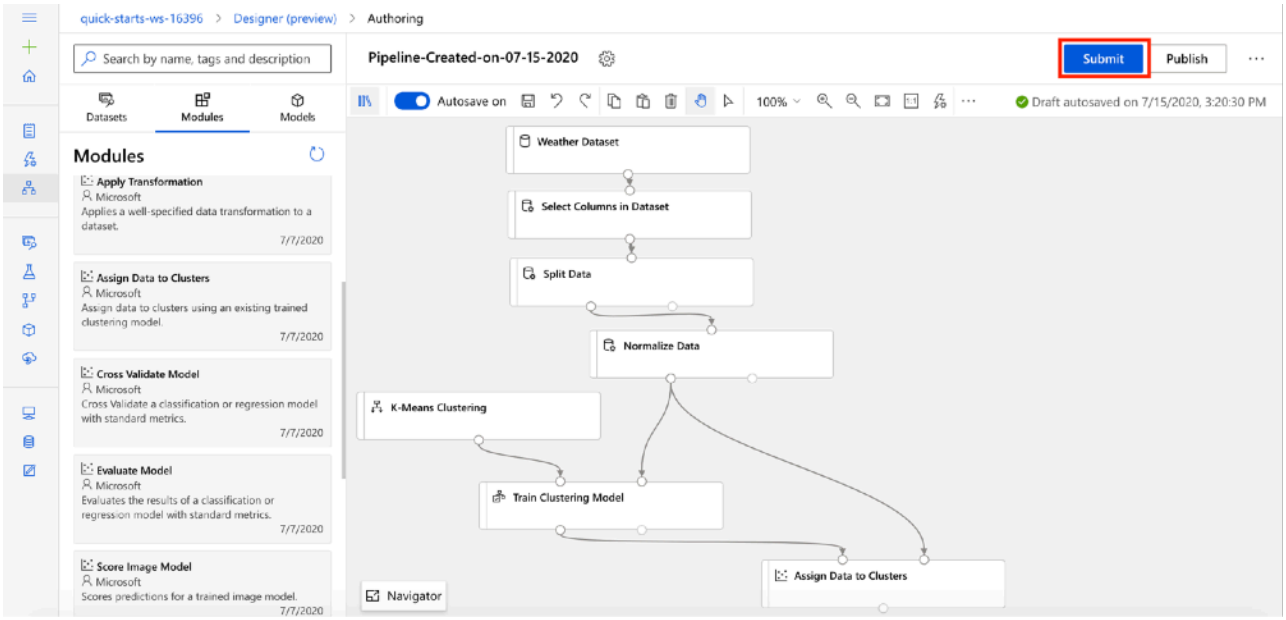
1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Assign Data to Clusters** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the first output of the `Train Clustering Model` module to the 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



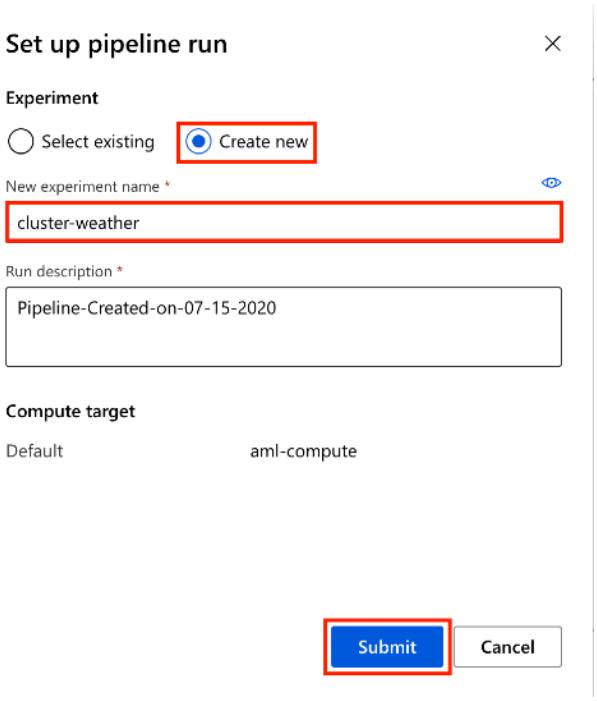
Exercise 2: Submit Training Pipeline

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**.



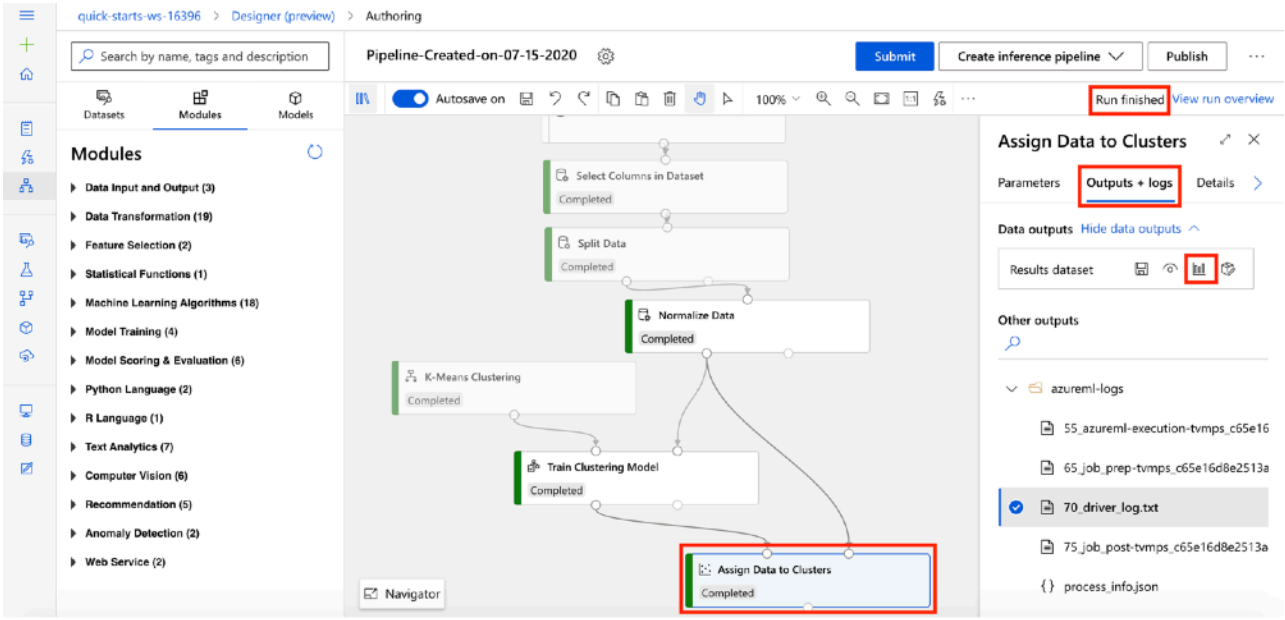
3. Wait for pipeline run to complete. It will take around **10 minutes** to complete the run.

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

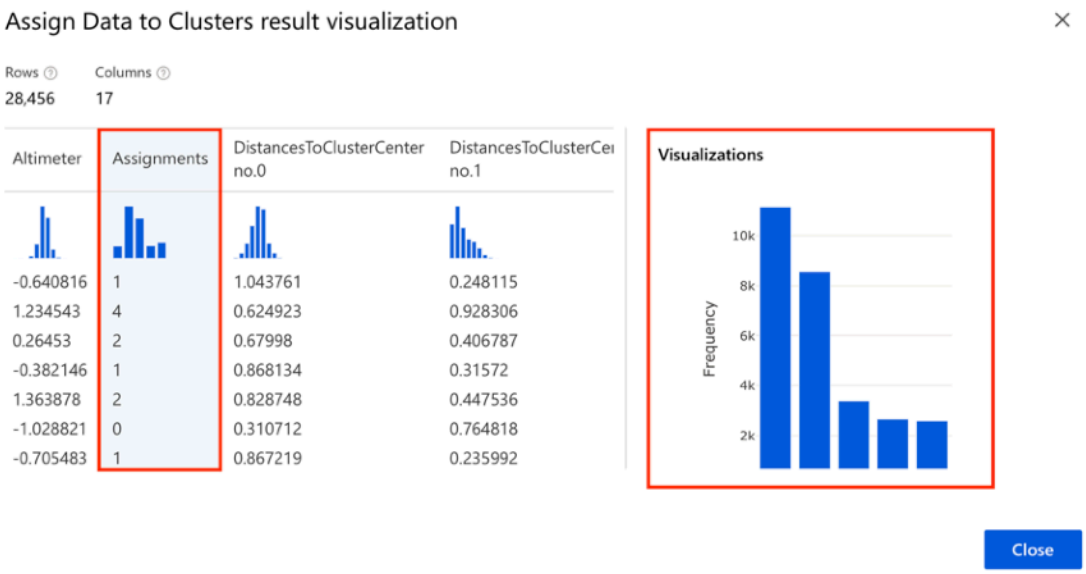
Task 1: Open the Visualization Dialog

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.

Lab 14 - Train a simple neural net model

Although neural networks are widely known for use in deep learning and modeling complex problems such as image recognition, they are easily adapted to regression problems. Any class of statistical models can be termed a neural network if they use adaptive weights and can approximate non-linear functions of their inputs. Thus neural network regression is suited to problems where a more traditional regression model cannot fit a solution.
Neural network regression is a supervised learning method, and therefore requires a tagged dataset, which includes a label column. Because a regression model predicts a numerical value, the label column must be a numerical data type.

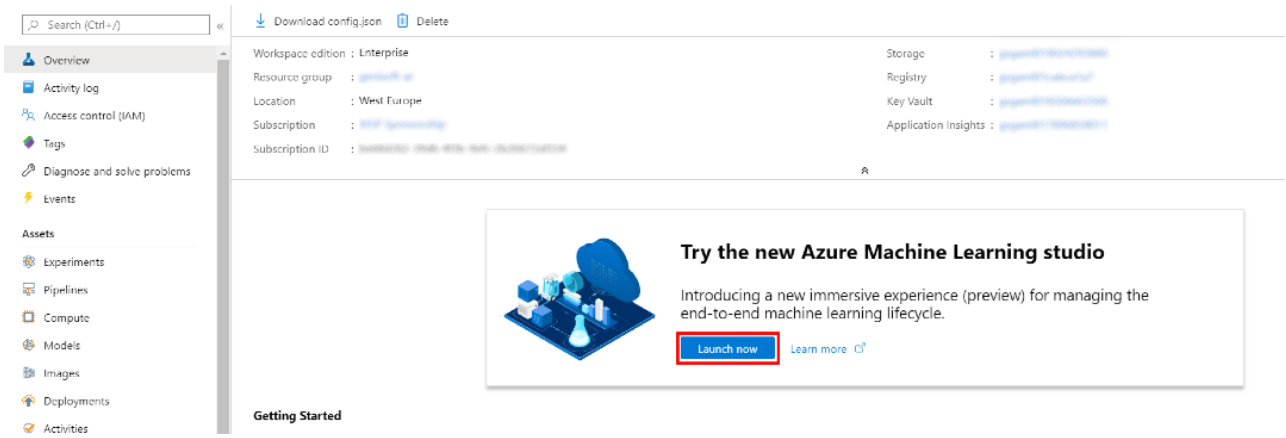
Overview

In this lab we will be using a subset of NYC Taxi & Limousine Commission - green taxi trip records available from Azure Open Datasets. The data is enriched with holiday and weather data. Based on the enriched dataset, we will configure the prebuilt Neural Network Regression module to create a regression model using a customizable neural network algorithm. We will train the model by providing the model and the NYC taxi dataset as an input to Train Model. The trained model can then be used to predict NYC taxi fares. We will do all of this from the Azure Machine Learning designer without writing a single line of code.

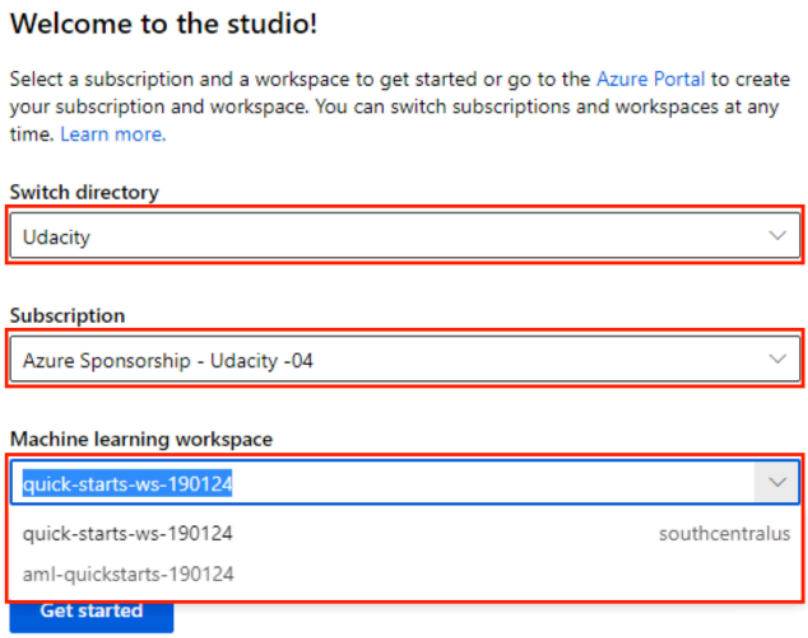
Exercise 1: Register Dataset with Azure Machine Learning studio

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.

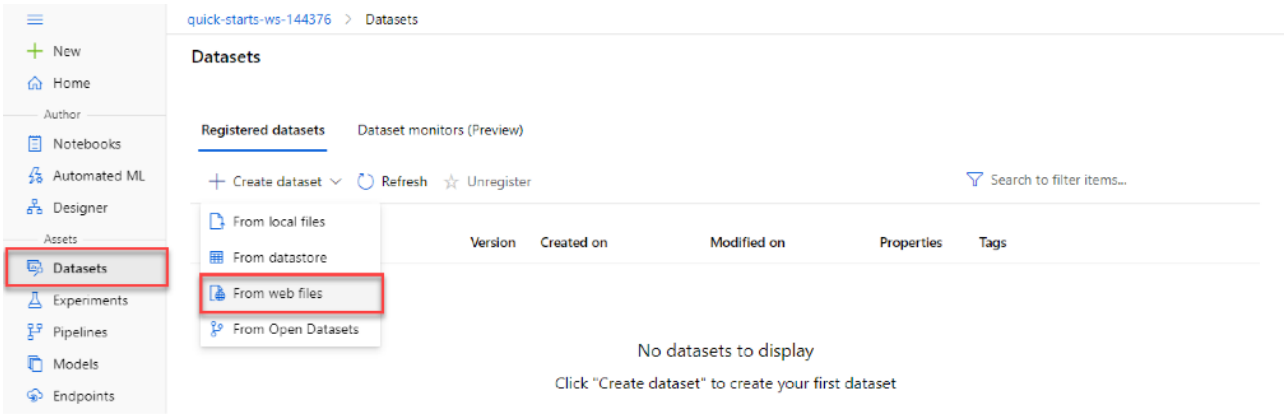


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



4. 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**.

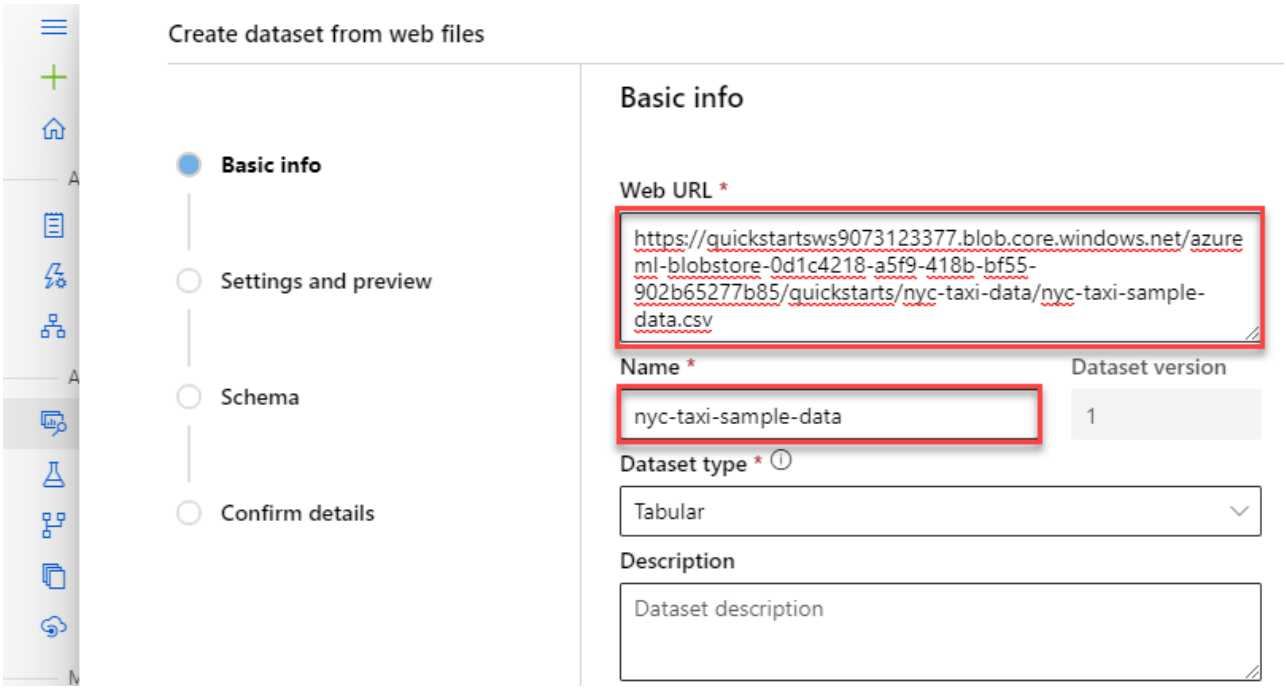
5. From the studio, select **Datasets, + Create dataset, From web files**. This will open the `Create dataset from web files` dialog on the right.



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

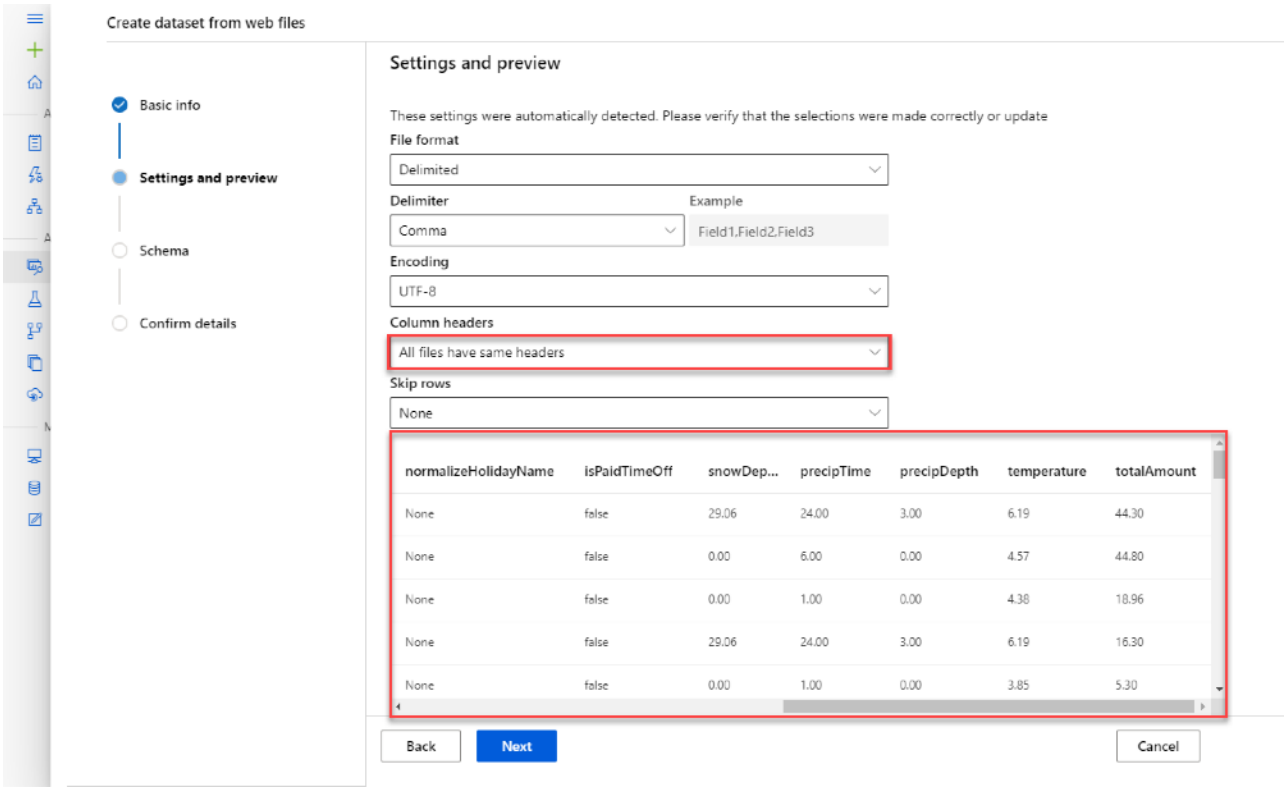
   https://introtomlsampledata.blob.core.windows.net/data/nyc-taxi/nyc-taxi-sample-data.csv

7. Provide `nyc-taxi-sample-data` as the Name, leave the remaining values at their defaults and select **Next**.



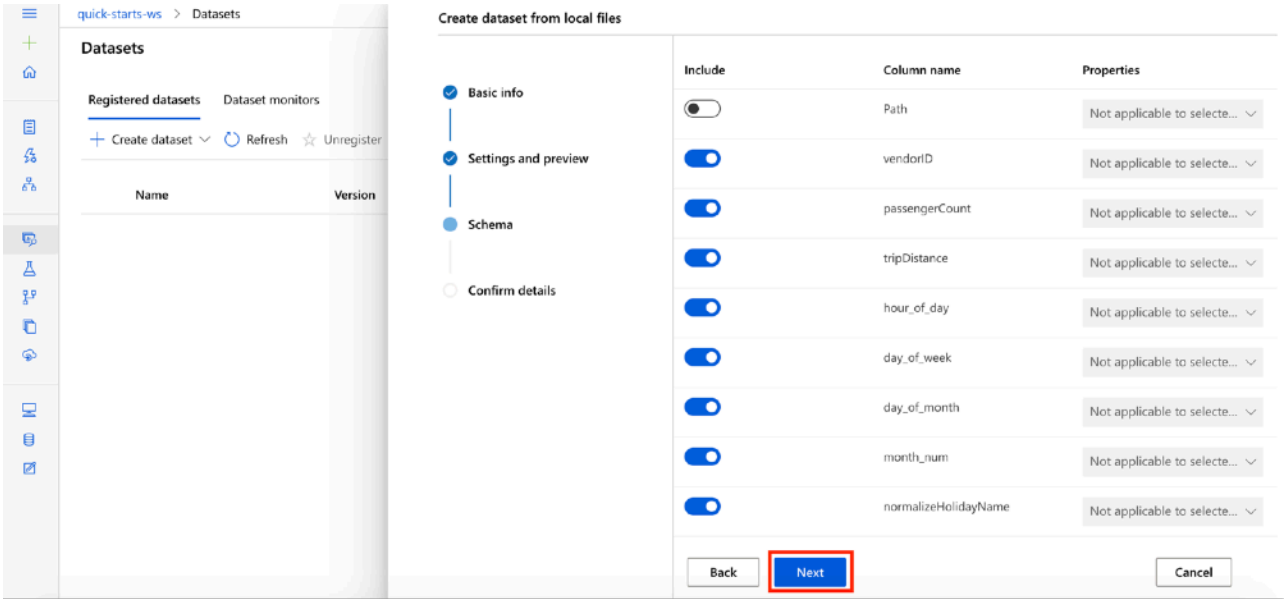
Task 2: Preview Dataset

1. On the Settings and preview panel, set the column headers drop down to `All files have same headers`.
2. Scroll the data preview to right to observe the target column: `totalAmount`. After you are done reviewing the data, select **Next**



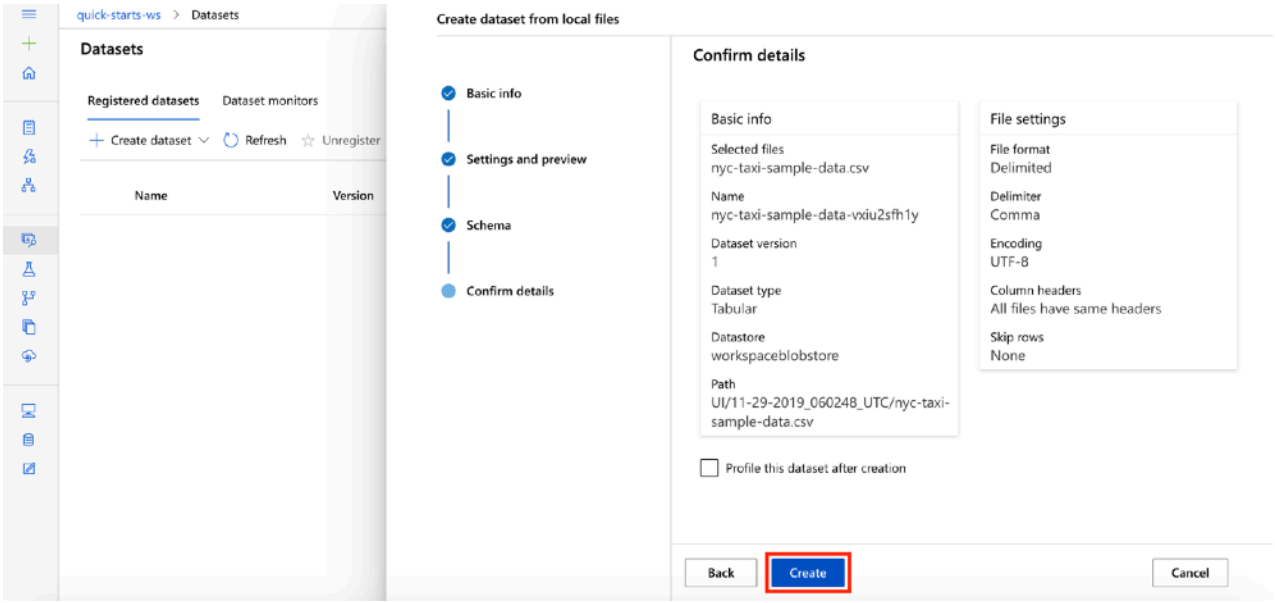
Task 3: Select Columns

1. Select columns from the dataset to include as part of your training data. Leave the default selections and select **Next**



Task 4: Create Dataset

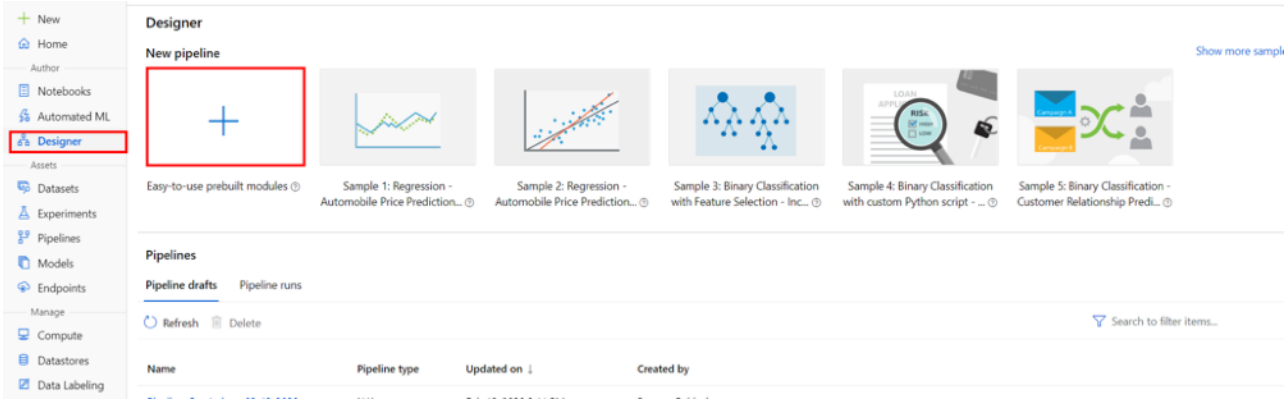
1. Confirm the dataset details and select **Create**



Exercise 2: Create New Training Pipeline

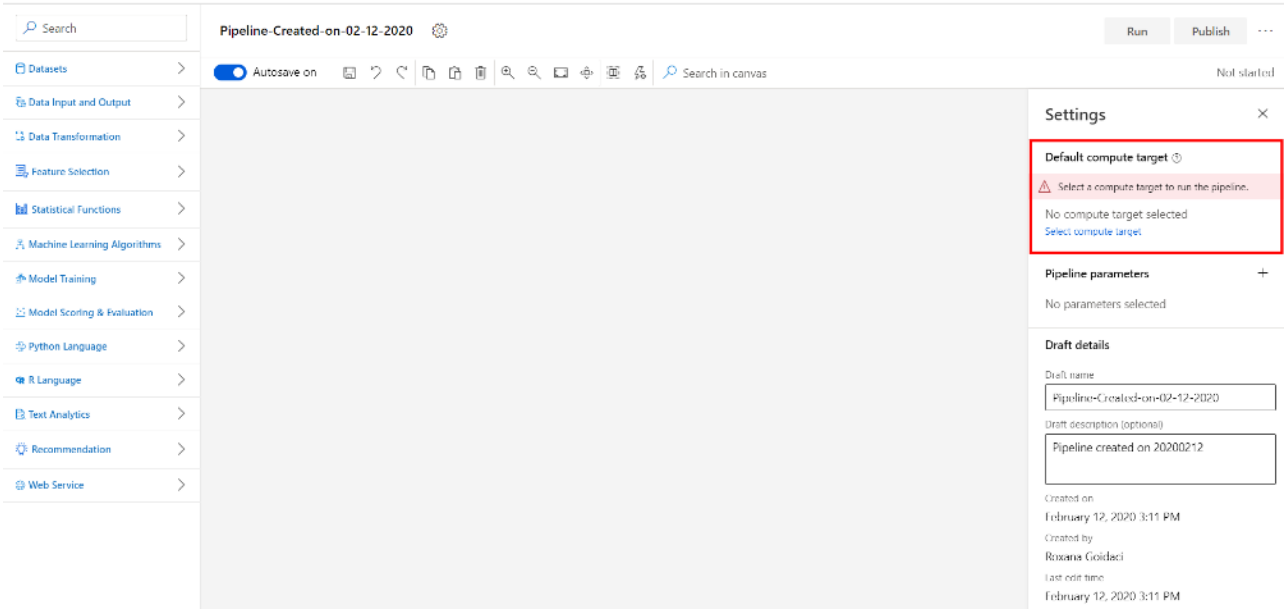
Task 1: Open Pipeline Authoring Editor

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



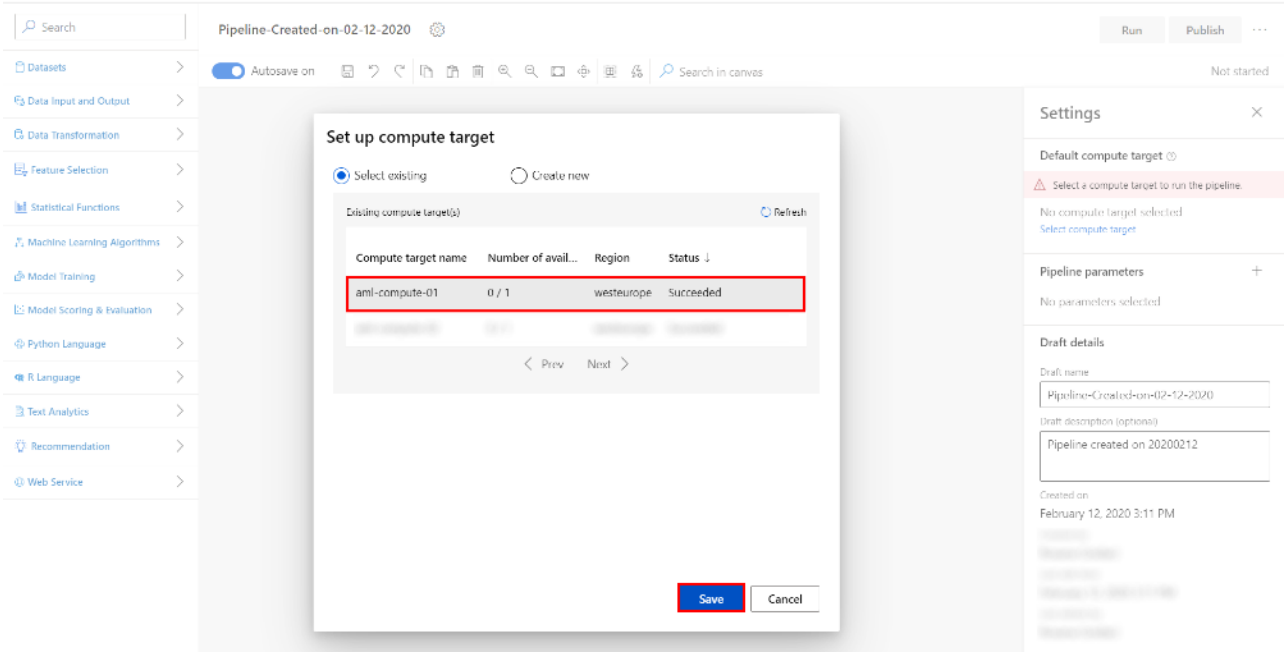
Task 2: Setup Compute Target

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



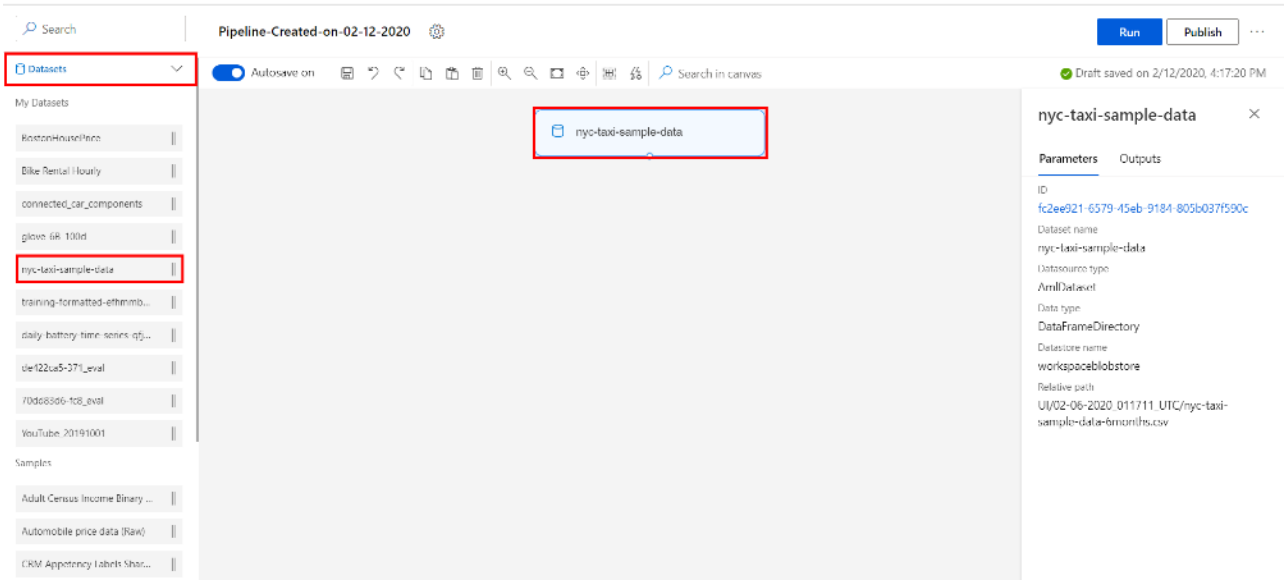
2. In the `Set up compute target` editor, select the available compute, and then select **Save**.

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.



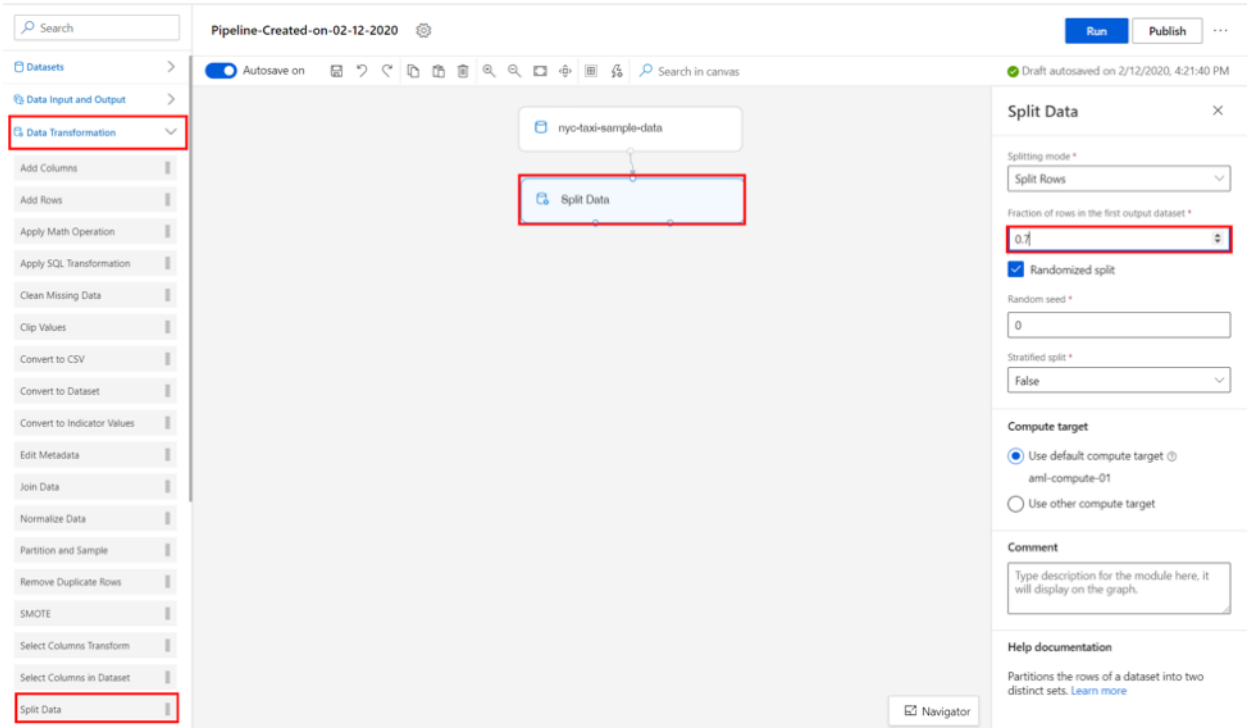
Task 3: Add Dataset

1. Select **Datasets** section in the left navigation. Next, select **My Datasets, nyc-taxi-sample-data** and drag and drop the selected dataset on to the canvas.



Task 4: Split Dataset

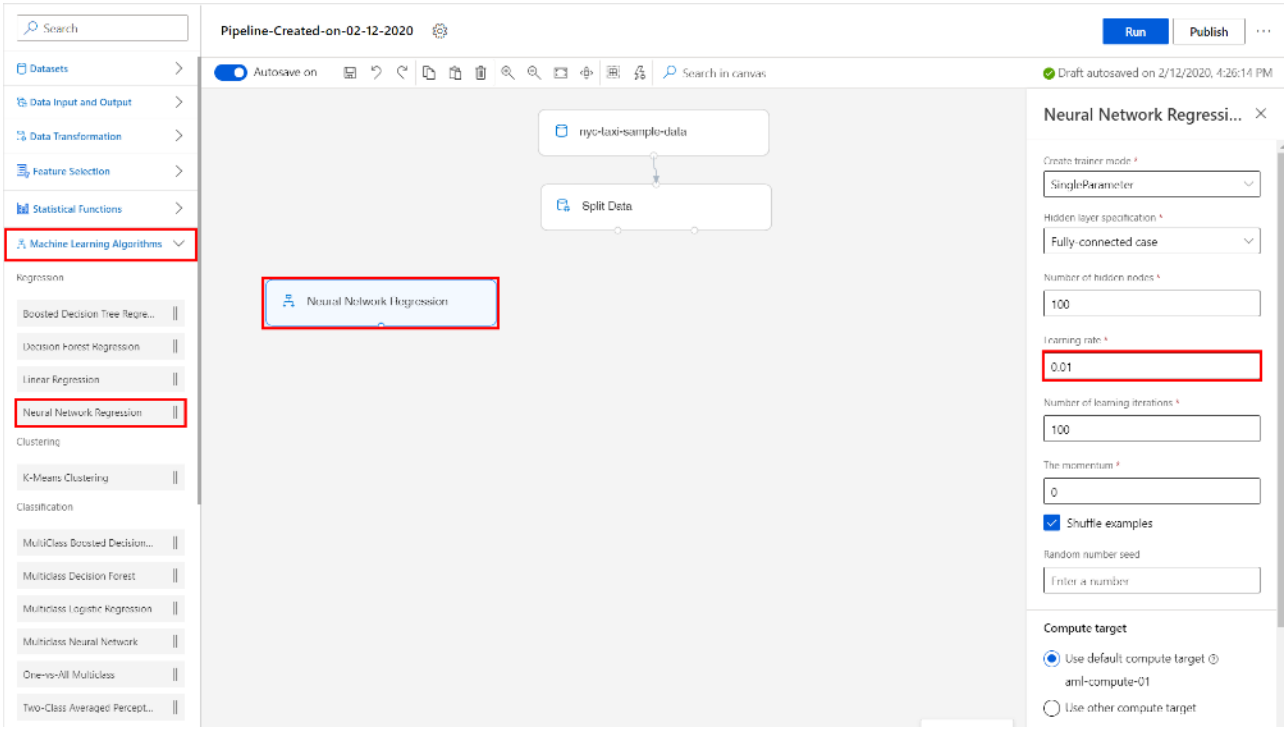
1. Select **Data Transformation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Split Data** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Fraction of rows in the first output dataset: **0.7**
    4. Connect the `Dataset` to the `Split Data` module



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.

Task 5: Initialize Regression Model

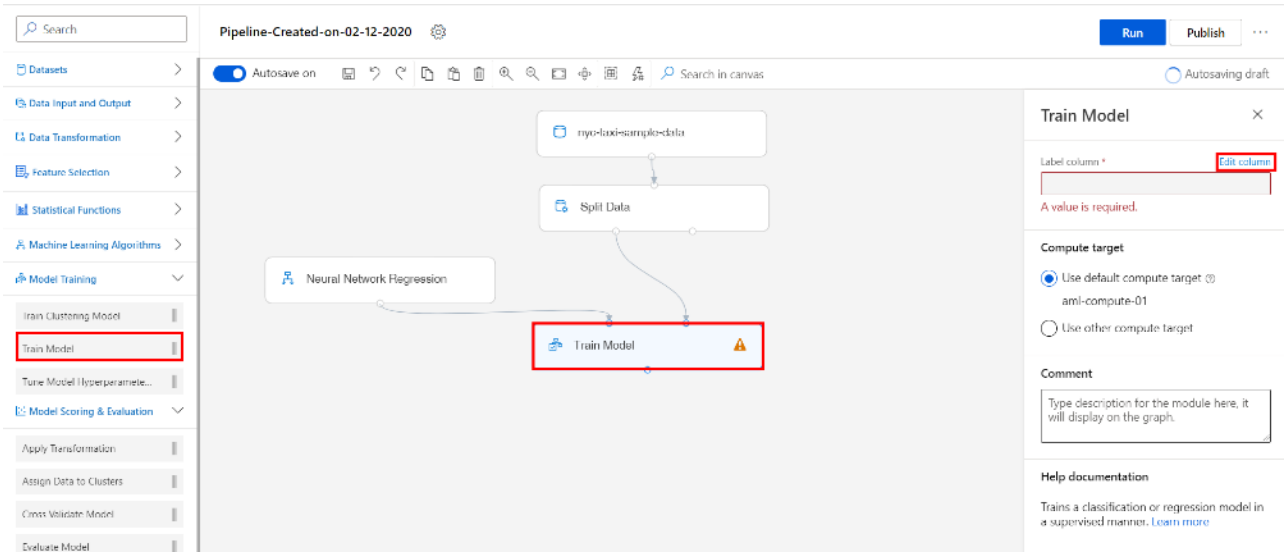
1. Select **Machine Learning Algorithms** section in the left navigation. Follow the steps outlined below:
    1. Select the **Neural Network Regression** prebuilt module, in the Regression category.
    2. Drag and drop the selected module on to the canvas
    3. Create trainer mode: **Single Parameter**. This option indicates how you want the model to be trained.
    4. Hidden layer specification: **Fully connected case**.
    5. For Learning rate: **0.01**.



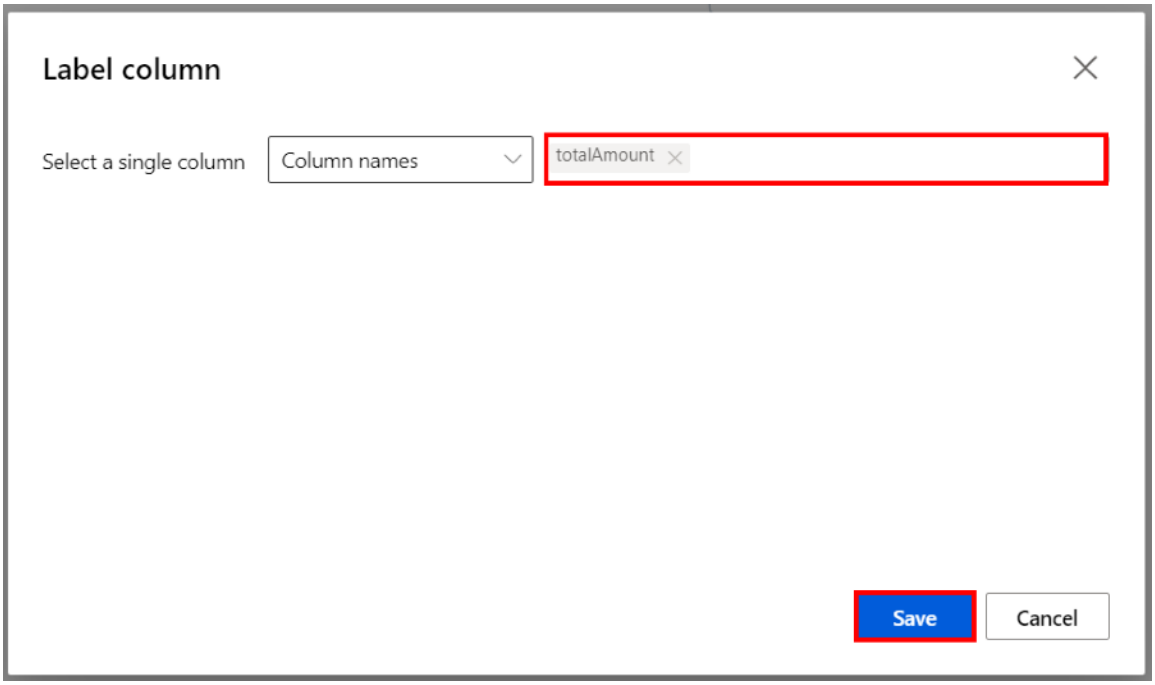
Note: Because the number of nodes in the input layer is determined by the number of features in the training data, in a regression model there can be only one node in the output layer.

Task 6: Setup Train Model Module

1. Select **Model Training** section in the left navigation. Follow the steps outlined below:
    1. Select the **Train Model** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the `Neural Network 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
    5. Select the **Edit column** link to open the `Label column` editor

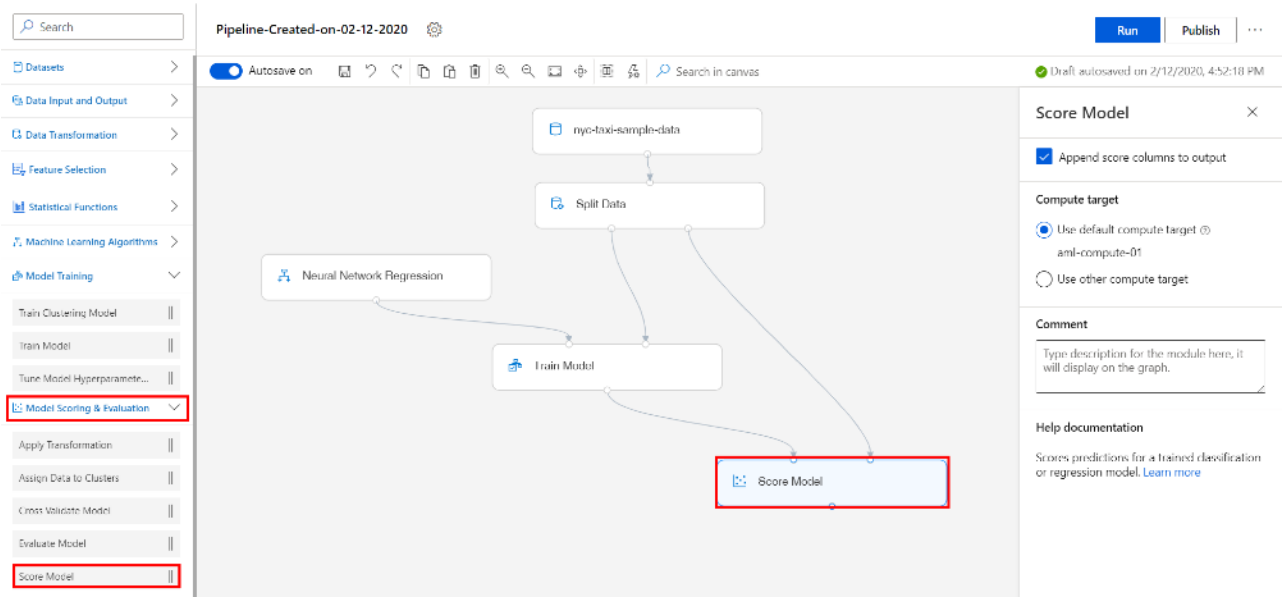


6. The `Label column` editor allows you to specify your `Label or Target column`. Type in the label column name **totalAmount** and then select **Save**.



Task 7: Setup Score Model Module

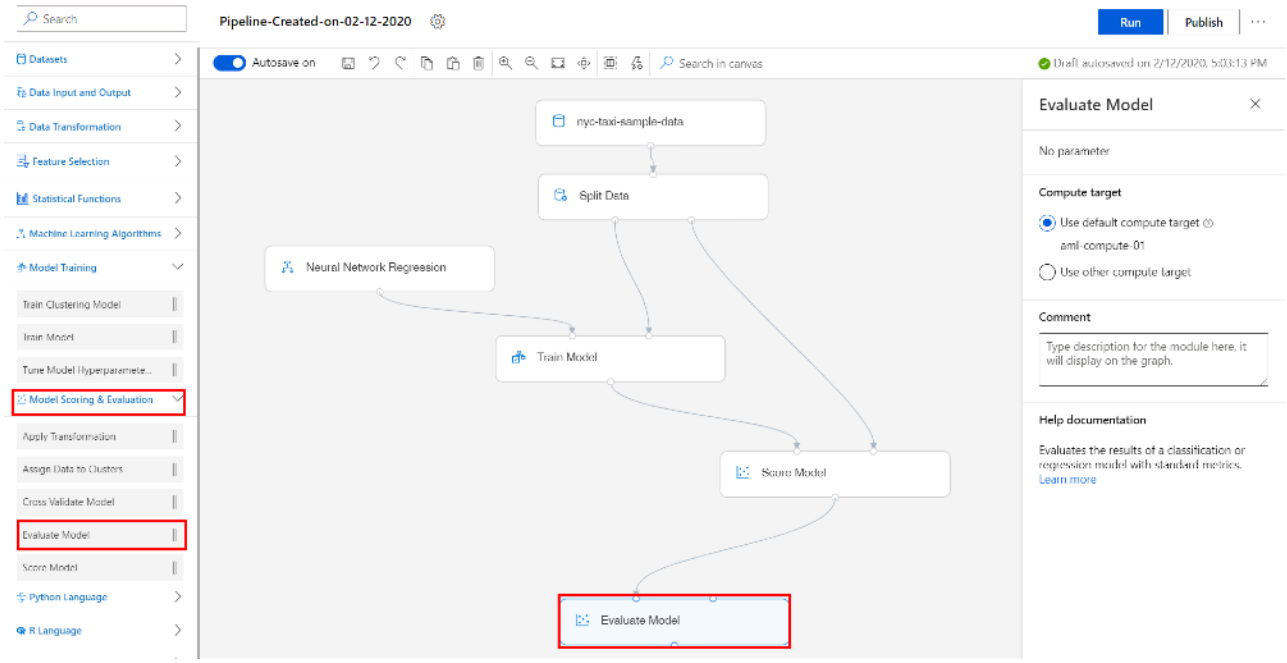
1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Score Model** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the `Train Model` module to the first input of the `Score Model` module
    4. Connect the second output of the `Split Data` module to the second input of the `Score Model` module



Note that *`Split Data`* module will feed data for both model training and model scoring. The first output (0.7 fraction) will connect with the *`Train Model`* module and the second output (0.3 fraction) will connect with the *`Score Model`* module.

Task 8: Setup Evaluate Model Module

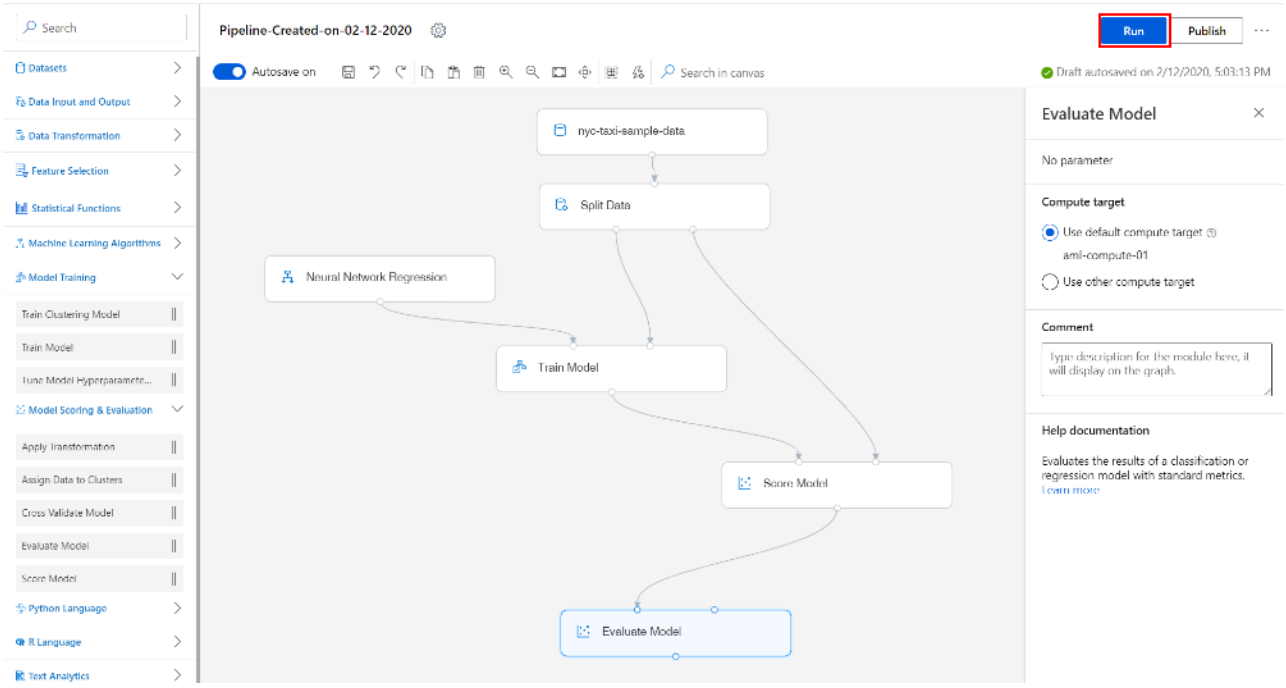
1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Evaluate Model** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the `Score Model` module to the first input of the `Evaluate Model` module



Exercise 3: Submit Training Pipeline

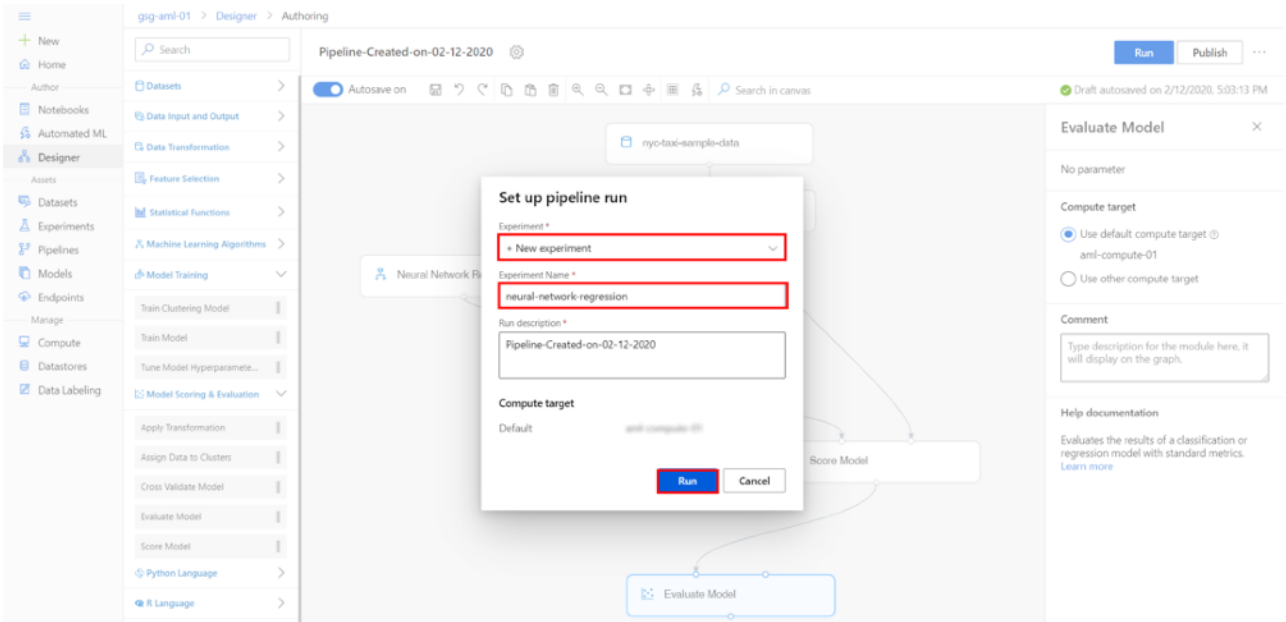
Task 1: Create Experiment and Submit Pipeline

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



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

3. In the `Setup pipeline run editor`, select **Experiment, Create new** and provide `New experiment name:` **neural-network-regression**, and then select **Submit**.

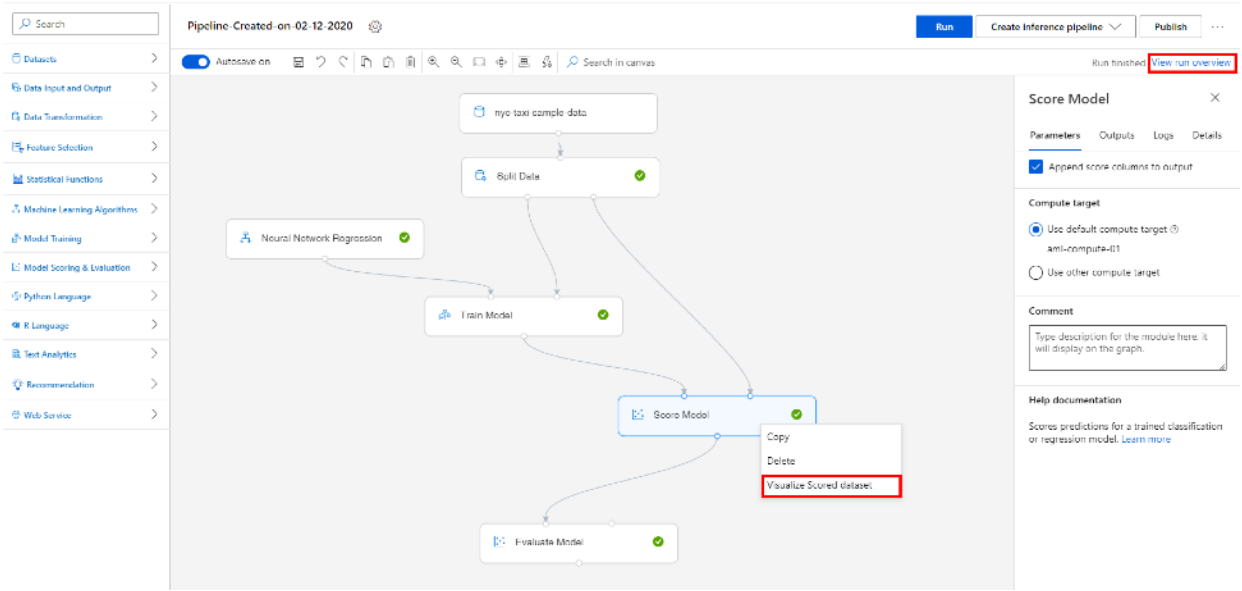


4. Wait for pipeline run to complete. It will take around **8 minutes** to complete the run.
5. While you wait for the model training to complete, you can learn more about the training algorithm used in this lab by selecting Neural Network Regression module.

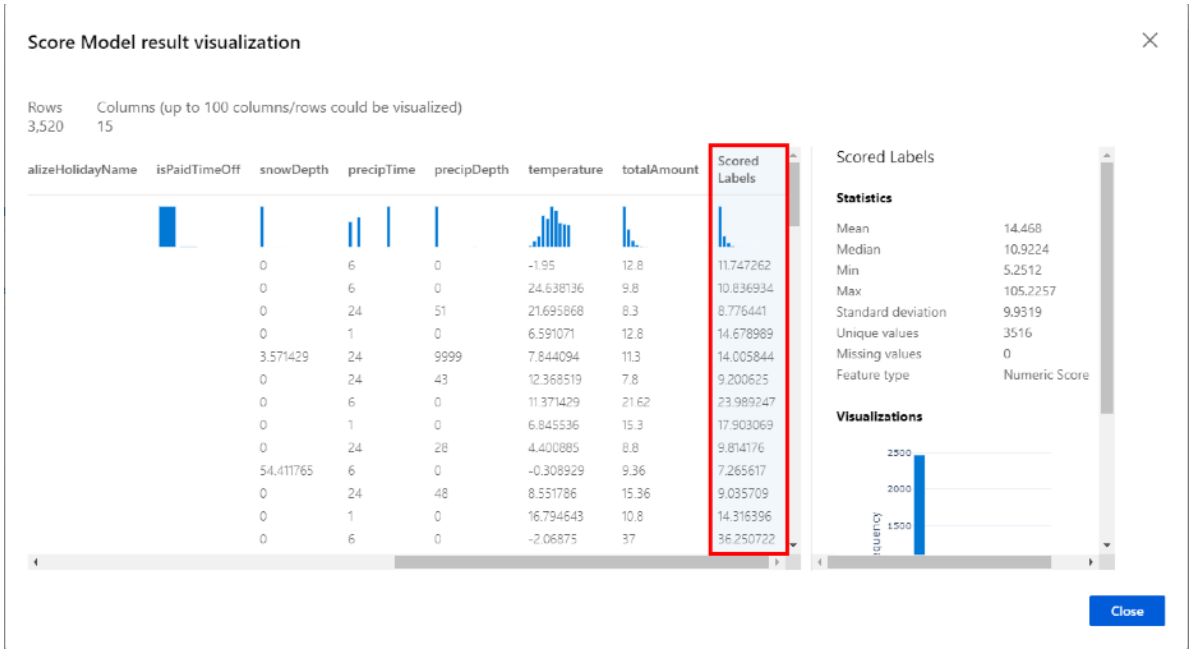
Exercise 4: Visualize Training Results

Task 1: Visualize the Model Predictions

1. Select **Score Model, Outputs, Visualize** to open the `Score Model result visualization` dialog or just simply right-click the `Score Model` module and select **Visualize Scored Dataset**.

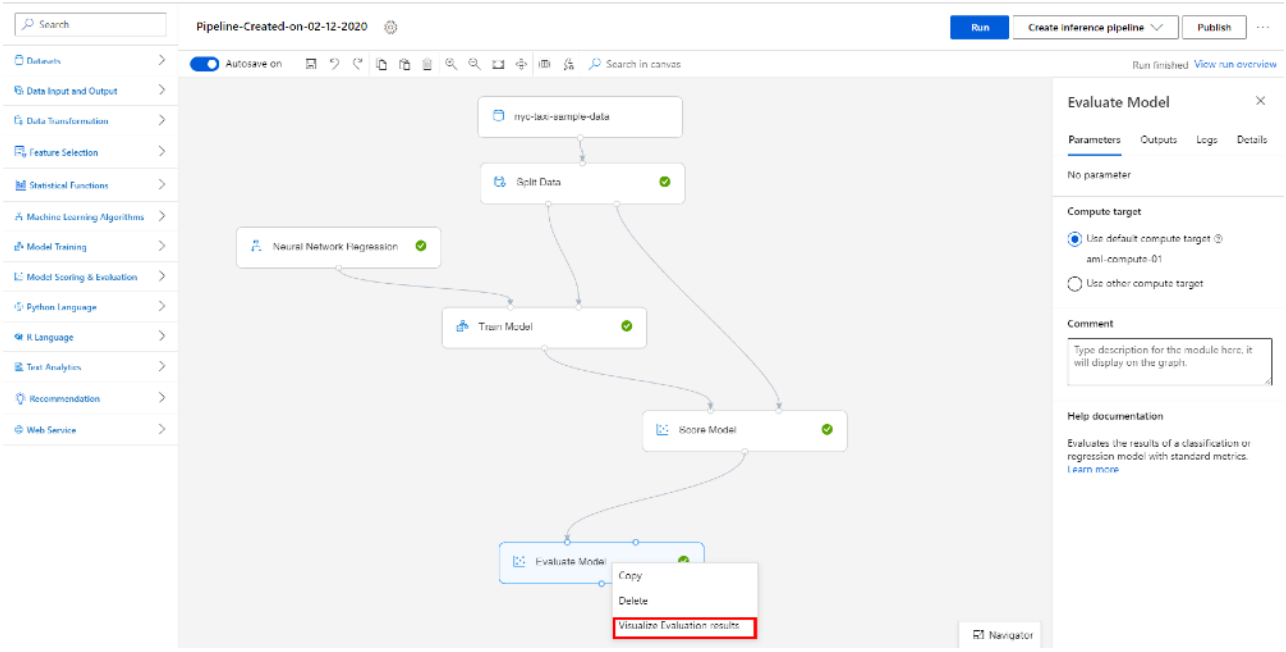


2. Observe the predicted values under the column **Scored Labels**. You can compare the predicted values (`Scored Labels`) with actual values (`totalAmount`).

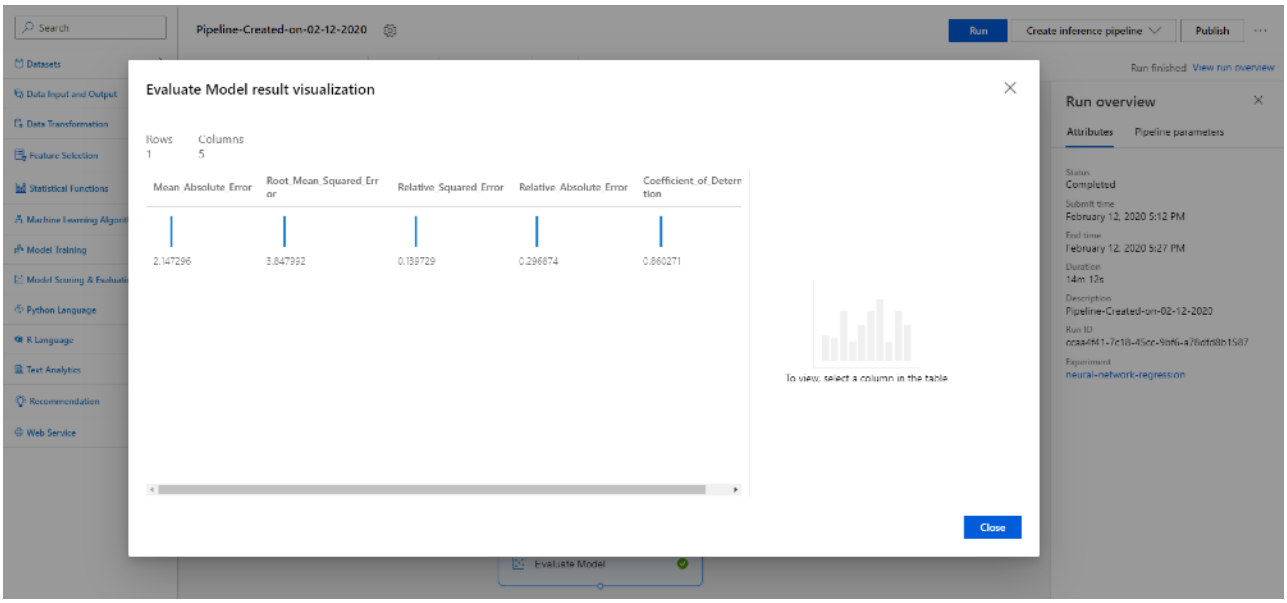


Task 2: Visualize the Evaluation Results

1. Select **Evaluate Model, Outputs, Visualize** to open the `Evaluate Model result visualization` dialog or just simply right-click the `Evaluate Model` module and select **Visualize Evaluation Results**.



2. Evaluate the model performance by reviewing the various evaluation metrics, such as **Mean Absolute Error**, **Root Mean Squared Error**, etc.



Next Steps

Congratulations! You have trained a simple neural net model using the prebuilt Neural Network Regression module in the AML visual designer. 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.

Lab 15 - Train a simple recommender

The main aim of a recommendation system is to recommend one or more items to users of the system. Examples of an item to be recommended, might be a movie, restaurant, book, or song. In general, the user is an entity with item preferences such as a person, a group of persons, or any other type of entity you can imagine.
There are two principal approaches to recommender systems:

- The `content-based` approach, which makes use of features for both users and items. Users can be described by properties such as age or gender. Items can be described by properties such as the author or the manufacturer. Typical examples of content-based recommendation systems can be found on social matchmaking sites.
- The `Collaborative filtering` approach, which uses only identifiers of the users and the items. It is based on a matrix of ratings given by the users to the items. The main source of information about a user is the list the items they've rated and the similarity with other users who have rated the same items.

The SVD recommender module in Azure Machine Learning designer is based on the Single Value Decomposition algorithm. It uses identifiers of the users and the items, and a matrix of ratings given by the users to the items. It's a typical example of collaborative recommender.

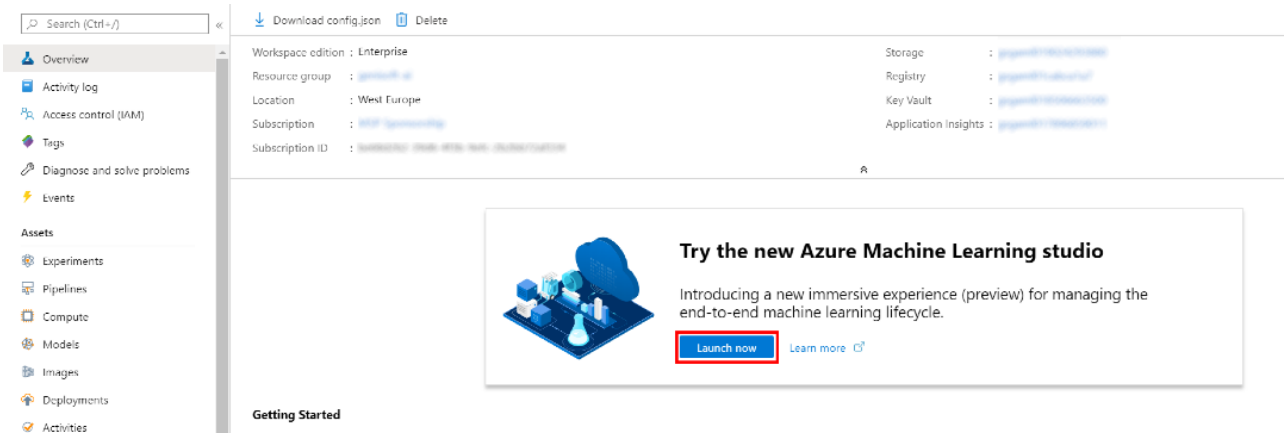
Overview

In this lab, we make use of the Train SVD Recommender module available in Azure Machine Learning designer (preview), to train a movie recommender engine. We use the collaborative filtering approach: the model learns from a collection of ratings made by users on a subset of a catalog of movies. Two open datasets available in Azure Machine Learning designer are used the `IMDB Movie Titles` dataset joined on the movie identifier with the `Movie Ratings` dataset. The Movie Ratings data consists of approximately 225,000 ratings for 15,742 movies by 26,770 users, extracted from Twitter using techniques described in the original paper by Dooms, De Pessemier and Martens. The paper and data can be found on GitHub.
We will both train the engine and score new data, to demonstrate the different modes in which a recommender can be used and evaluated. The trained model will predict what rating a user will give to unseen movies, so we'll be able to recommend movies that the user is most likely to enjoy. We will do all of this from the Azure Machine Learning designer without writing a single line of code.

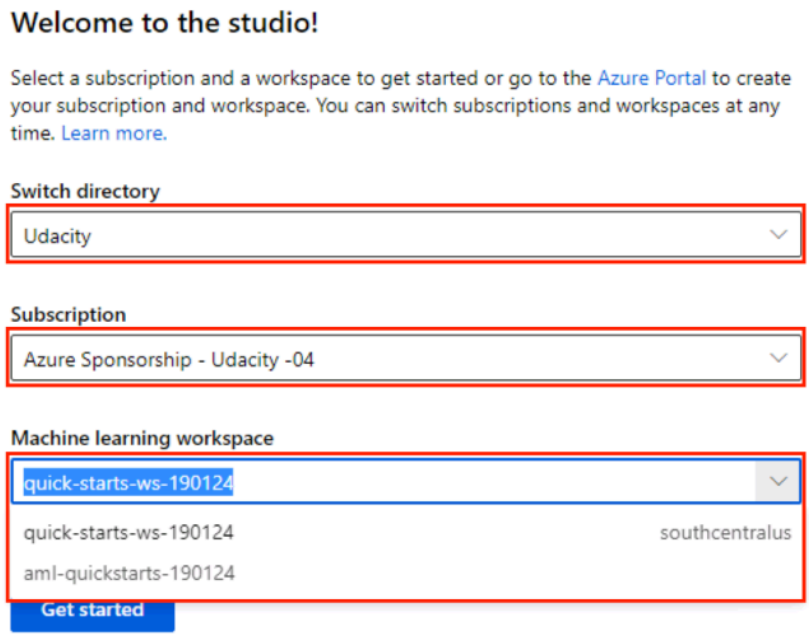
Exercise 1: Create New Training Pipeline

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.

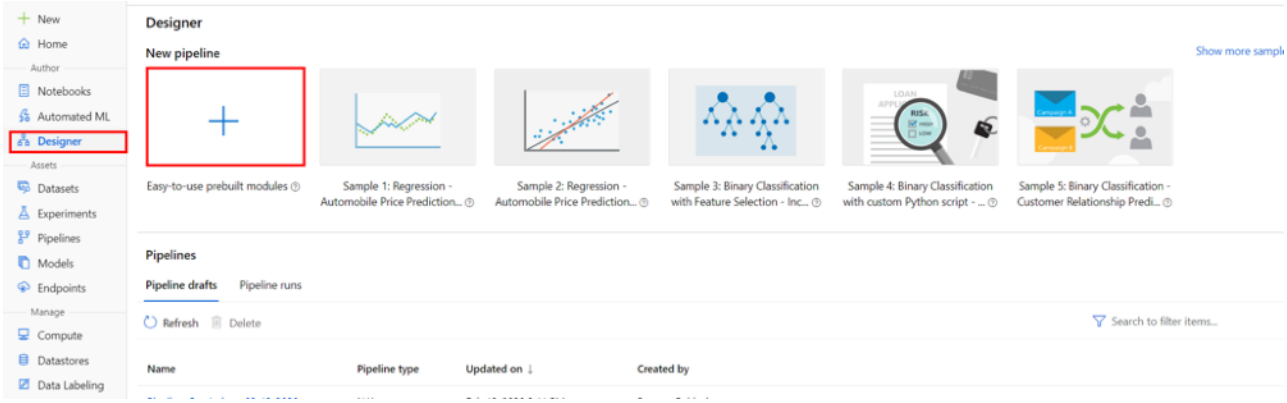


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



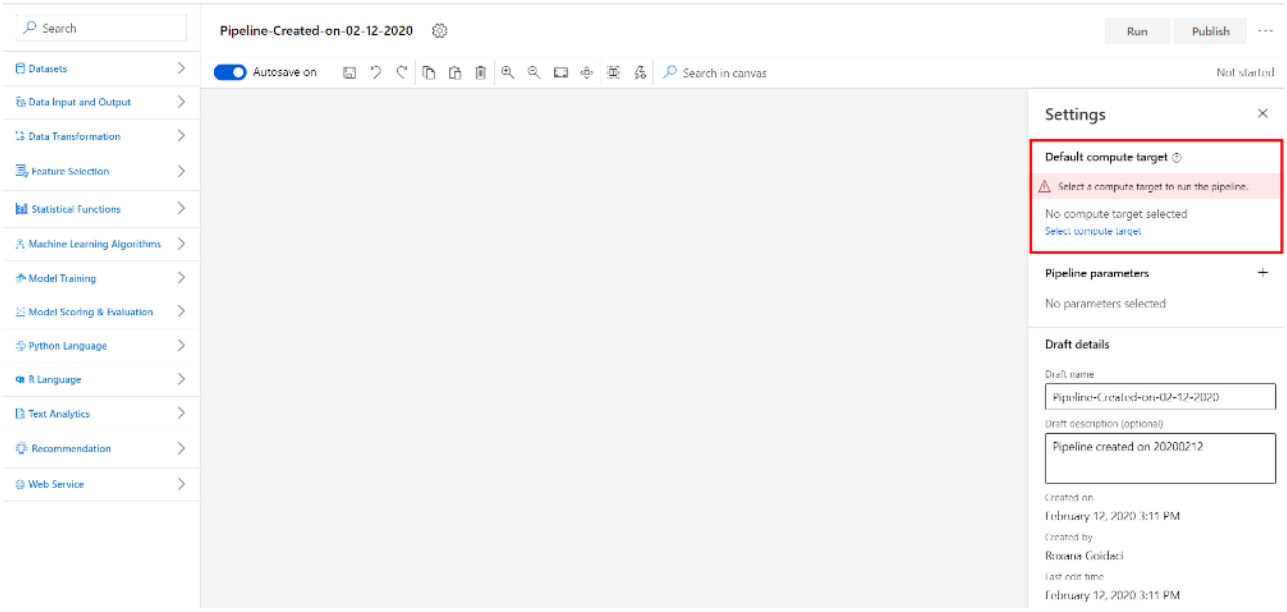
For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

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



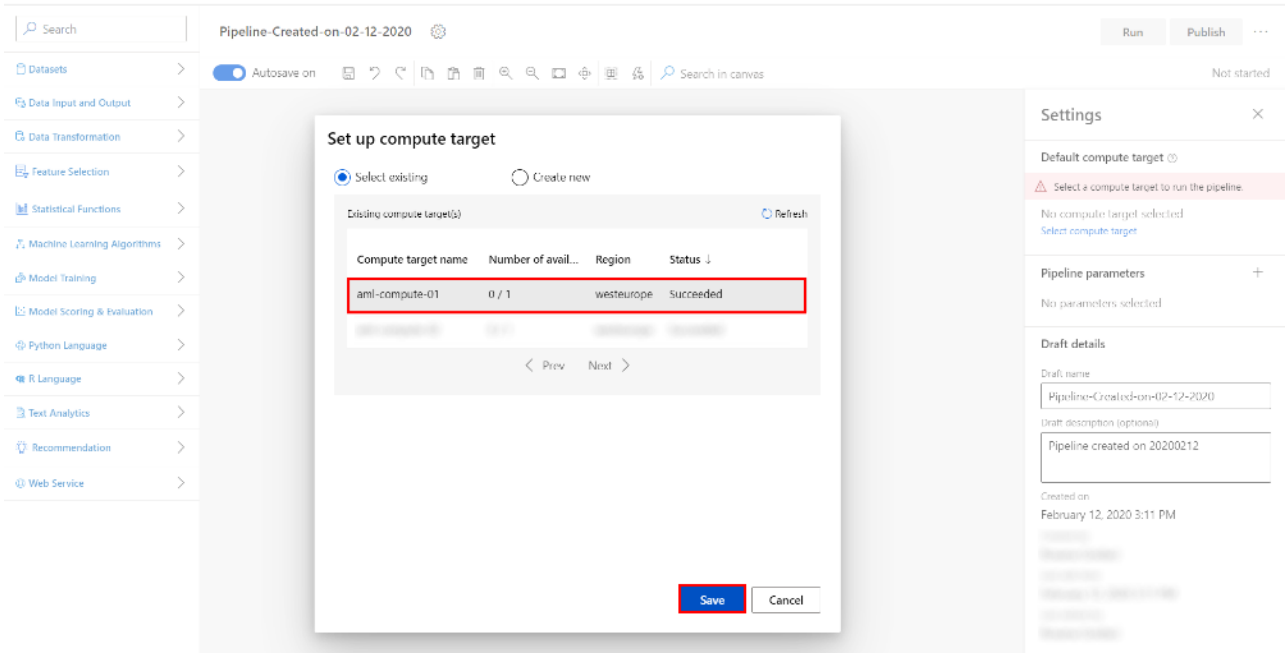
Task 2: Setup Compute Target

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



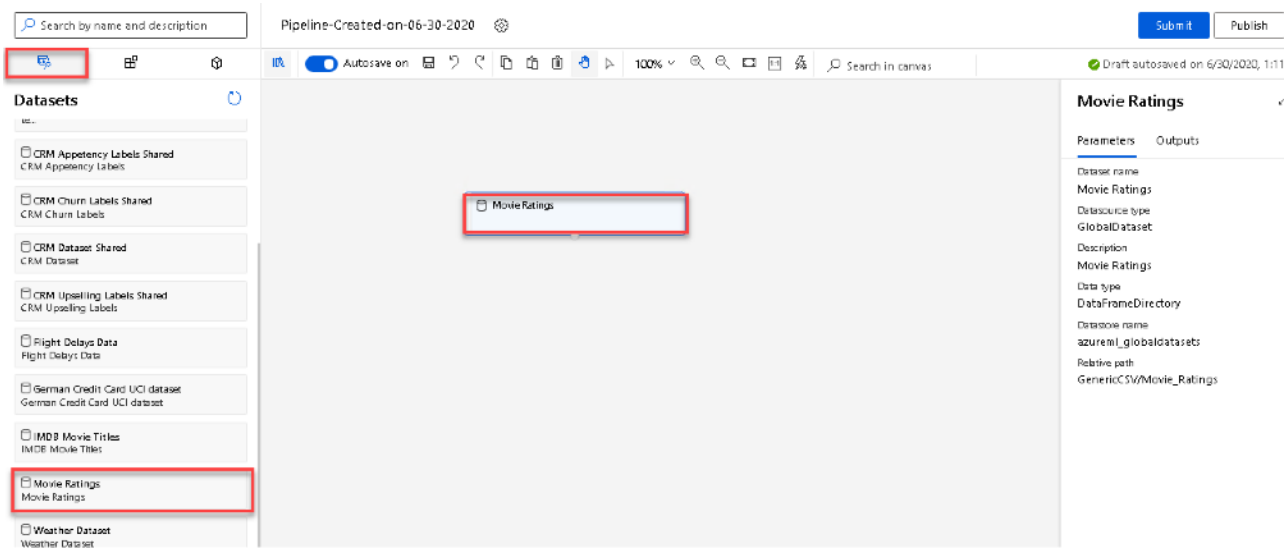
2. In the `Set up compute target` editor, select the available compute, and then select **Save**.

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.

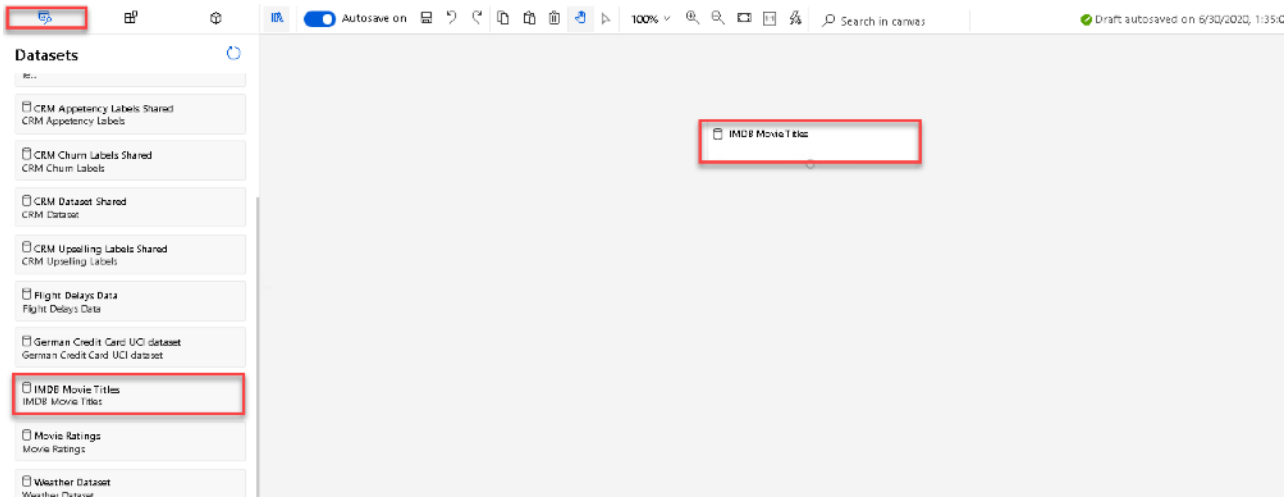

Search
Datasets
Data Input and Output
Data Transformation
Feature Selection
Statistical Functions
Machine Learning Algorithms
Model Training
Model Scoring & Evaluation
Python Language
R Language
Text Analytics
Recommendation
Web Service
Pipeline-Created-on-02-12-2020
Run
Publish
Autosave on
Search in canvas
Not started
Set up compute target
Select existing
Create new
Existing compute target(s)
Refresh
Compute target name
Number of avail...
Region
Status
aml-compute-01
0 / 1
westeurope
Succeeded
Prev
Next
Save
Cancel
Settings
Default compute target
Select a compute target to run the pipeline.
No compute target selected
Select compute target
Pipeline parameters
No parameters selected
Draft details
Draft name
Pipeline-Created-on-02-12-2020
Draft description (optional)
Pipeline created on 20200212
Created on
February 12, 2020 3:11 PM

Task 3: Add Sample Datasets

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

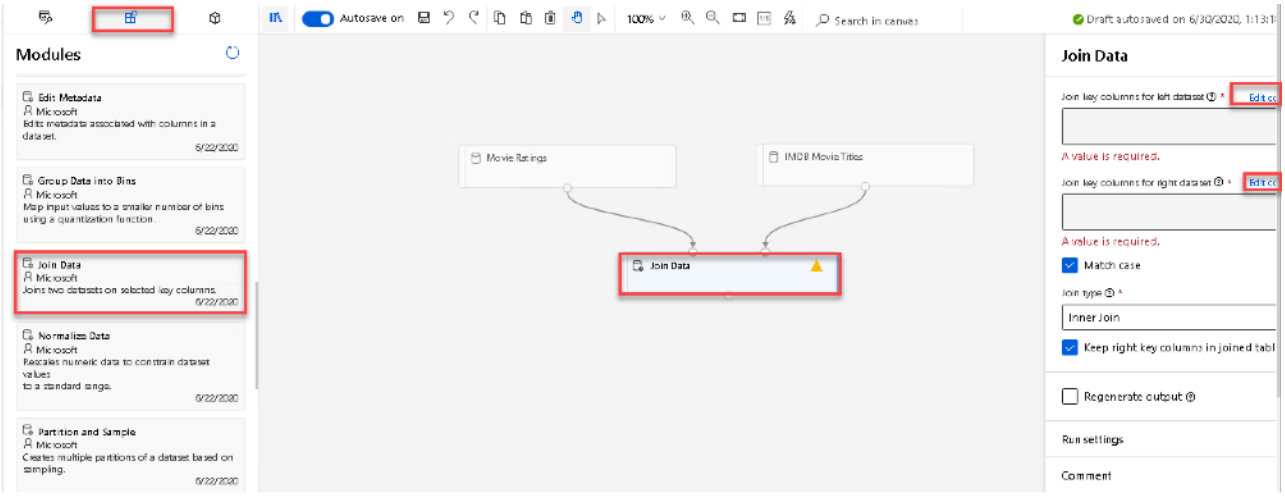


2. Select **Datasets** section in the left navigation. Next, select **Samples, IMDB Movie Titles** and drag and drop the selected dataset on to the canvas.

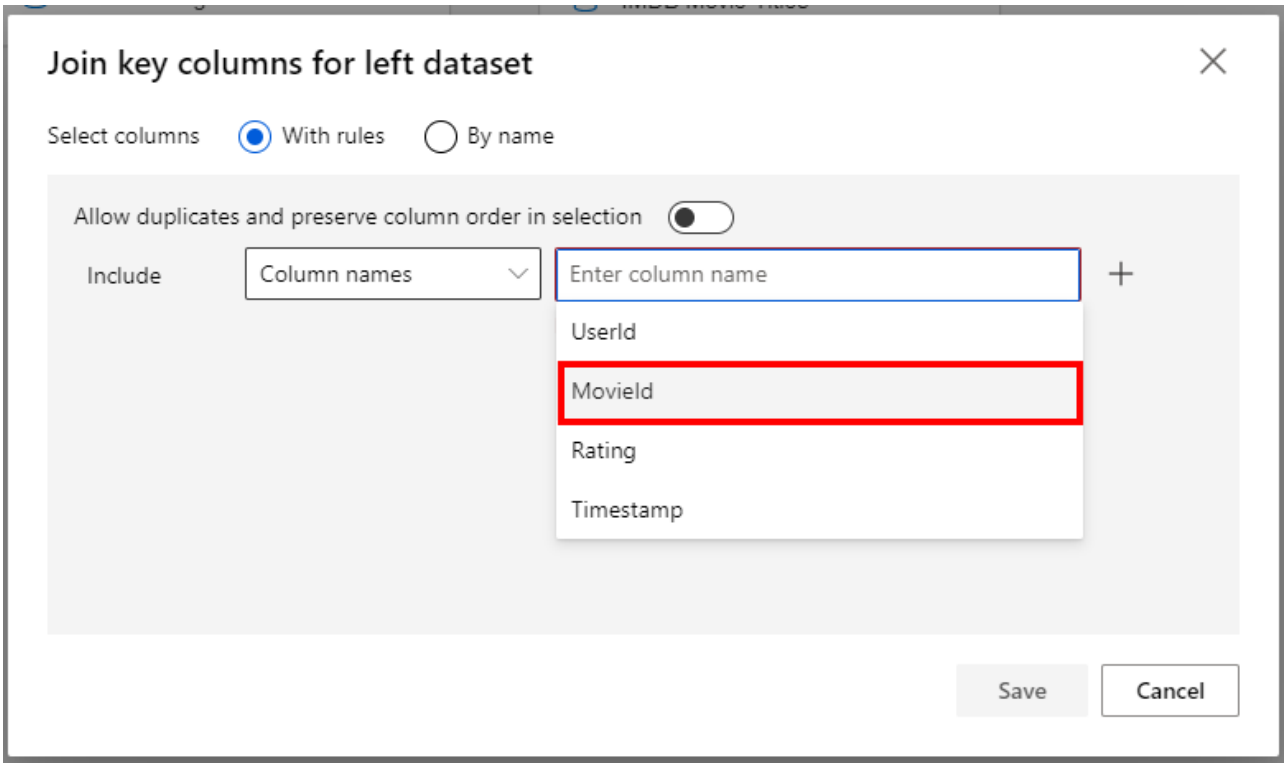


Task 4: Join the two datasets on Movie ID

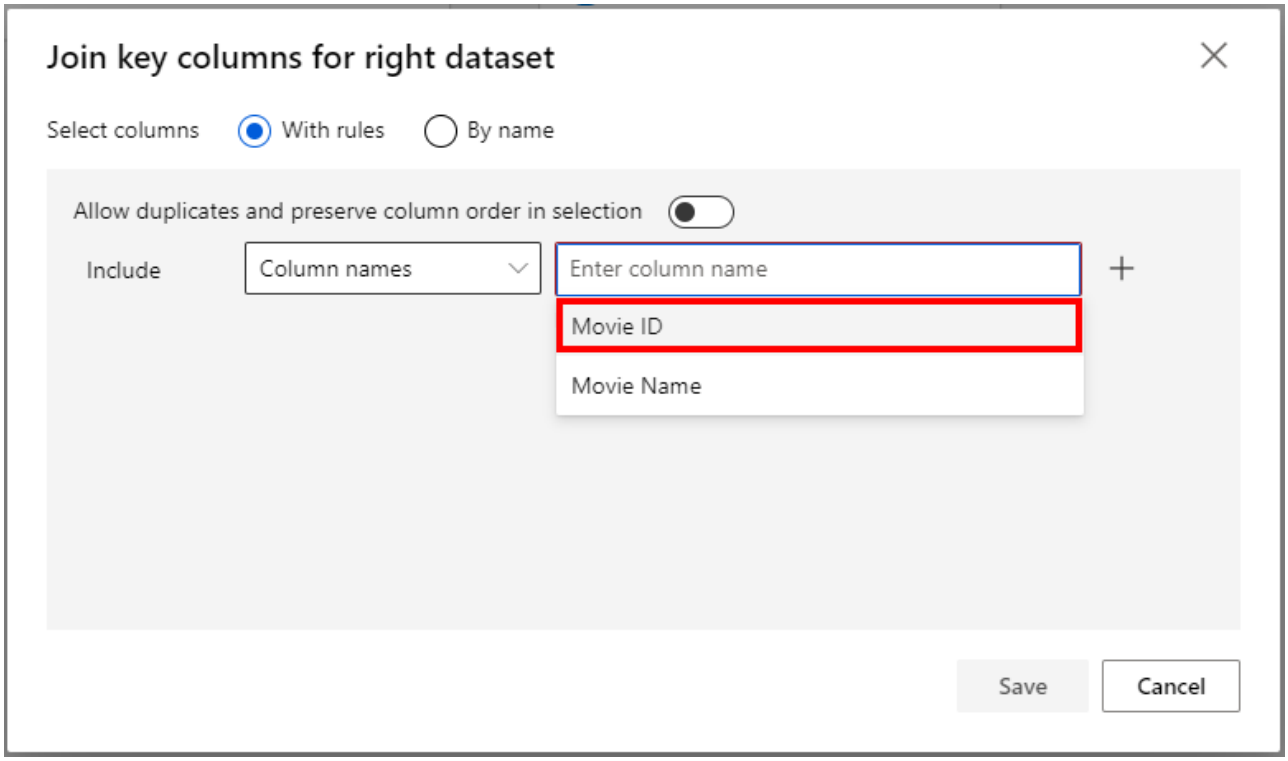
1. Select **Data Transformation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Join Data** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the output of the `Movie Ratings` module to the first input of the `Join Data` module.
    4. Connect the output of the `IMDB Movie Titles` module to the second input of the `Join Data` module.



2. Select the `Join Data` module.
3. Select the **Edit column** link to open the `Join key columns for left dataset` editor. Select the **MovieId** column in the `Enter column name` field.



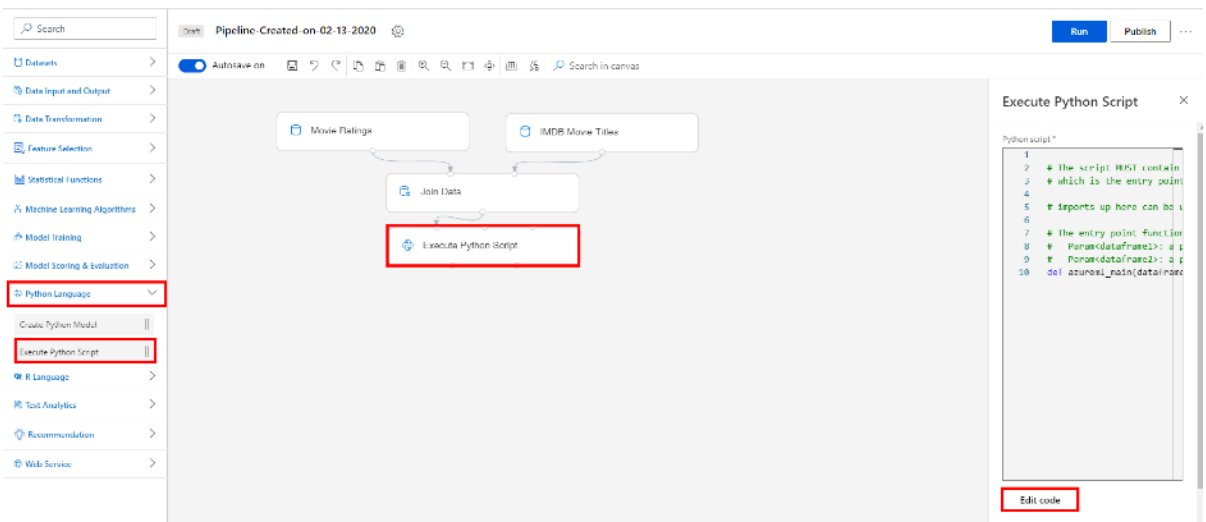
4. Select the **Edit column** link to open the `Join key columns for right dataset` editor. Select the **Movie ID** column in the `Enter column name` field.



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.

Task 5: Select Columns UserId, Movie Name, Rating using a Python script

1. Select **Python Language** section in the left navigation. Follow the steps outlined below:
    1. Select the **Execute Python Script** prebuilt module.
    2. Drag and drop the selected module on to the canvas.
    3. Connect the `Join Data` output to the input of the `Execute Python Script` module.



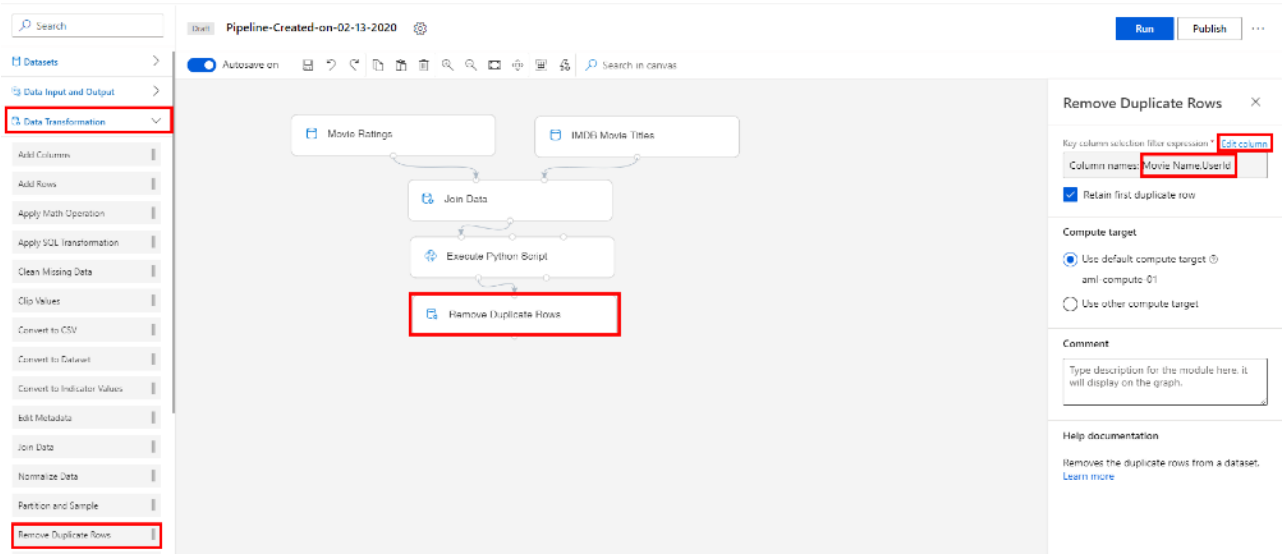
2. Select **Edit code** to open the `Python script` editor, clear the existing code and then enter the following lines of code to select the UserId, Movie Name, Rating columns from the joined dataset. Please ensure that there is no indentation for the first line and the second and third lines are indented.

```
def azureml_main(dataframe1 = None, dataframe2 = None):
    df1 = dataframe1[['UserId','Movie Name','Rating']]
    return df1,
```

Note: In other pipelines, for selecting a list of columns from a dataset, we could have used the `Select Columns from Dataset` prebuilt module. This one returns the columns in the same order as in the input dataset. This time we need the output dataset to be in the format: user id, movie name, rating.This column order is required at the input of the Train SVD Recommender module.

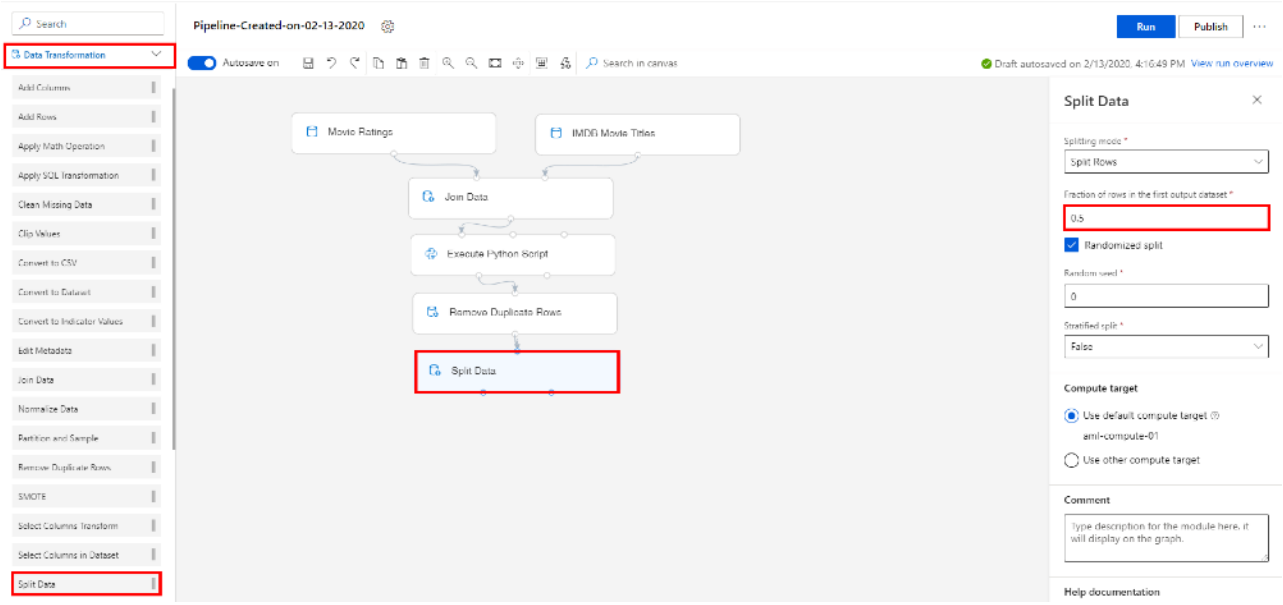
Task 6: Remove duplicate rows with same Movie Name and UserId

1. Select **Data Transformation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Remove Duplicate Rows** prebuilt module.
    2. Drag and drop the selected module on to the canvas.
    3. Connect the first output of the `Execute Python Script` to the input of the `Remove Duplicate Rows` module.
    4. Select the **Edit columns** link to open the `Select columns` editor and then enter the following list of columns to be included in the output dataset: **Movie Name**, **UserId**.



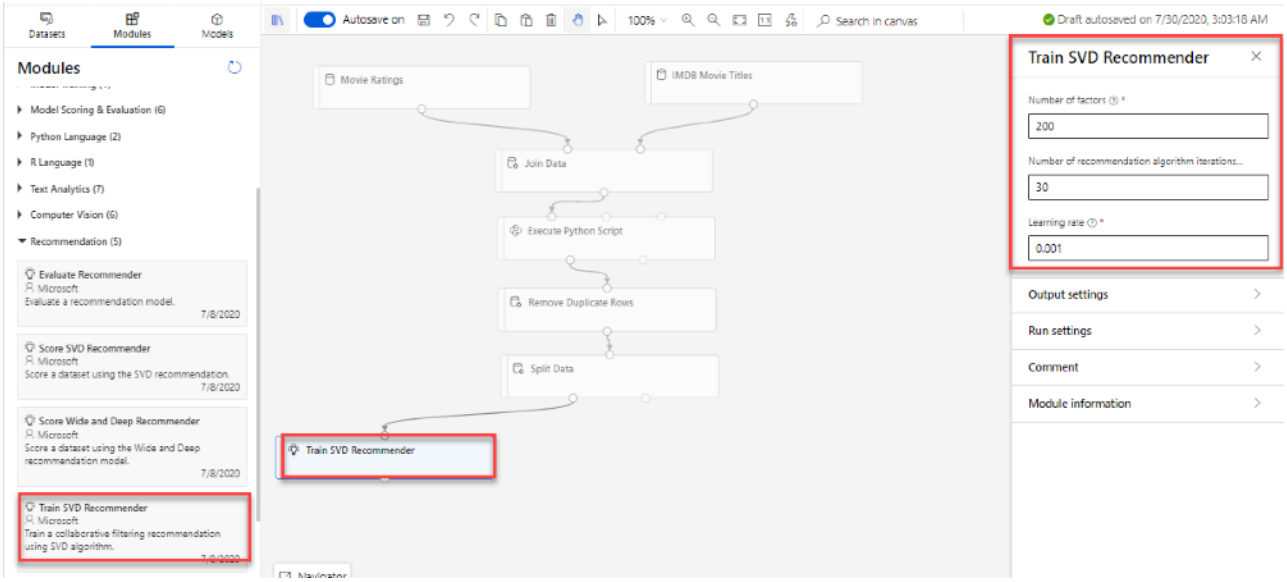
Task 7: Split the dataset into training set (0.5) and test set (0.5)

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. Fraction of rows in the first output dataset: **0.5**
    4. Connect the `Dataset` to the `Split Data` module



Task 8: Initialize Recommendation Module

1. Select **Recommendation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Train SVD Recommender** prebuilt module.
    2. Drag and drop the selected module on to the canvas
    3. Connect the first output of the `Split Data` module to the input of the `Train SVD Recommender` module
    4. Number of factors: **200**. This option specify the number of factors to use with the recommender. With the number of users and items increasing, it's better to set a larger number of factors. But if the number is too large, performance might drop.
    5. Number of recommendation algorithm iterations: **30**. This number indicates how many times the algorithm should process the input data. The higher this number is, the more accurate the predictions are. However, a higher number means slower training. The default value is 30.
    6. For Learning rate: **0.001**. The learning rate defines the step size for learning.

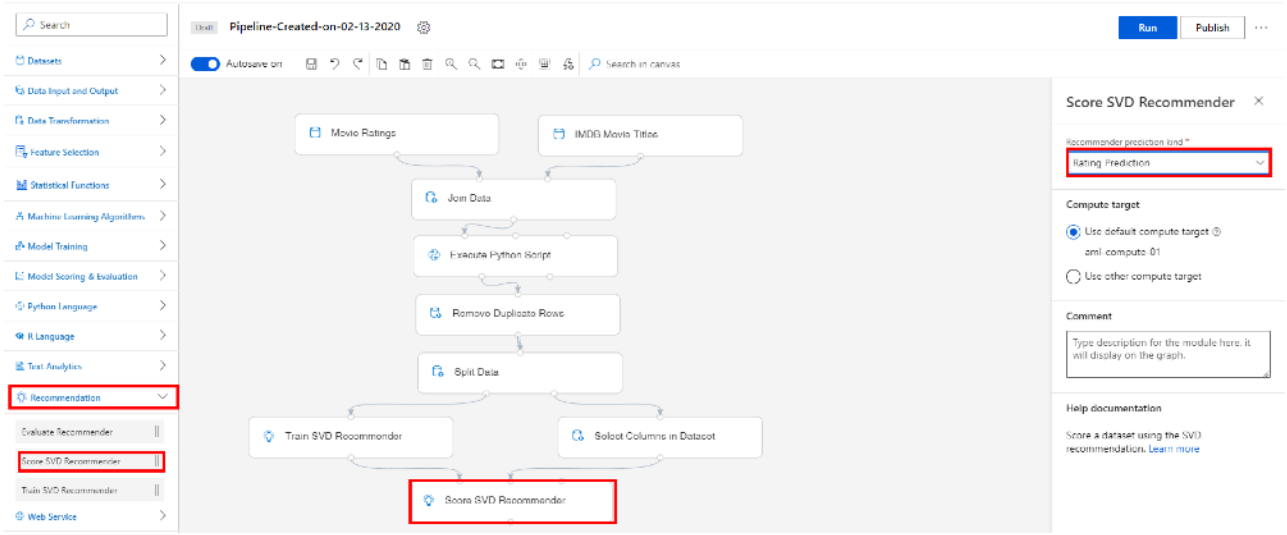


Task 9: Select Columns UserId, Movie Name from the test set

1. Select **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 `Split Data` second output to the input of the `Select columns in Dataset` module.
    4. Select the **Edit columns** link to open the `Select columns` editor and then enter the following list of columns to be included in the output dataset: **UserId**, **Movie Name**.

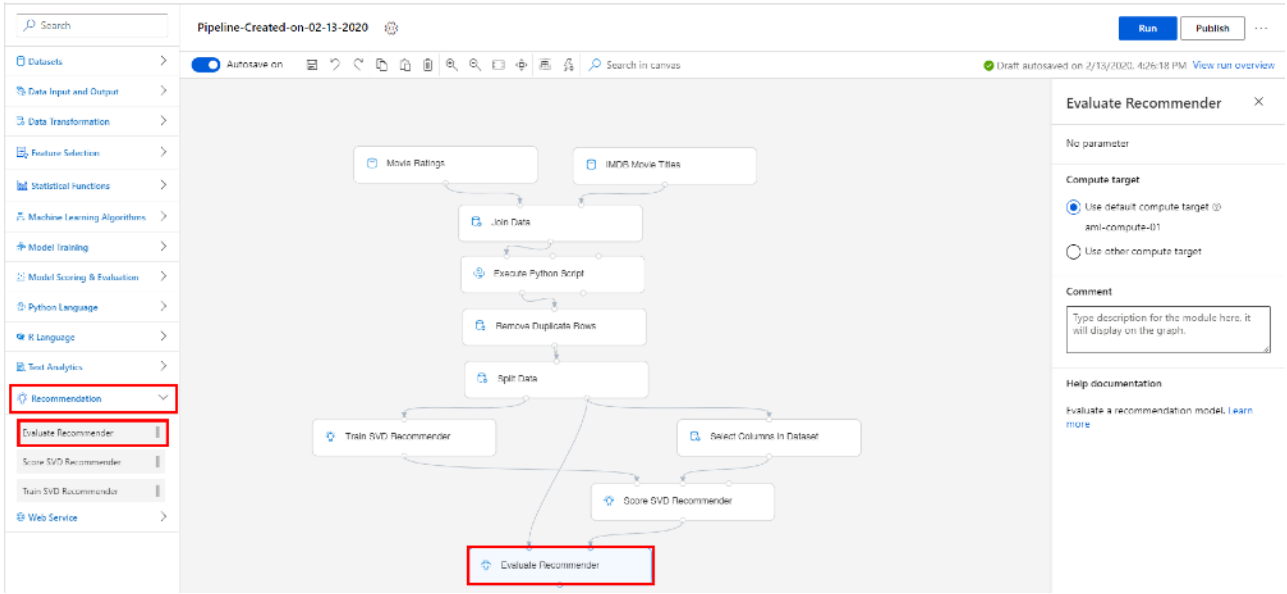
Task 10: Configure the Score SVD Recommender

1. Select **Recommendation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Score SVD Recommender** prebuilt module.
    2. Drag and drop the selected module on to the canvas
    3. Connect the output of the `Train SVD Recommender` module to the first input of the `Score SVD Recommender` module, which is the Trained SVD Recommendation input.
    4. Connect the output of the `Select Columns in Dataset` module to the second input of the `Score SVD Recommender` module, which is the Dataset to score input.
    5. Select the `Score SVD Recommender` module on the canvas.
    6. Recommender prediction kind: **Rating Prediction**. For this option, no other parameters are required. When you predict ratings, the model calculates how a user will react to a particular item, given the training data. The input data for scoring must provide both a user and the item to rate.



Task 11: Setup Evaluate Recommender Module

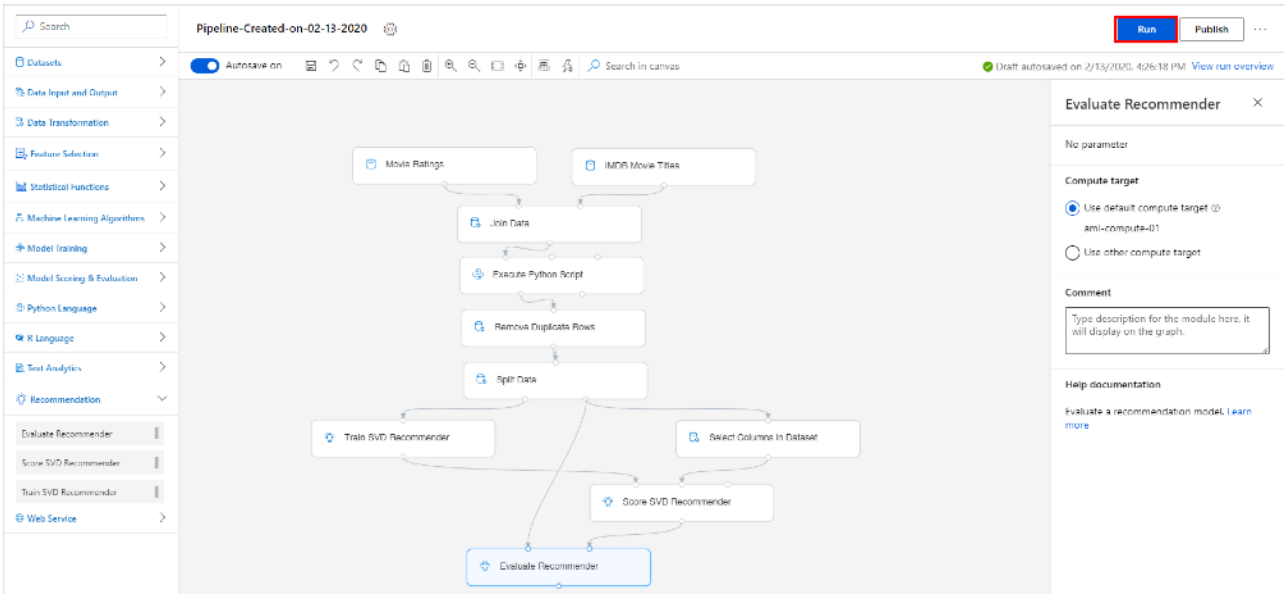
1. Select **Recommendation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Evaluate Recommender** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the `Score SVD Recommender` module to the second input of the `Evaluate Recommender` module, which is the Scored dataset input.
    4. Connect the second output of the `Split Data` module (train set) to the first input of the `Evaluate Recommender` module, which is the Test dataset input.



Exercise 2: Submit Training Pipeline

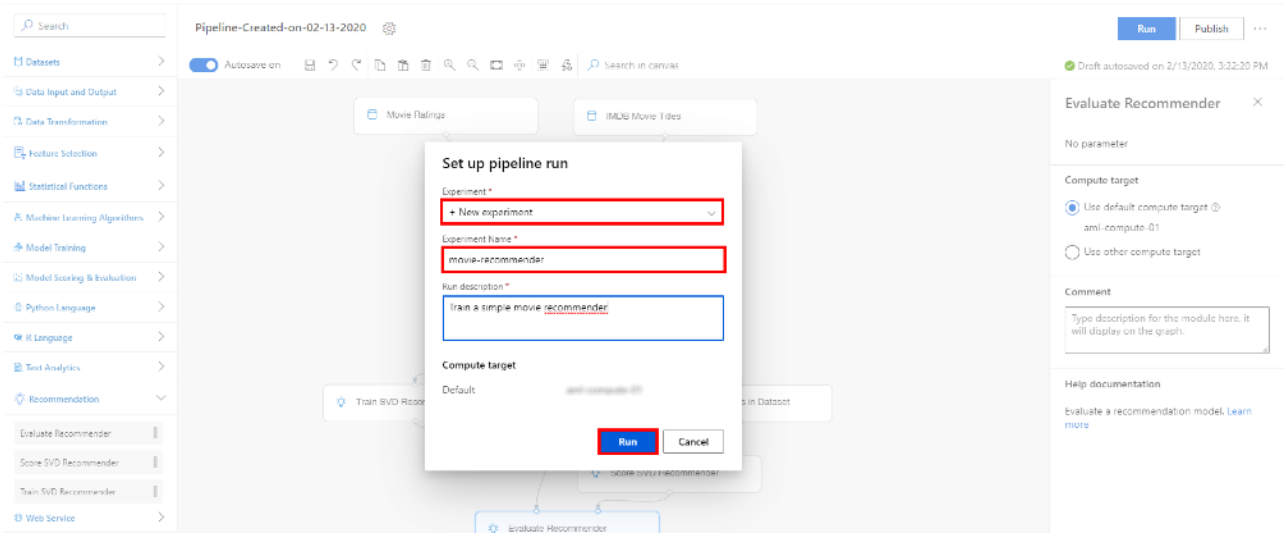
Task 1: Create Experiment and Submit Pipeline

1. Select **Submit** on the right corner of the canvas 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:` **movie-recommender**, and then select **Submit**.

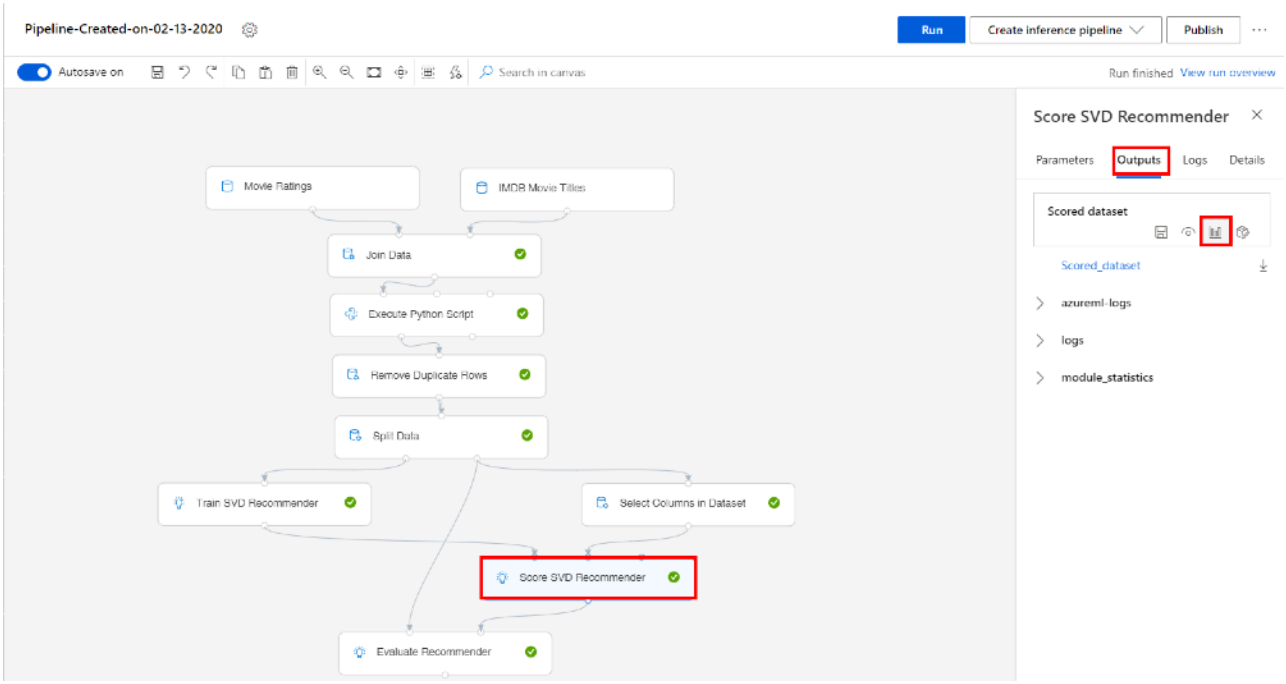


3. Wait for pipeline run to complete. It will take around **20 minutes** to complete the run.
4. While you wait for the model training to complete, you can learn more about the SVD algorithm used in this lab by selecting Train SVD Recommender.

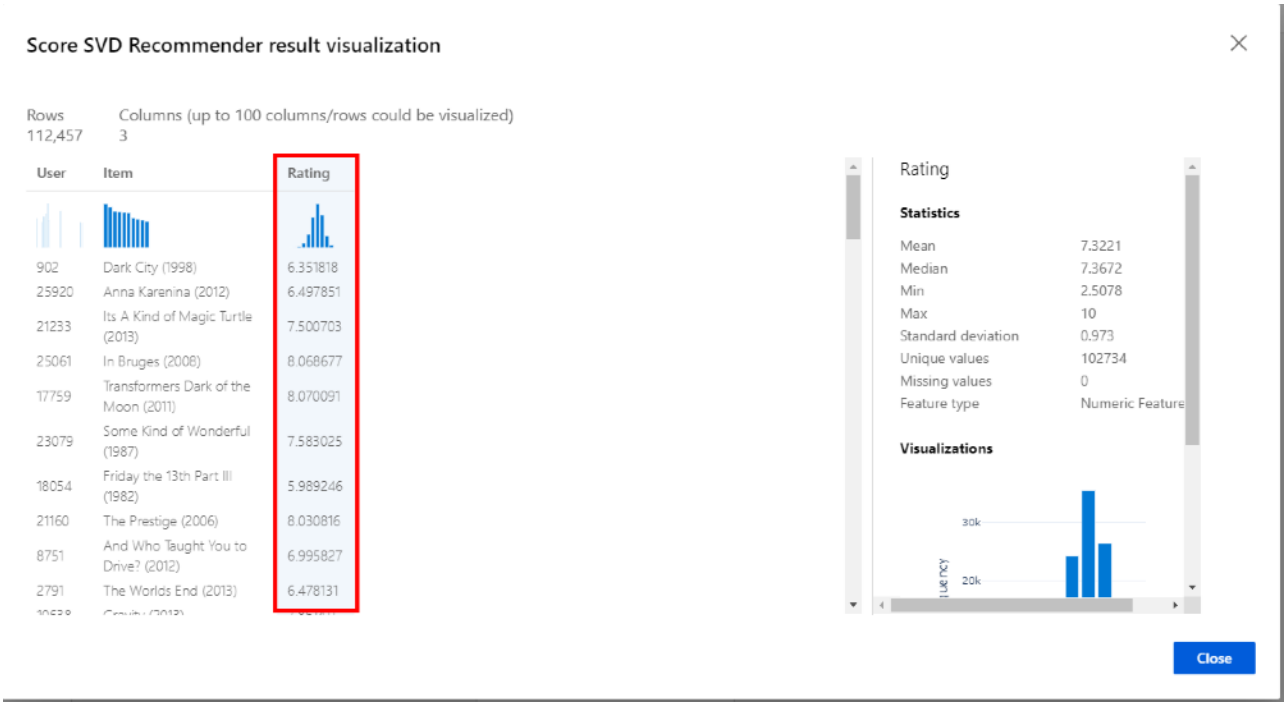
Exercise 3: Visualize Scoring Results

Task 1: Visualize the Scored dataset

1. Select **Score SVD Recommender, Outputs, Visualize** to open the `Score SVD Recommender result visualization` dialog or just simply right-click the `Score SVD Recommender` module and select **Visualize Scored dataset**.

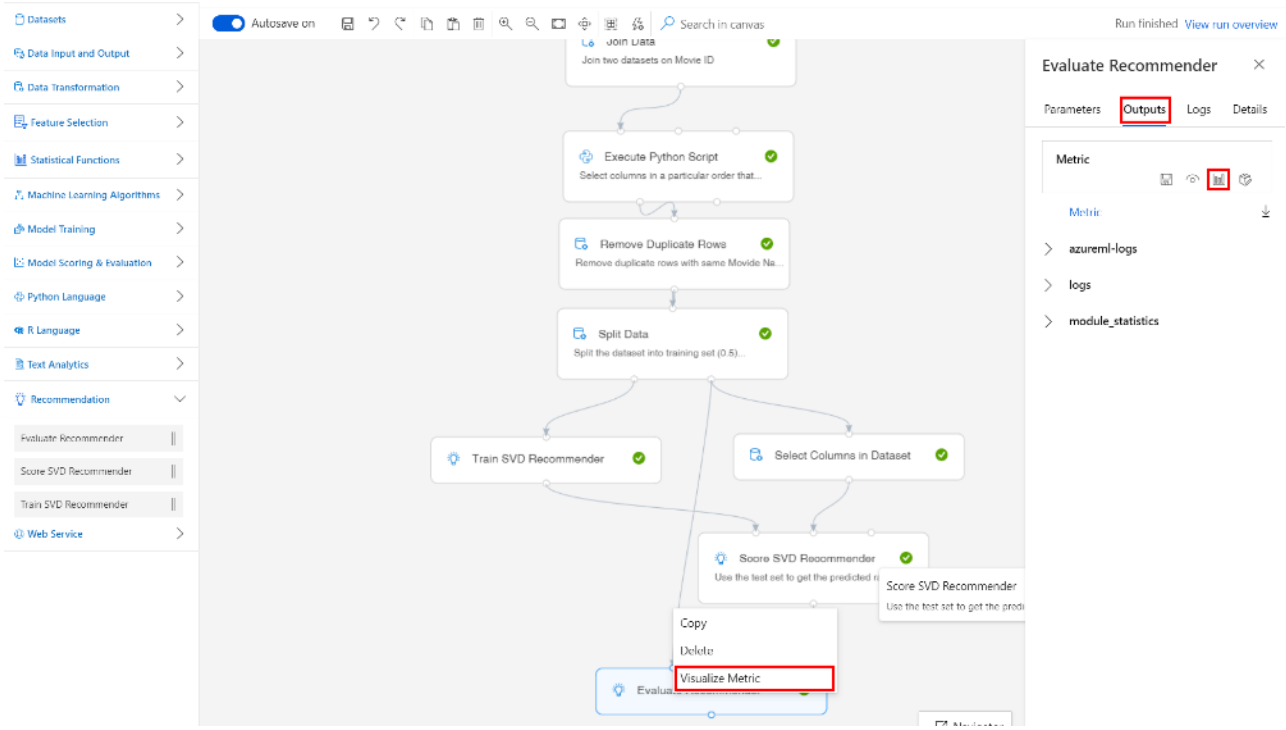


2. Observe the predicted values under the column **Rating**.



Task 2: Visualize the Evaluation Results

1. Select **Evaluate Recommender, Outputs, Visualize** to open the `Evaluate Recommender result visualization` dialog or just simply right-click the `Evaluate Recommender` module and select **Visualize Evaluation Results**.



2. Evaluate the model performance by reviewing the various evaluation metrics, such as **Mean Absolute Error**, **Root Mean Squared Error**, etc.

Next Steps

Congratulations! You have trained a simple movie recommender using the prebuilt Recommender modules in the AML visual designer. 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.

Lab 16 - Train a simple text classifier

In text classification scenarios, the goal is to assign a piece of text, such as a document, a news article, a search query, an email, a tweet, support tickets, customer feedback, user product review, to predefined classes or categories. Some examples of text classification applications are: categorizing newspaper articles into topics, organizing web pages into hierarchical categories, spam email filtering, sentiment analysis, predicting user intent from search queries, support tickets routing, and customer feedback analysis.

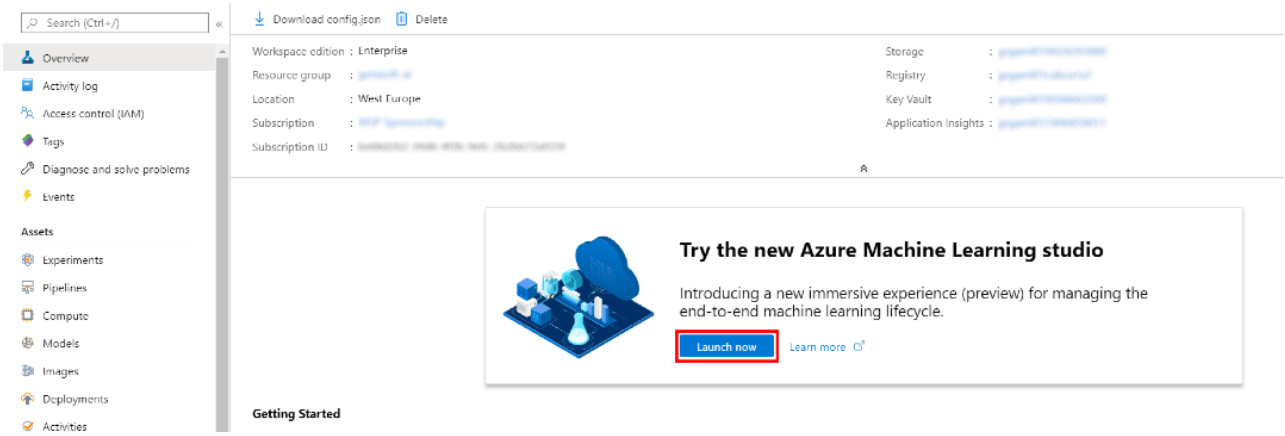
Overview

In this lab we demonstrate how to use text analytics modules available in Azure Machine Learning designer (preview) to build a simple text classification pipeline. We will create a training pipeline and initialize a **multiclass logistic regression classifier** to predict the company category with Wikipedia SP 500 dataset derived from Wikipedia. The dataset manages articles of each S&P 500 company. Before uploading to Azure Machine Learning designer, the dataset was processed as follows: extracted text content for each specific company, removed wiki formatting, removed non-alphanumeric characters, converted all text to lowercase, known company categories added. Articles could not be found for some companies, so that's why the number of records is less than 500.

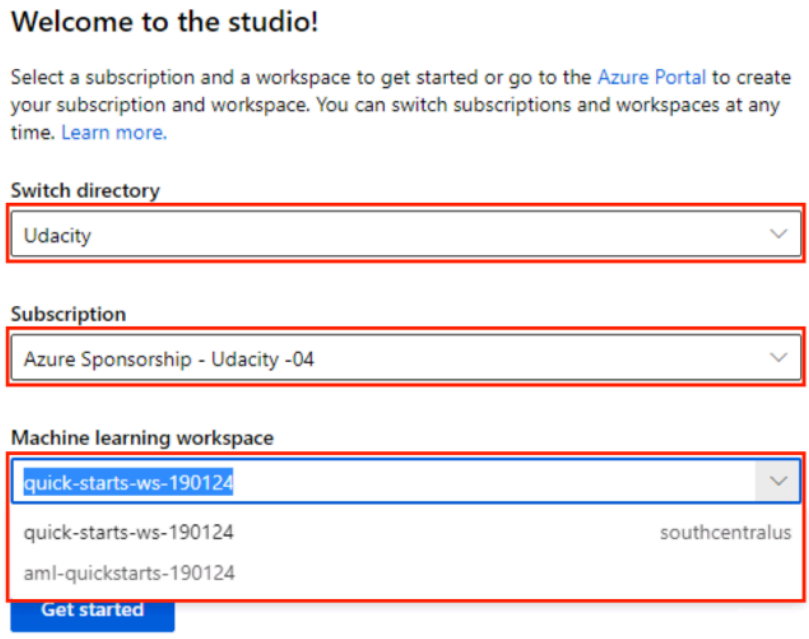
Exercise 1: Create New Training Pipeline

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.

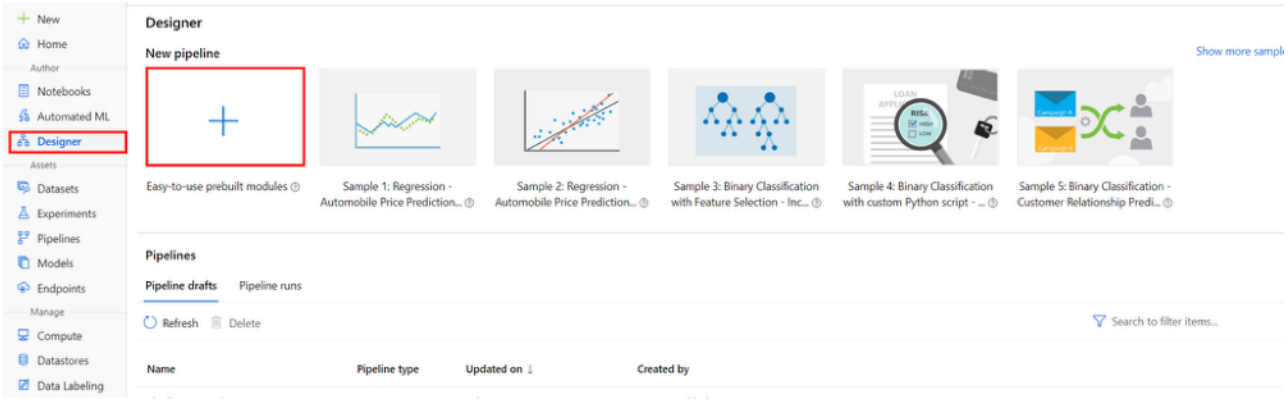


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



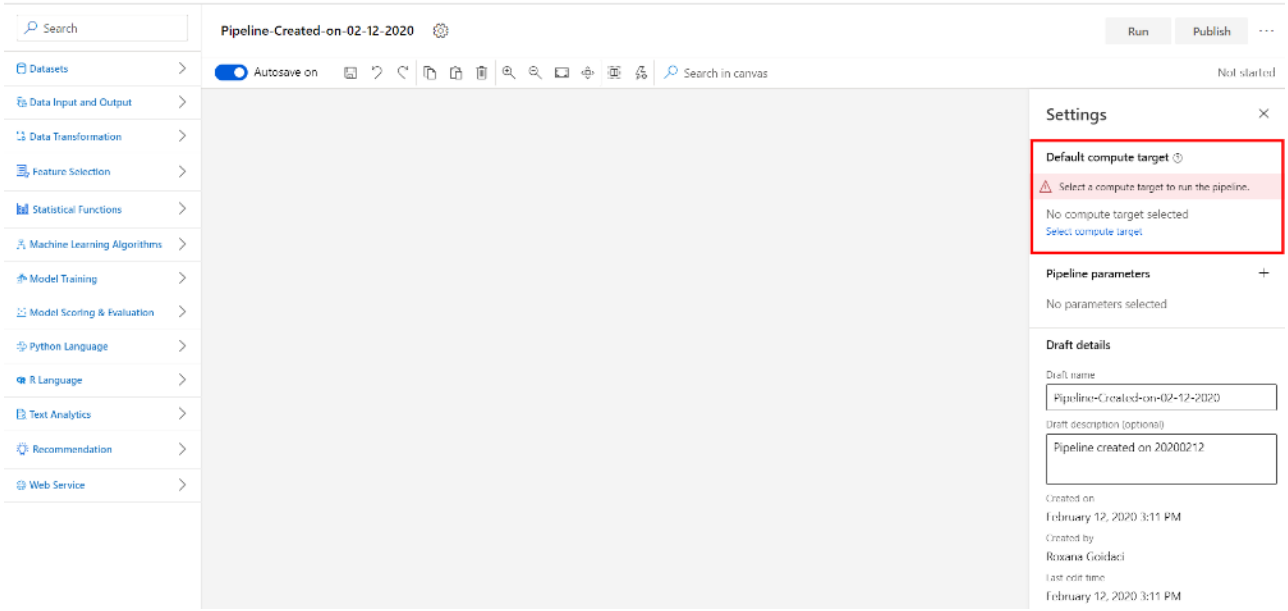
For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

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

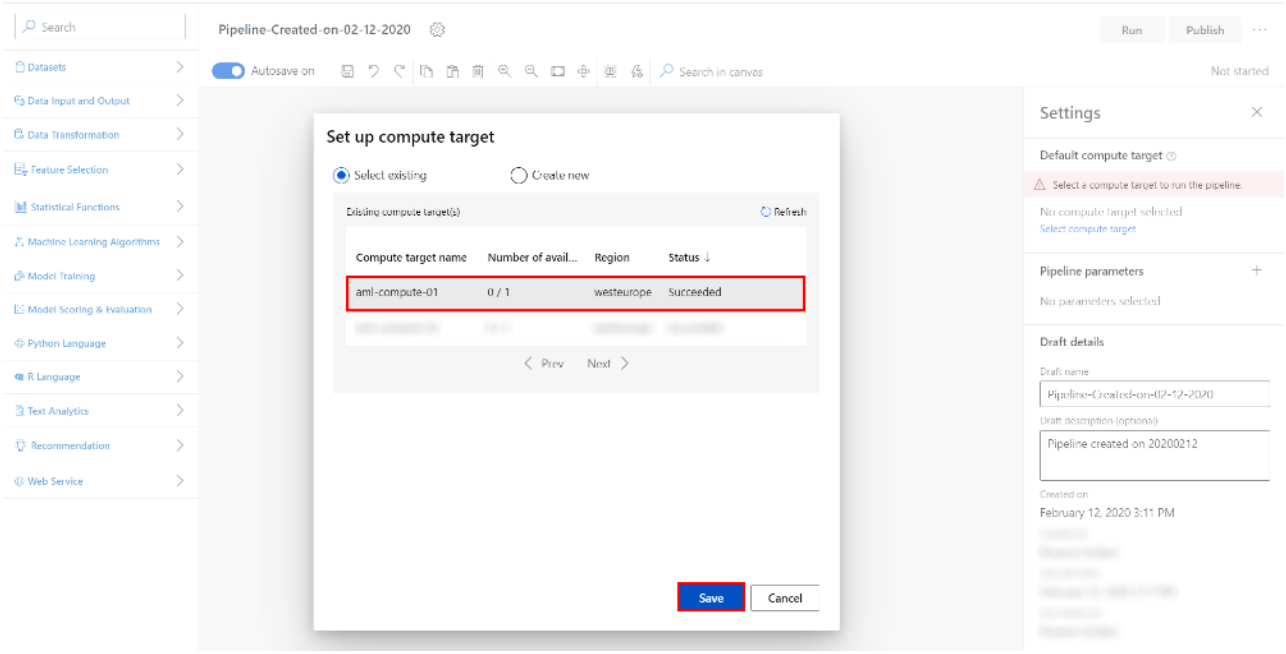


Task 2: Setup Compute Target

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



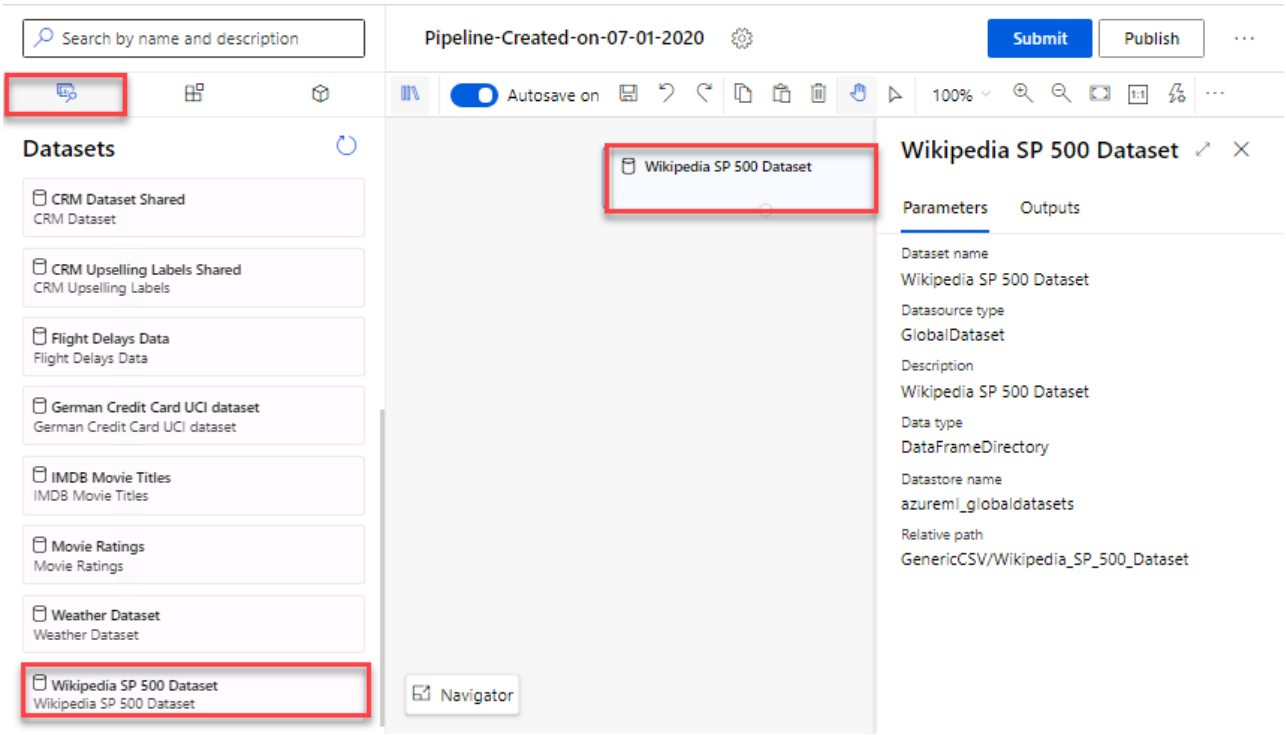
2. In the `Set up compute target` editor, select the available compute, and then select **Save**.



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.

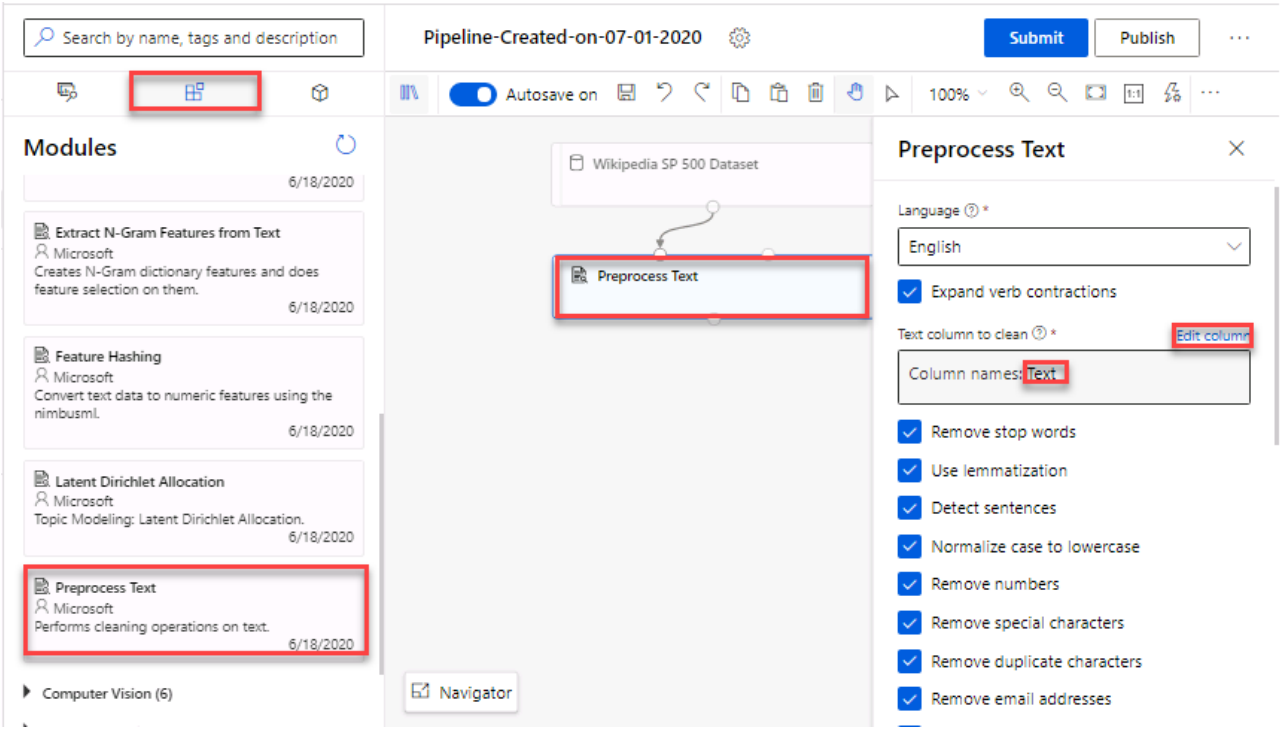
Task 3: Add Wikipedia SP 500 Sample Datasets

1. Select **Datasets** section in the left navigation. Next, select **Samples, Wikipedia SP 500 Dataset** and drag and drop the selected dataset on to the canvas.



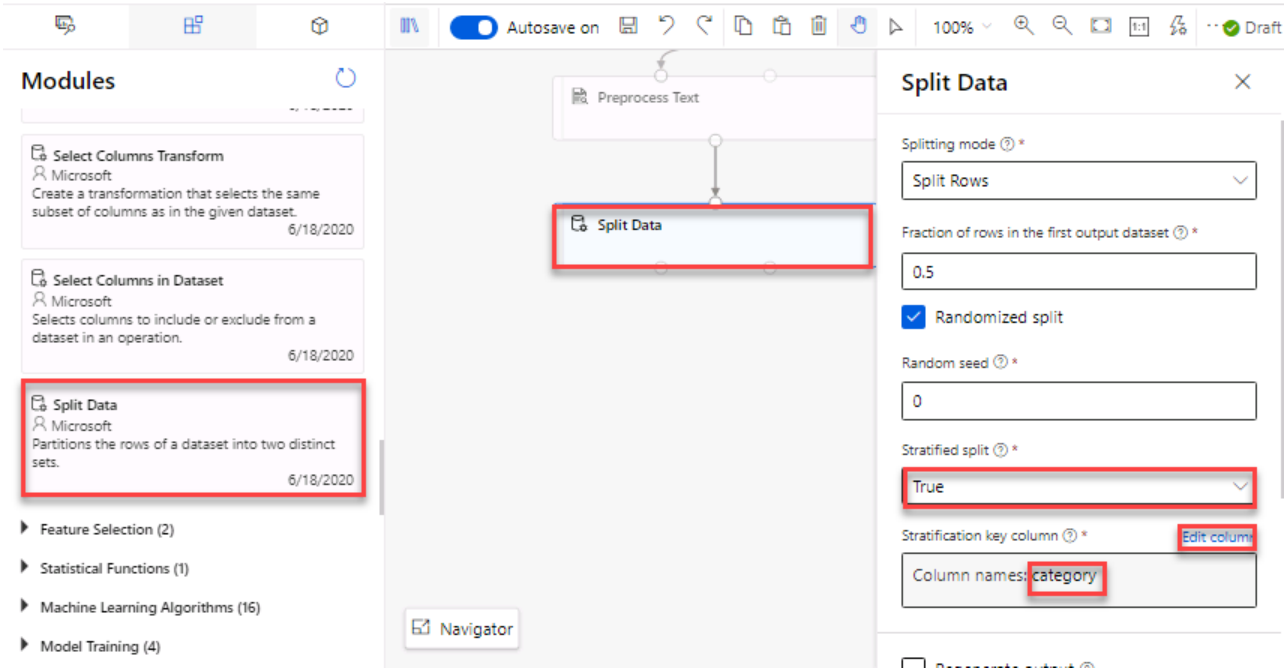
Task 4: Preprocess text for following steps

1. Select **Text Analytics** section in the Modules . Follow the steps outlined below:
    1. Select the **Preprocess Text** prebuilt module.
    2. Drag and drop the selected module on to the canvas.
    3. Connect the output of the `Wikipedia SP 500` module to the input of the `Preprocess Text` module. The dataset input of this prebuilt module needs to be connected to a dataset that has at least one column containing text.
    4. Select the language from the `Language` dropdown list: **English**.
    5. Select the **Edit column** link to open the `Text column to clean` editor. Select the **Text** column in the `Enter column name` field.
    6. Leave all the other options checked, as in the default configuration of the `Preprocess Text` module.



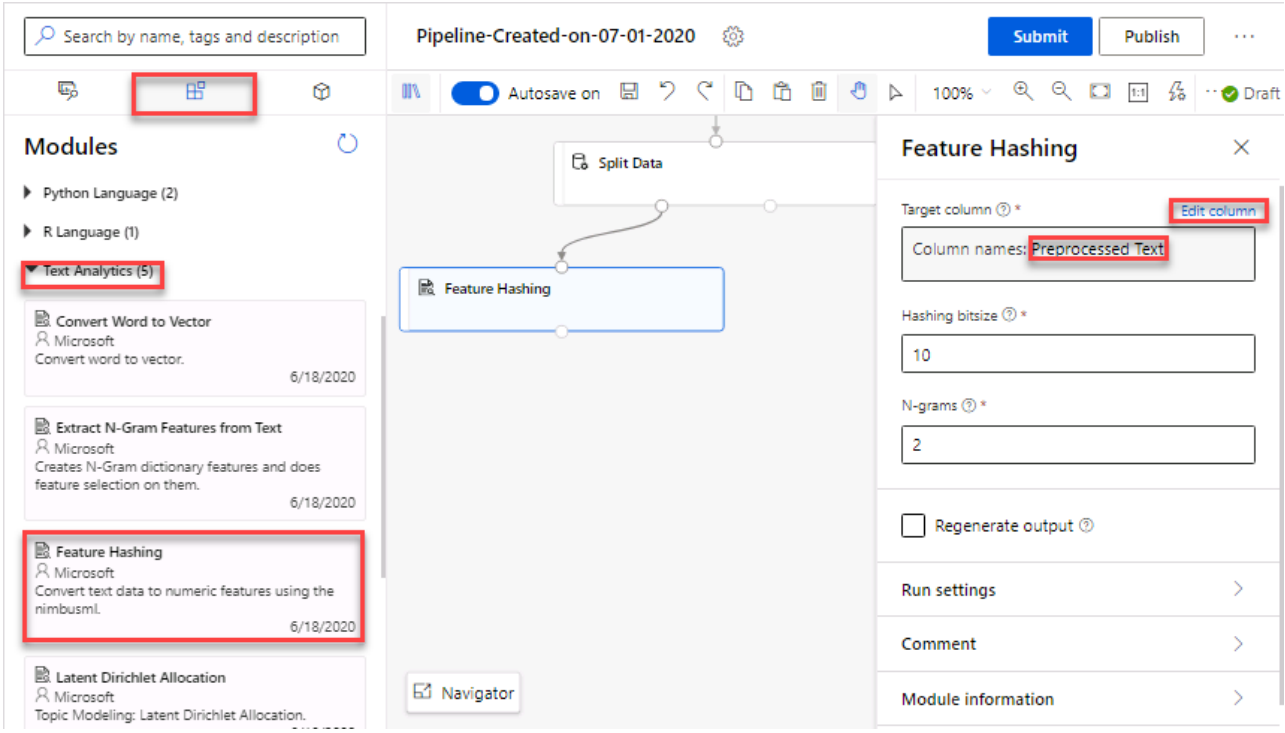
Task 5: Split the dataset into training set (0.5) and test set (0.5)

1. Select **Data Transformation** section in the Modules. Follow the steps outlined below:
    1. Select the **Split Data** prebuilt module.
    2. Drag and drop the selected module on to the canvas.
    3. Fraction of rows in the first output dataset: **0.5**.
    4. Stratified seed: **True**.
    5. Select the **Edit column** link to open the `Stratification key column` editor. Select the **Category** column in the `Enter column name` field.
    6. Connect the output of the `Preprocess Text` module to the input of the `Split Data` module.



Task 6: Convert the plain text of the articles to integers with Feature Hashing module, on the training set

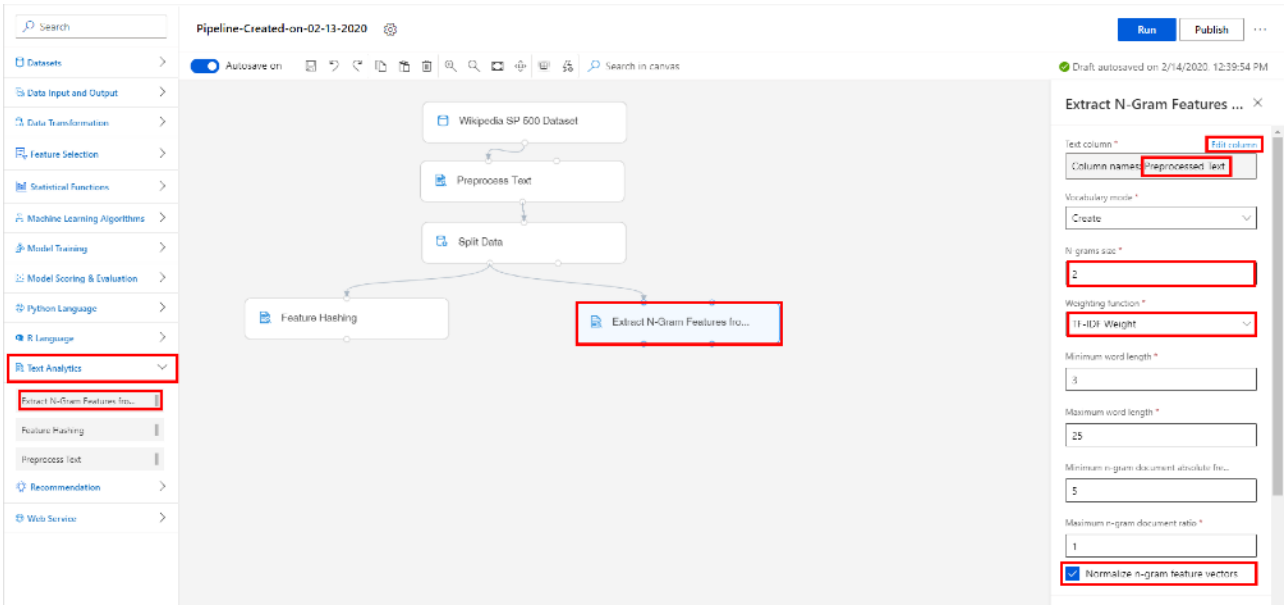
1. Select **Text Analytics** section in the left navigation. Follow the steps outlined below:
    1. Select the **Feature Hashing** prebuilt module.
    2. Drag and drop the selected module on to the canvas.
    3. Connect the first output of the `Split Data` to the input of the `Feature Hashing` module.
    4. Select the **Edit columns** link to open the `Target column` editor and then enter the **Preprocessed Text** column for the Target column field.
    5. Set `Hashing bitsize`: **10** and set the number of `N-grams`: **2**.



The goal of using feature hashing is to reduce dimensionality; also it makes the lookup of feature weights faster at classification time because it uses hash value comparison instead of string comparison.

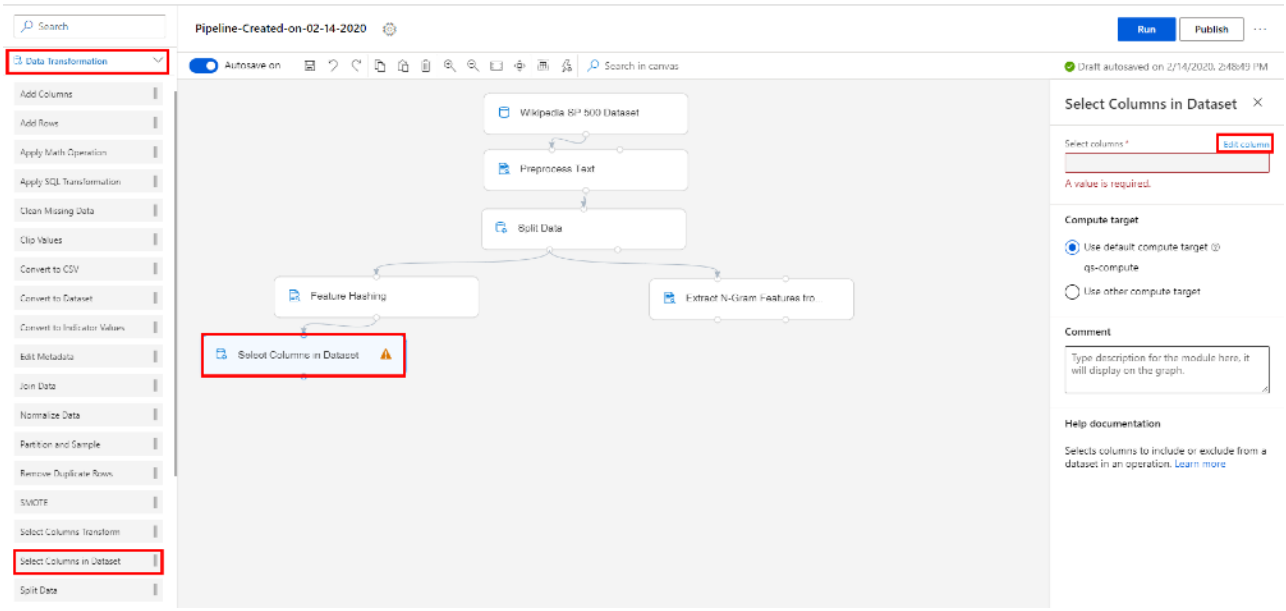
Task 7: Featurize unstructured text data with Extract N-Gram Feature from Text module, on the training set

1. Select **Text Analytics** section in the left navigation. Follow the steps outlined below:
    1. Select the **Extract N-Gram Feature from Text** prebuilt module.
    2. Drag and drop the selected module on to the canvas.
    3. Connect the first output of the `Split Data` to the input of the `Extract N-Gram Feature from Text` module.
    4. Select the **Edit columns** link to open the `Text column` editor and then enter the **Preprocessed Text** column for the Target column field.
    5. Leave default selection of `Vocabulary mode` to **Create** to indicate that you're creating a new list of n-gram features.
    6. Set `N-grams size`: **2** (which is the maximum size of the n-grams to extract and store)
    7. Select `Weighting function`: **TF-IDF Weight**. (This function calculates a term frequency/inverse document frequency score and assigns it to each n-gram. The value for each n-gram is its TF score multiplied by its IDF score.)
    8. Check the option to **Normalize n-gram feature vectors**. If this option is enabled, each n-gram feature vector is divided by its L2 norm.

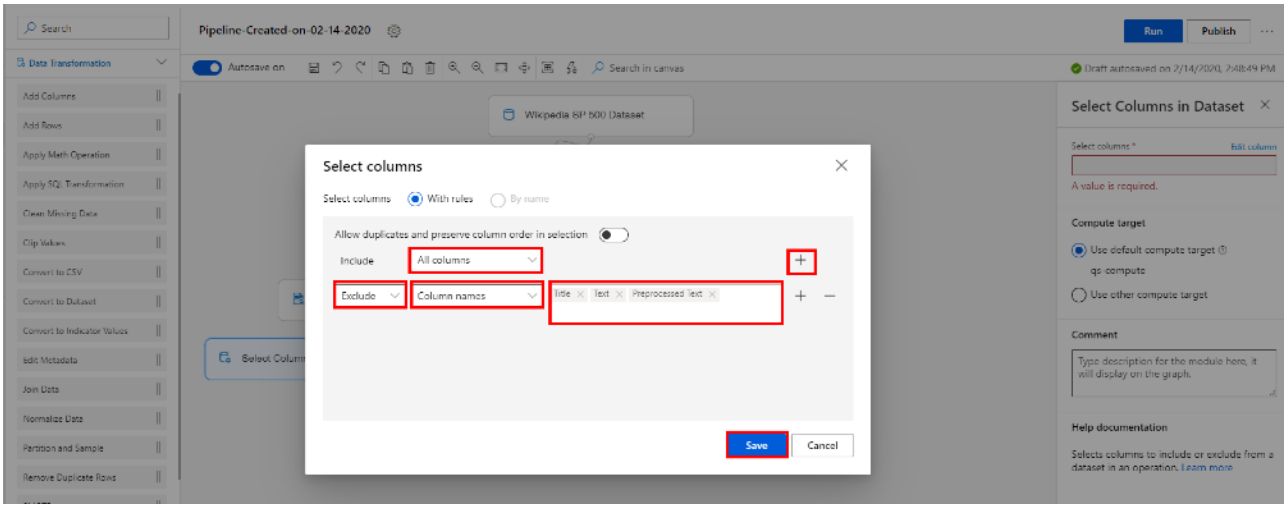


Task 8: Remove text columns from dataset

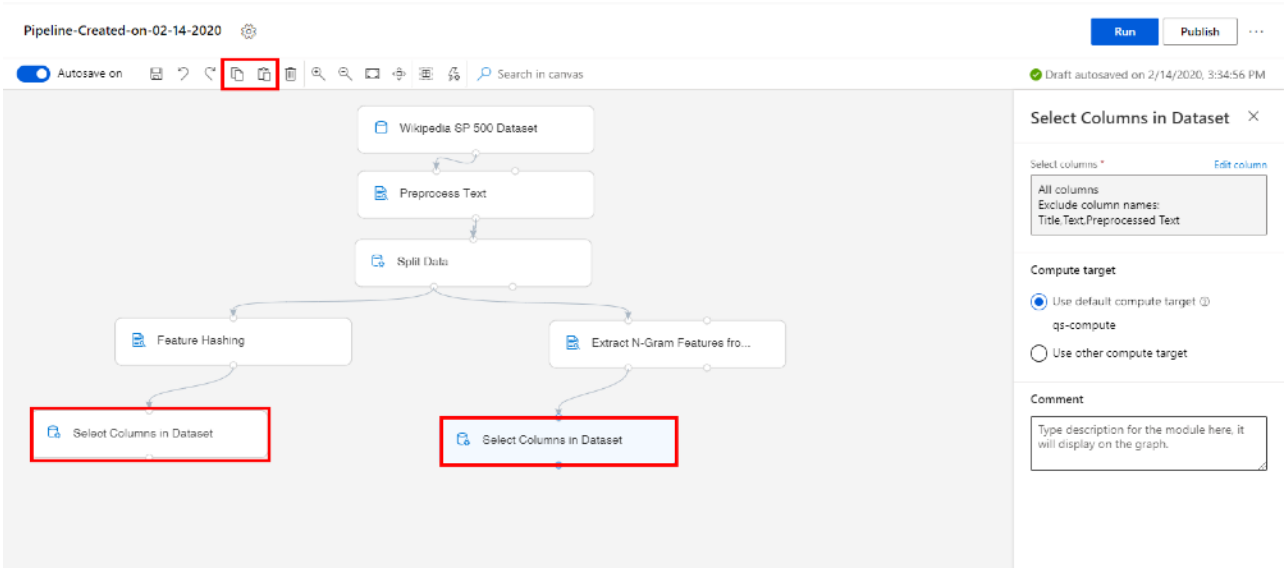
1. Select **Data Transformation** section in the left navigation. Follow the steps outlined below to add two Select Columns in Dataset modules on both featurization branches:
    1. Select the **Select Columns in Dataset** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the input `Select Columns in Dataset` module to the `Feature Hashing` module output.



2. Select the **Edit columns** link to open the `Select columns` editor and then select **All columns**.
3. Click on the **+** icon to add another operation line and select the **Exclude, Column names** option. Enter to following list of columns to be excluded: **Title**, **Text**, **Preprocessed Text**. Select **Save** to close the editor.

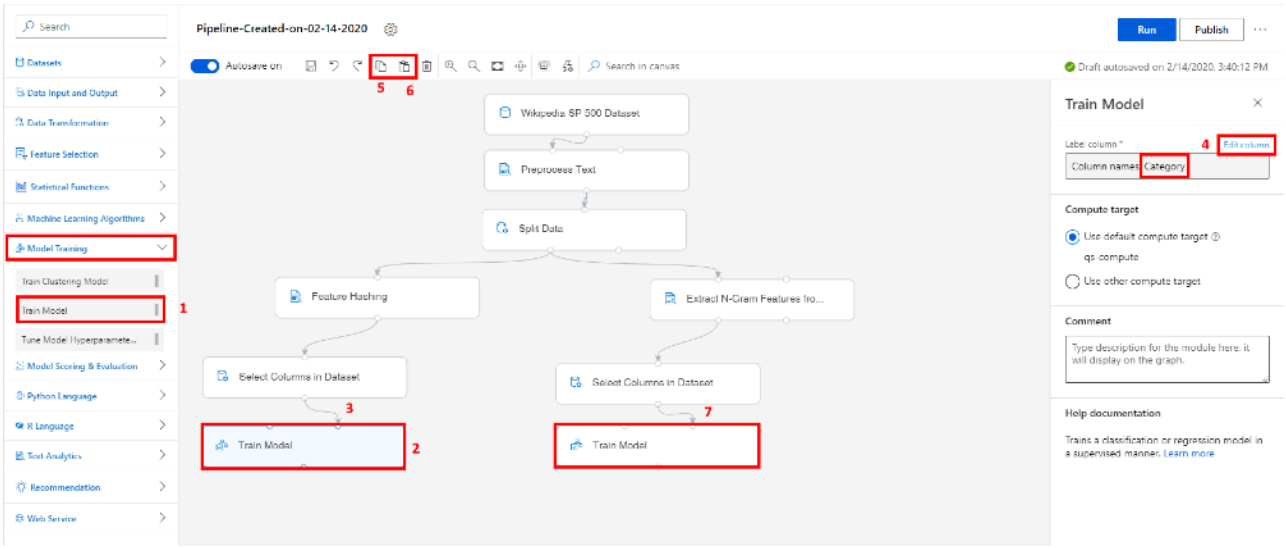


4. Select the `Select Columns in Dataset` module and click on the **Copy** icon and then **Paste** it on the canvas. Connect the copied module to the `Extract N-Gram Feature from Text` module.



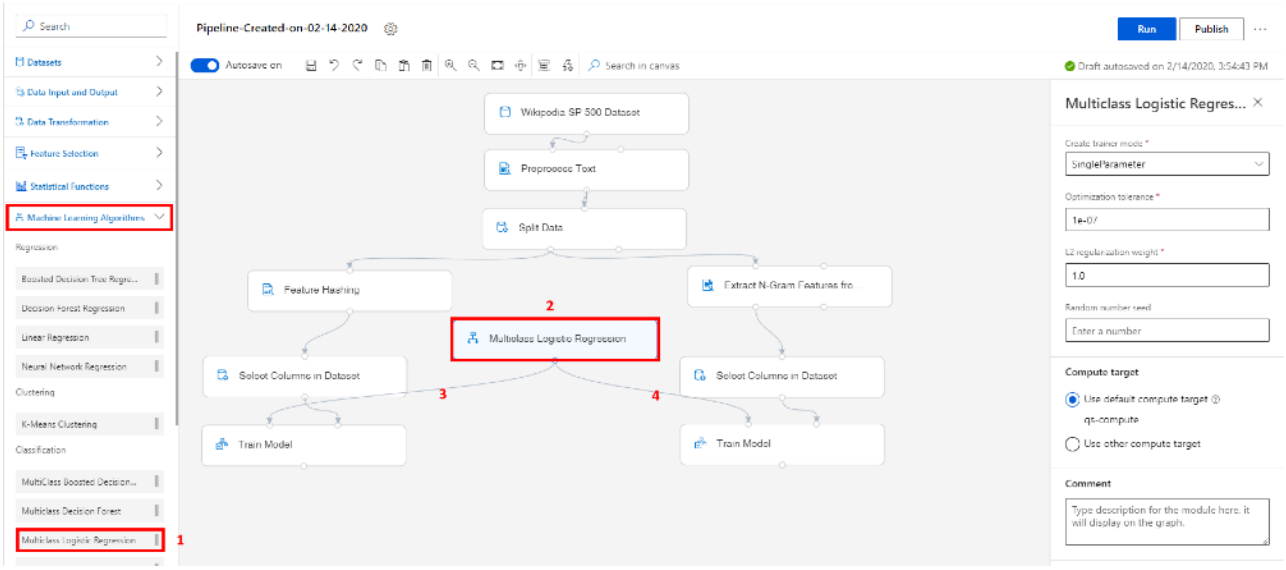
Task 9: Add the Train Model Modules

1. Select **Model Training** section in the left navigation. Follow the steps outlined below:
    1. Select the **Train Model** prebuilt module.
    2. Drag and drop the selected module on to the canvas
    3. Connect the first output of the `Select Columns in Dataset` module on the left of the canvas, to the second input (the Dataset input) of the newly added `Train Model` module.
    4. Select the **Edit columns** link to open the `Label column` editor and then enter the **Category** column. Select **Save** to close the editor.
    5. Select the `Train Model` module and click on the **Copy** icon.
    6. Click on the **Paste** icon to paste a second `Train Model` module on the canvas under the right branch of the pipeline tree.
    7. Connect this last `Train Model` module, by linking at the second input (the Dataset input), the output of the right most `Select Columns in Dataset` module on the canvas.



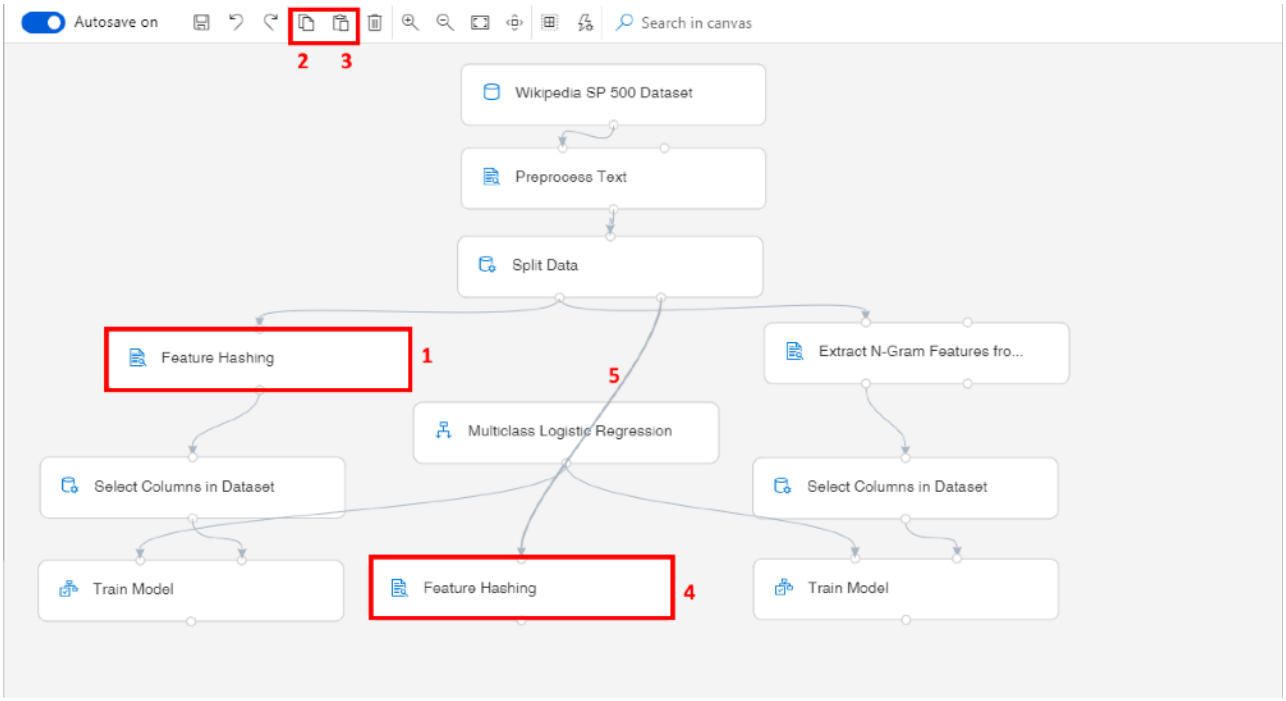
Task 10: Initialize Multiclass Logistic Regression Model

1. Select **Machine Learning Algorithms** section in the left navigation. Follow the steps outlined below:
    1. Select the **Multiclass Logistic Regression** prebuilt module, in the `Classification` category.
    2. Drag and drop the selected module on to the canvas
    3. Connect the output of the `Multiclass Logistic Regression` module to the first input of the left branch (the Feature Hashing approach) `Train Model` module.
    4. Connect the output of the `Multiclass Logistic Regression` module to the first input of the right branch (the N-Gram Features approach) `Train Model` module.



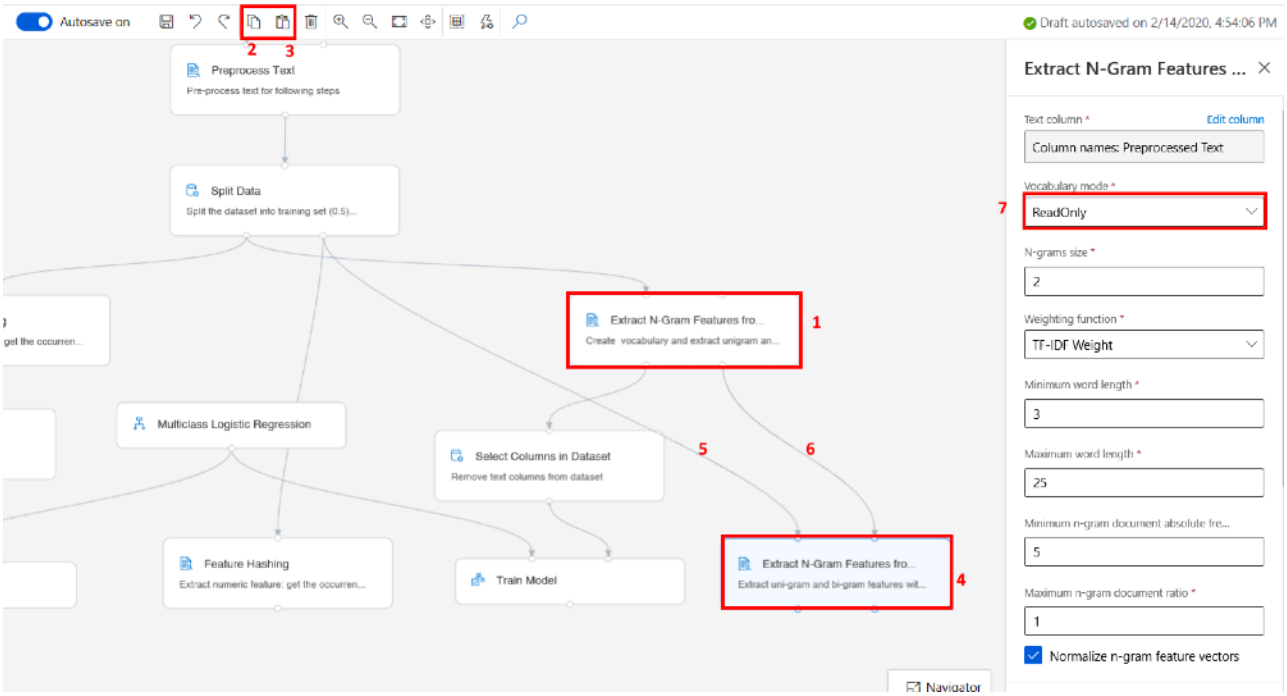
Task 11: Convert the plain text of the articles to integers with Feature Hashing module, on the test set

1. Select the existing **Feature Hashing** module.
2. Copy the **Feature Hashing** module using the **Copy** option from the top menu.
3. **Paste** the copied module on the canvas.
4. Position the new **Feature Hashing** module between the two `Train Model` modules on the canvas.
5. Connect the second output of the `Split Data`, the test set output, to the input of the newly added `Feature Hashing` module.



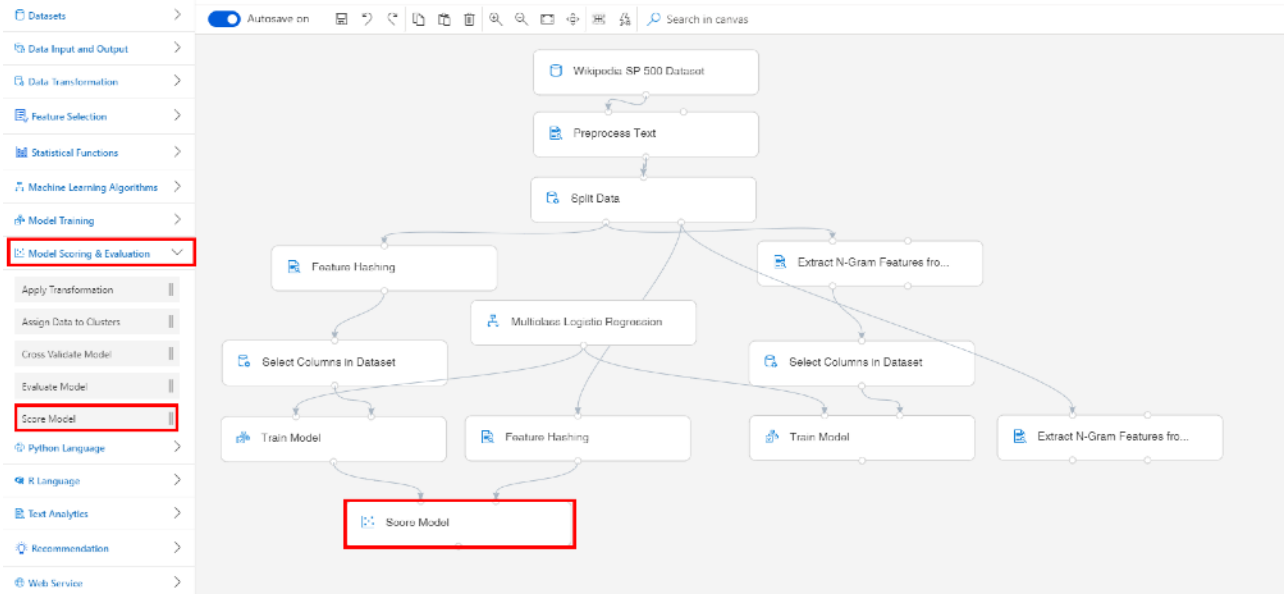
Task 12: Featurize unstructured text data with Extract N-Gram Feature from Text module, on the test set

1. Select the existing **Extract N-Gram Feature from Text** module.
2. Copy the **Extract N-Gram Feature from Text** module using the **Copy** option from the top menu.
3. **Paste** the copied module on the canvas.
4. Position the new **Extract N-Gram Feature from Text** module near the rightmost `Train Model` module on the canvas.
5. Connect the second output of the `Split Data`, the test set output, to the first input of the newly added `Extract N-Gram Feature from Text` module.
6. Connect the second output of the uppermost `Extract N-Gram Feature from Text`, to the second input of the copied `Extract N-Gram Feature from Text` module.
7. Select the newly added module and in the right settings pane, set the `Vocabulary mode` to **ReadOnly**

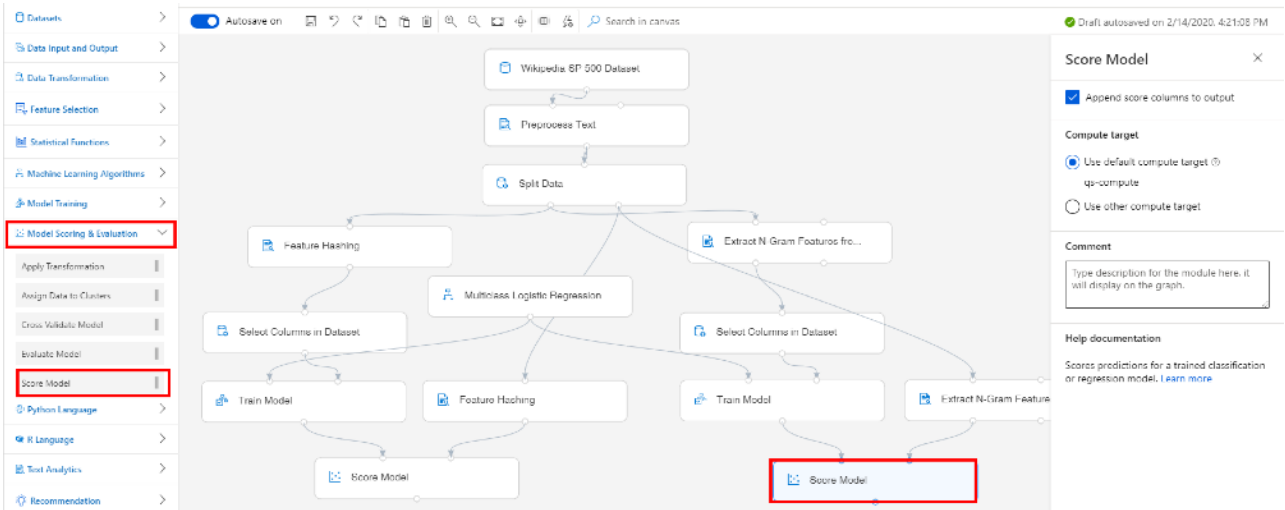


Task 13: Setup Score Model Modules

1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Score Model** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the leftmost `Train Model` module to the first input of the `Score Model` module
    4. Connect the output of the `Feature Hashing` module (the lower one, on the test set branch) to the second input of the `Score Model` module

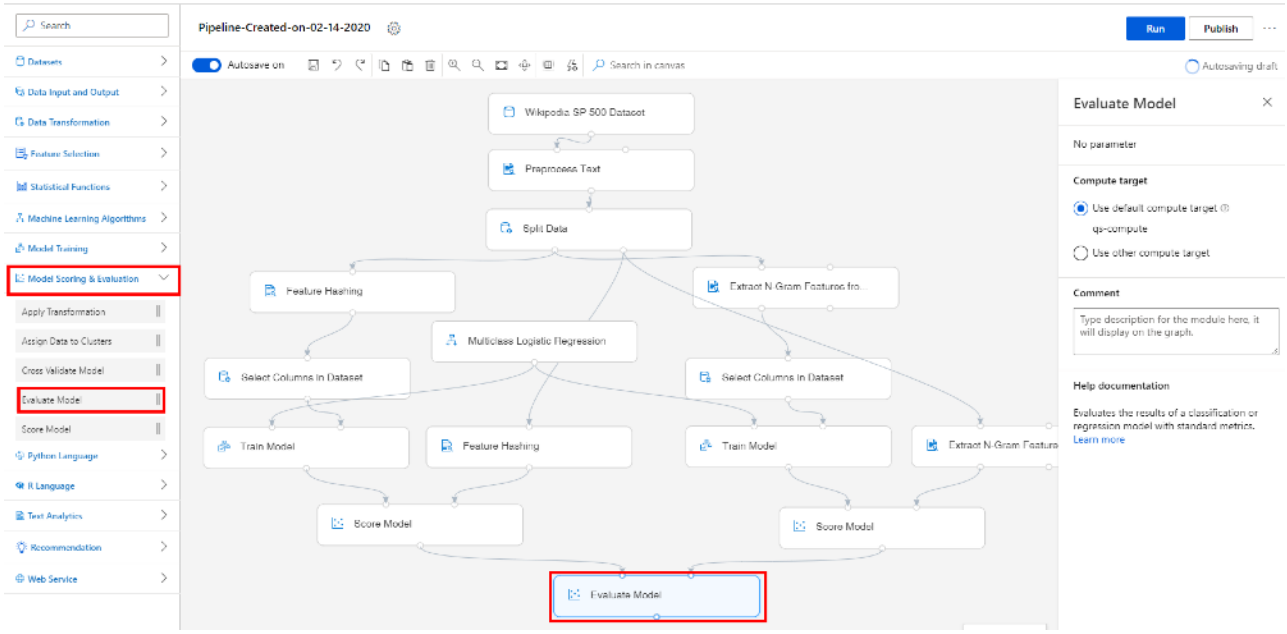


2. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Score Model** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the rightmost `Train Model` module to the first input of the `Score Model` module
    4. Connect the first output of the `Extract N-Gram Feature from Text` module (the lower one, on the test set branch) to the second input of the



Task 14: Setup Evaluate Model Module

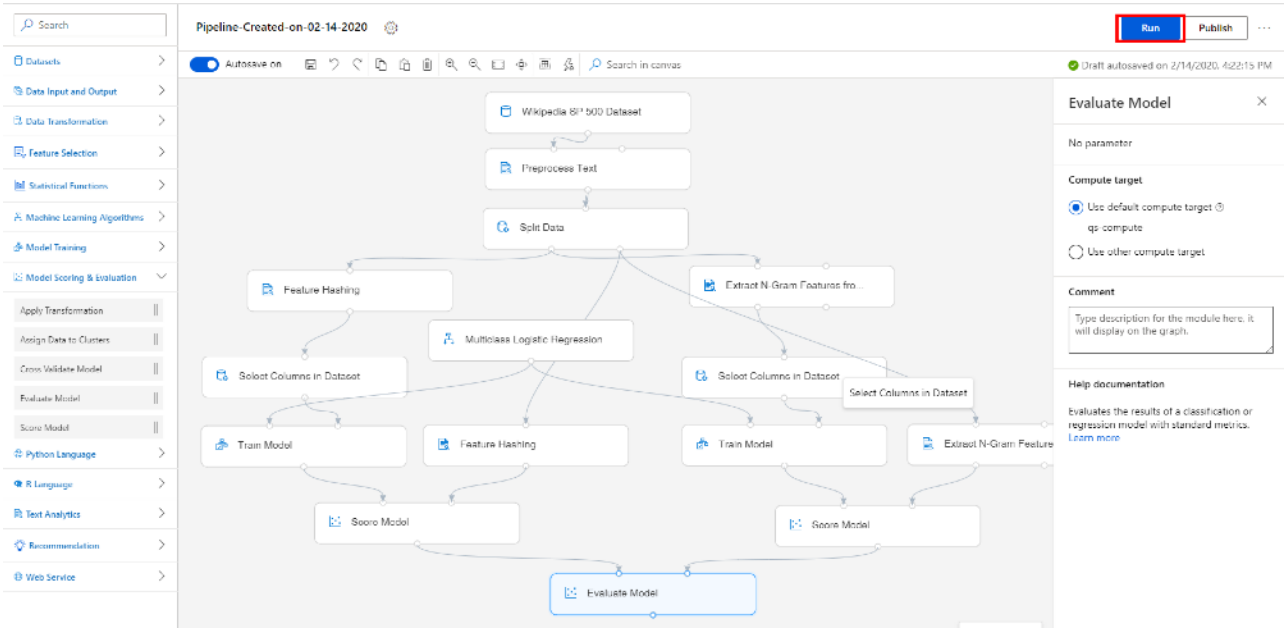
1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:
    1. Select the **Evaluate Model** prebuilt module
    2. Drag and drop the selected module on to the canvas
    3. Connect the two `Score Model` modules to the inputs of the `Evaluate Model` module



Exercise 2: Submit Training Pipeline

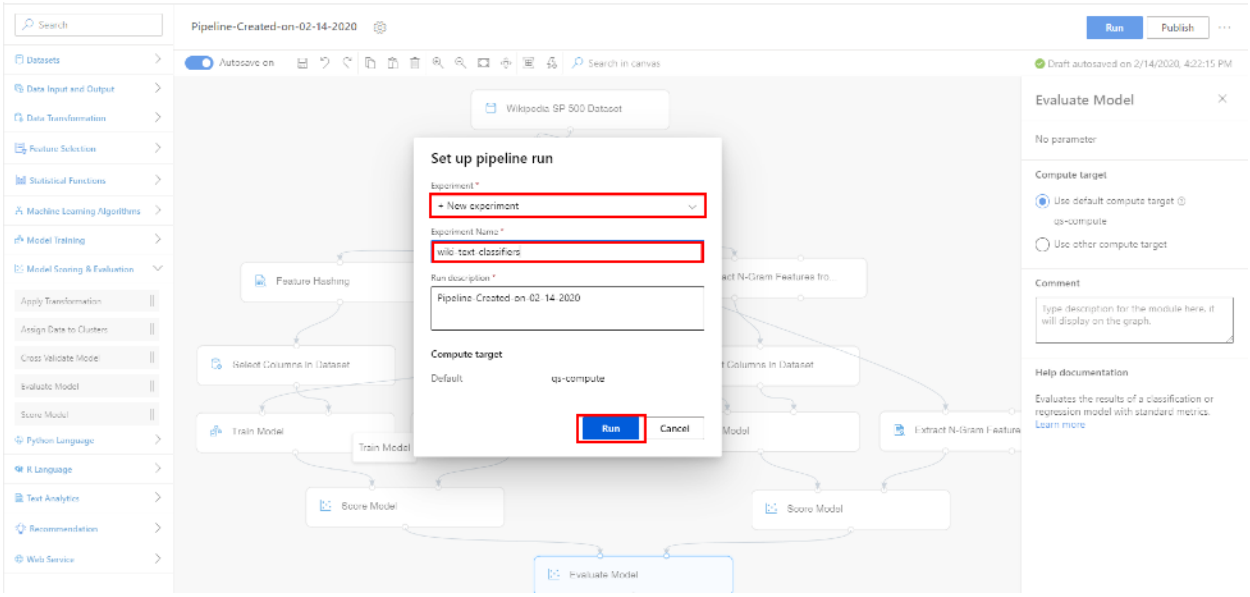
Task 1: Create Experiment and Submit Pipeline

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:` **wiki-text-classifier**, and then select **Submit**.

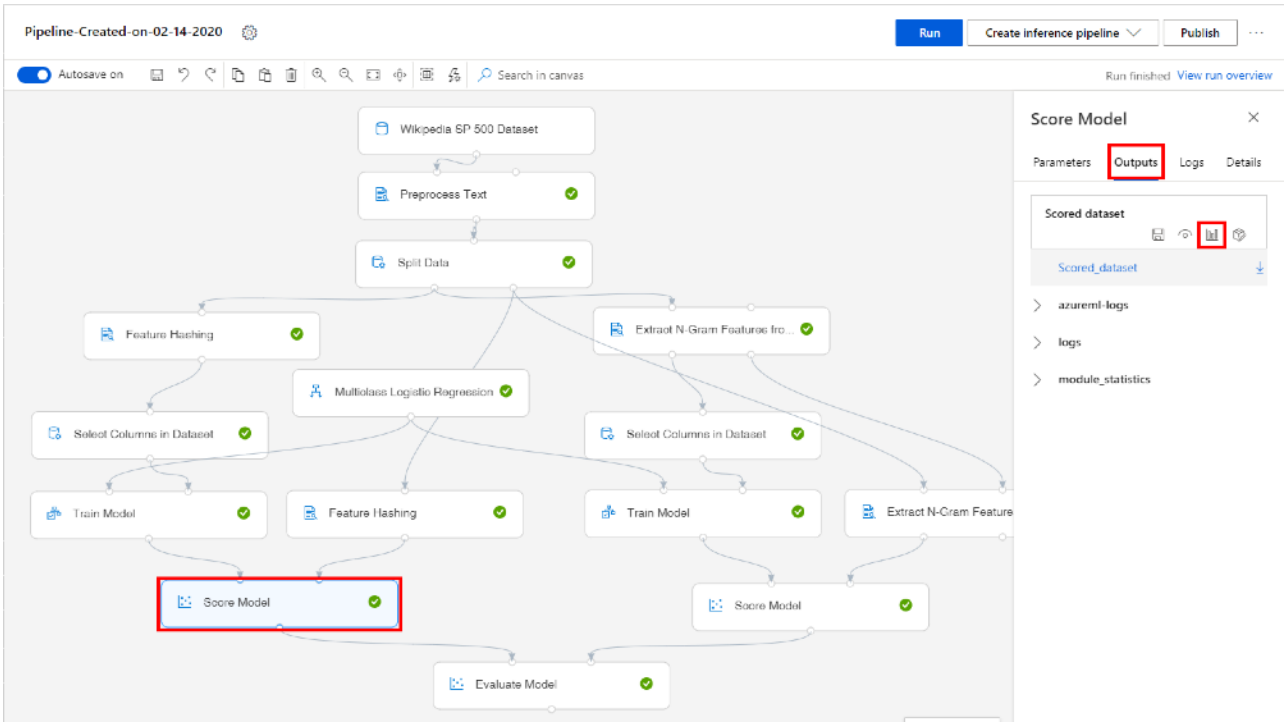


3. Wait for pipeline run to complete. It will take around **20 minutes** to complete the run.
4. While you wait for the model training to complete, you can learn more about the classification algorithm used in this lab by selecting Multiclass Logistic Regression module.

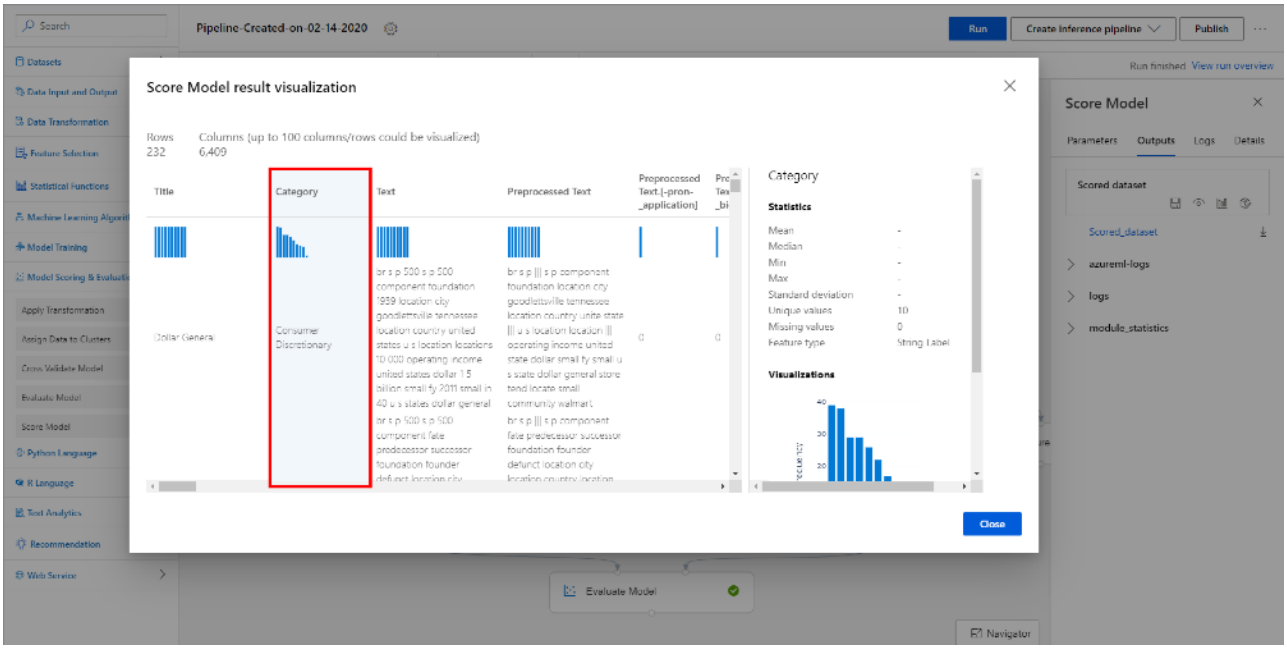
Exercise 3: Visualize Training Results

Task 1: Visualize the Model Predictions

1. Select **Score Model, Outputs, Visualize** to open the `Score Model result visualization` dialog or just simply right-click the `Score Model` module and select **Visualize Scored Dataset**.

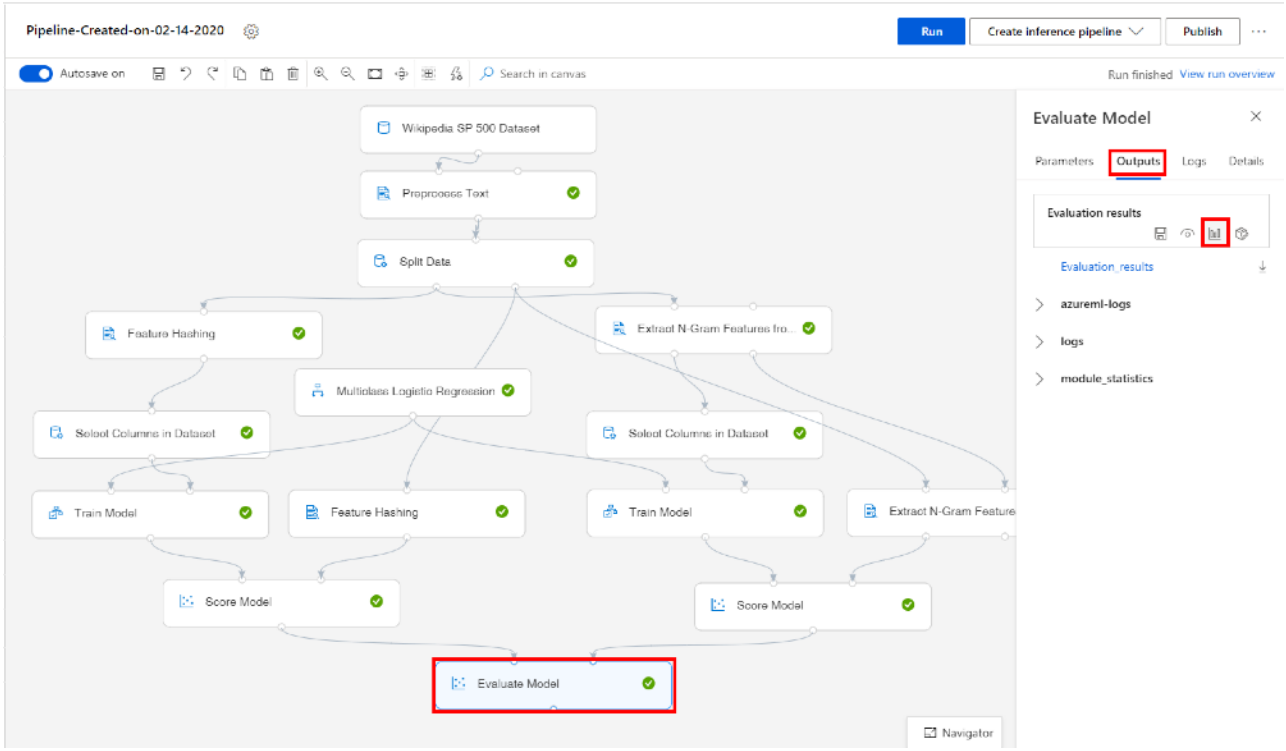


2. Observe the predicted values under the column **Category**.

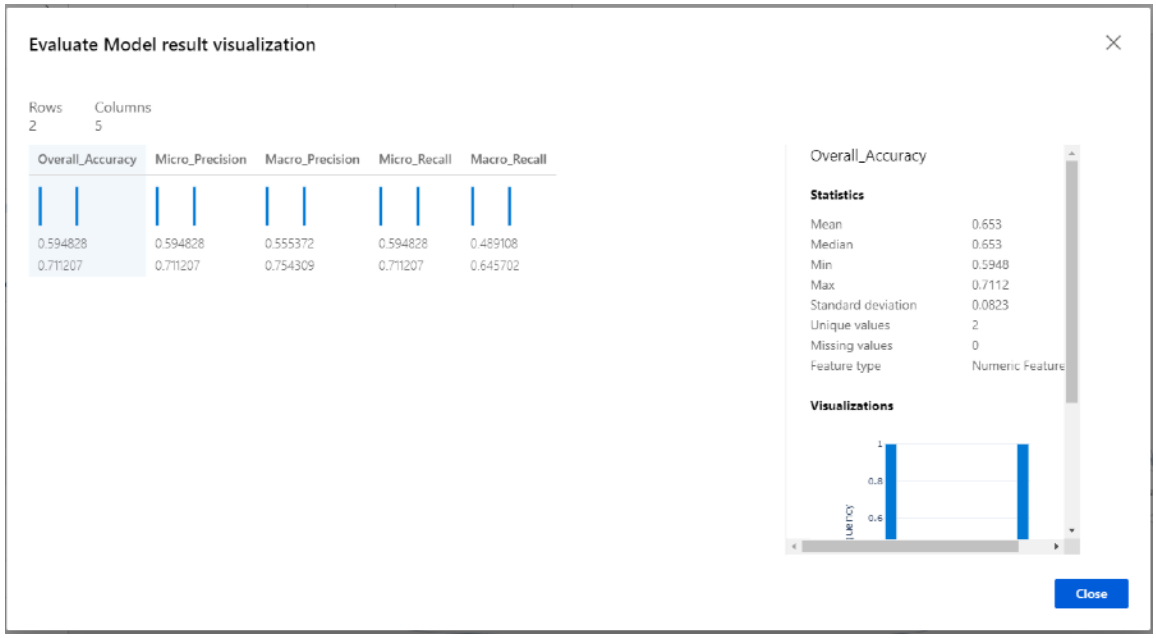


Task 2: Visualize the Evaluation Results

1. Select **Evaluate Model, Outputs, Visualize** to open the `Evaluate Model result visualization` dialog or just simply right-click the `Evaluate Model` module and select **Visualize Evaluation Results**.



2. Evaluate the model performance by reviewing the various evaluation metrics. Evaluate Model has two input ports, so that we could evaluate and compare scored datasets that are generated with different methods. In this sample, we compare the performance of the result generated with feature hashing method and n-gram method.



Next Steps

Congratulations! You have trained a simple text classifier and compared performance of the result generated with two different featurization modules. 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.

Lab 17 - Train a time-series forecasting model using Automated Machine Learning

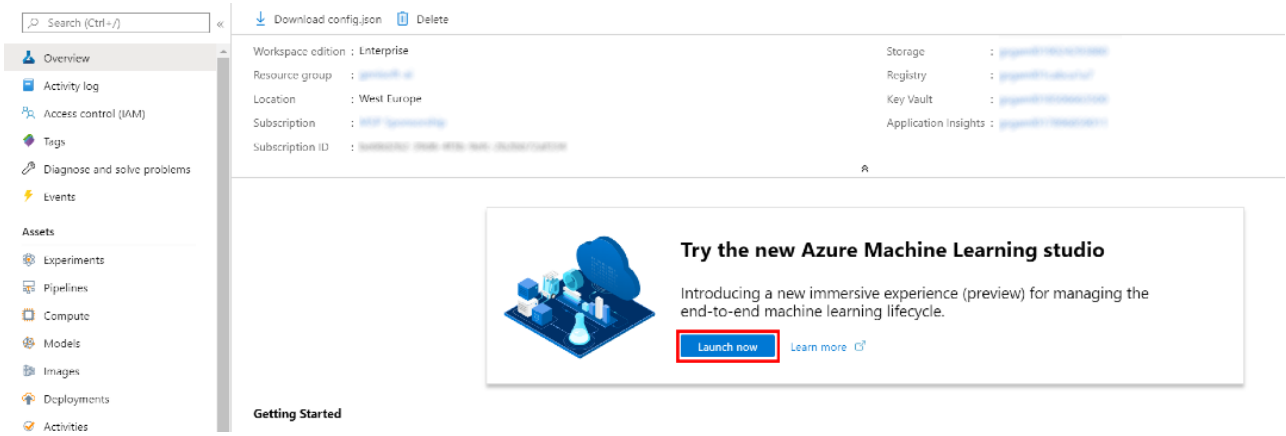
Overview

In this lab you will learn how the Automated Machine Learning capability in Azure Machine Learning (AML) can be used for the life cycle management of the manufactured vehicles and how AML helps in creation of better vehicle maintenance plans. To accomplish this, you will train a Linear Regression model to predict the number of days until battery failure using Automated Machine Learning available in AML studio.

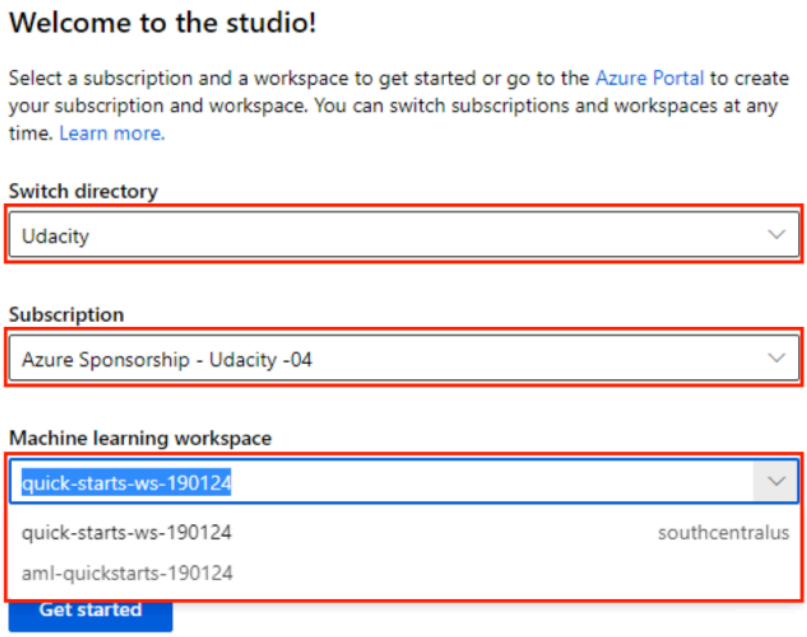
Exercise 1: Creating a model using automated machine learning

Task 1: Create an automated machine learning experiment using the Portal

1. In Azure portal, open the available machine learning workspace.
2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

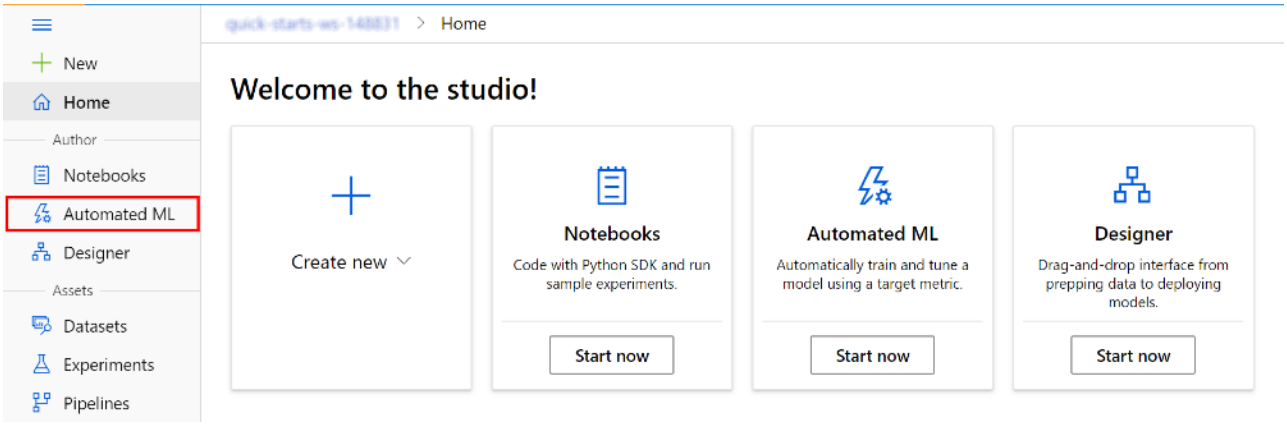


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:

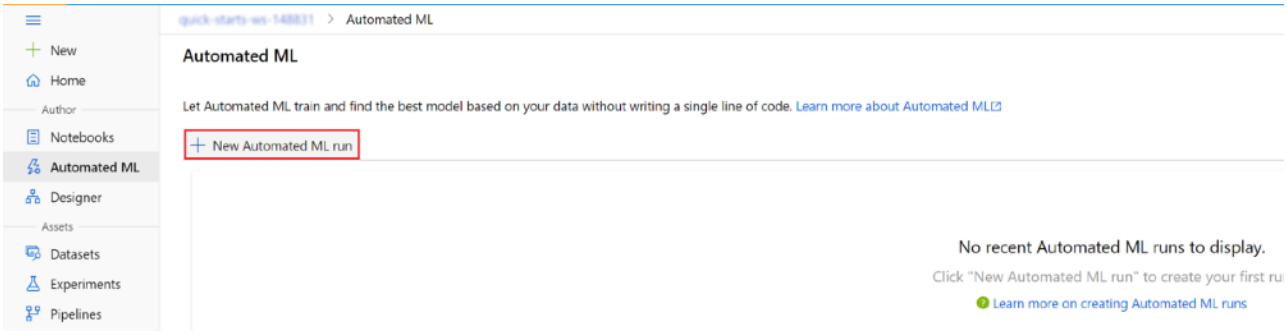


For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

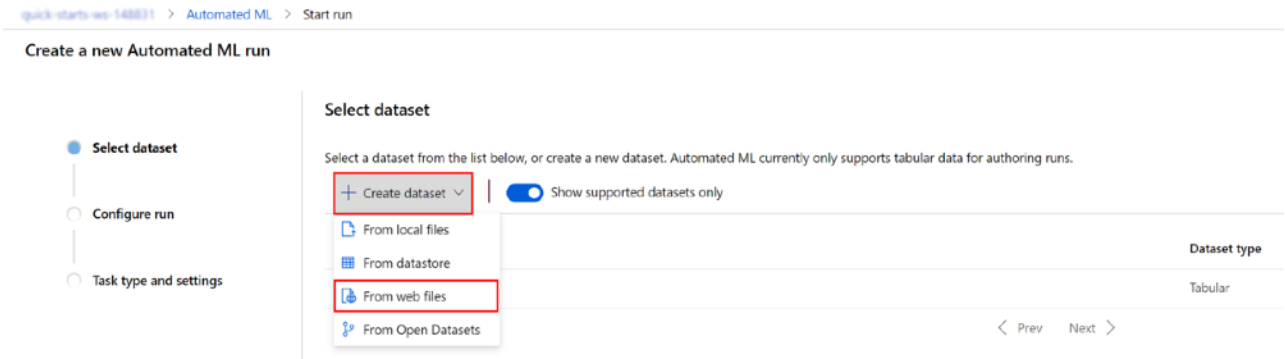
4. Select **Automated ML** in the left navigation bar.



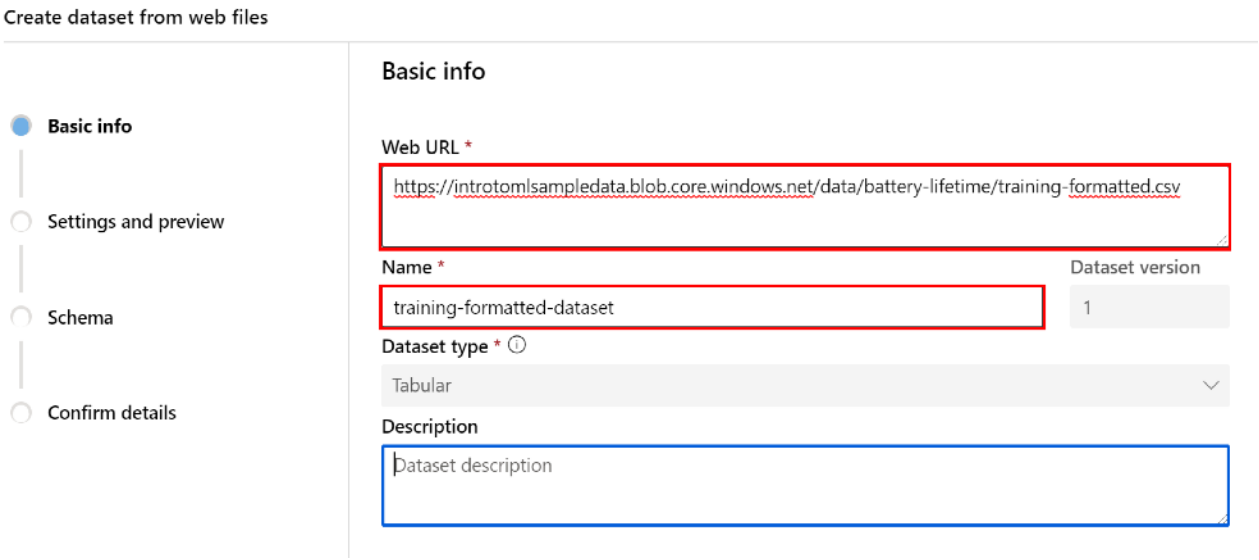
5. Select **New automated ML run** to start creating a new experiment.



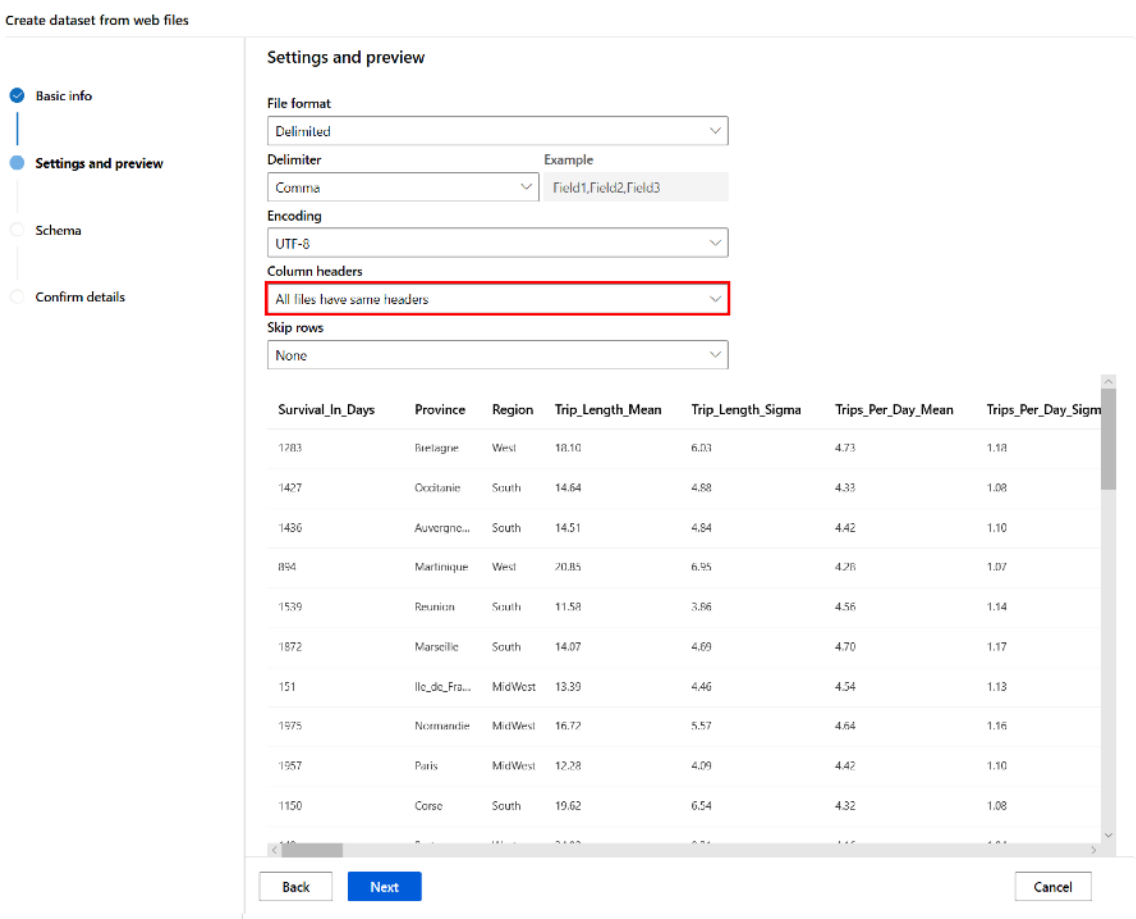
6. Select **Create dataset** and choose the **From web files** option from the drop-down.



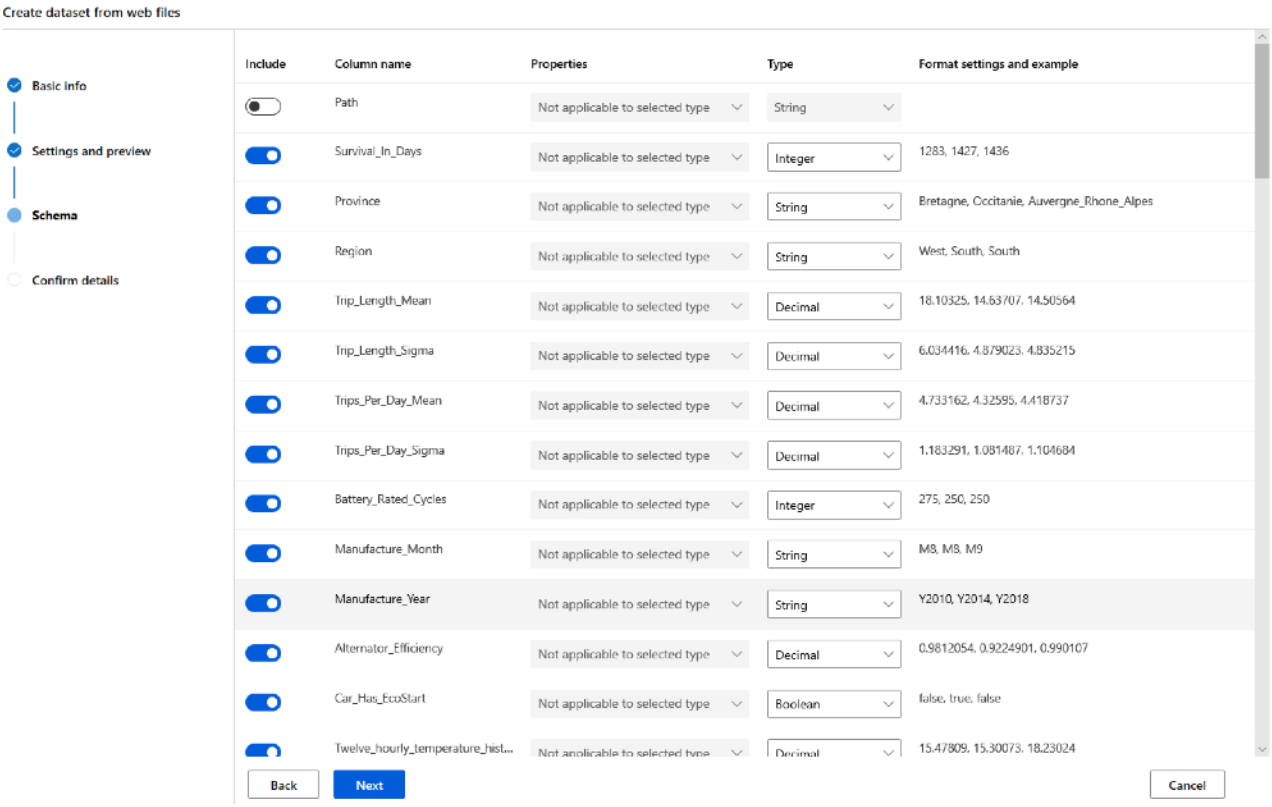
7. Fill in the training data URL in the `Web URL` field: `https://introtomlsampledata.blob.core.windows.net/data/battery-lifetime/training-formatted.csv`, make sure the name is set to `training-formatted-dataset`, and select **Next** to load a preview of the parsed training data.



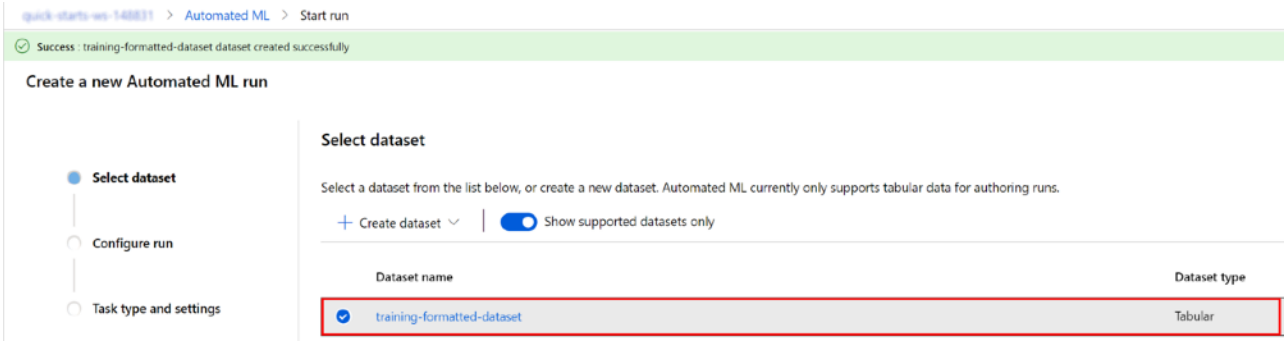
8. In the `Settings and preview` page, for the `Column headers` field, select `All files have same headers`. Scroll to the right to observe all of the columns in the data.



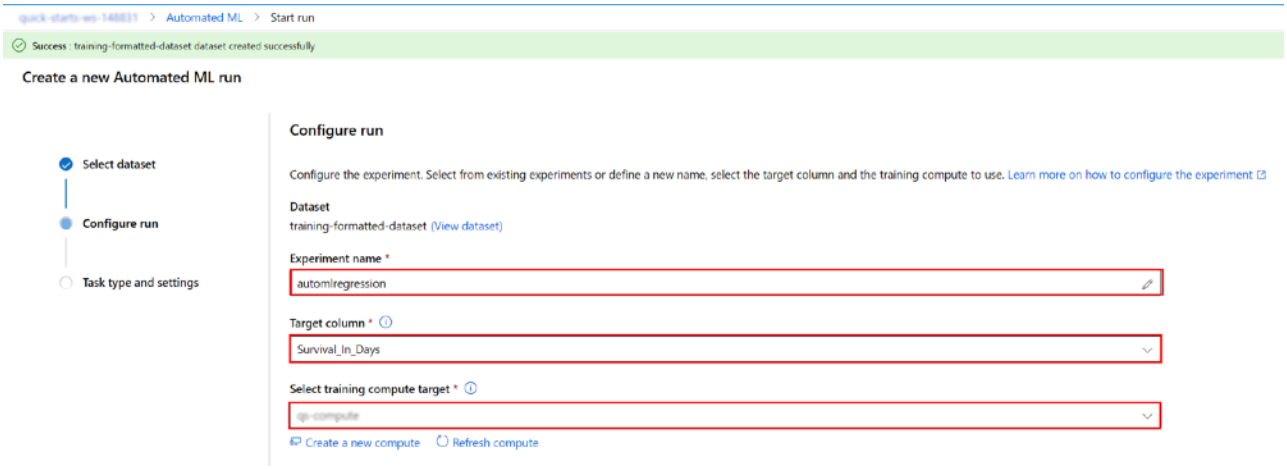
9. Select **Next** to check the schema and then confirm the dataset details by selecting **Next** and then **Create** on the confirmation page.



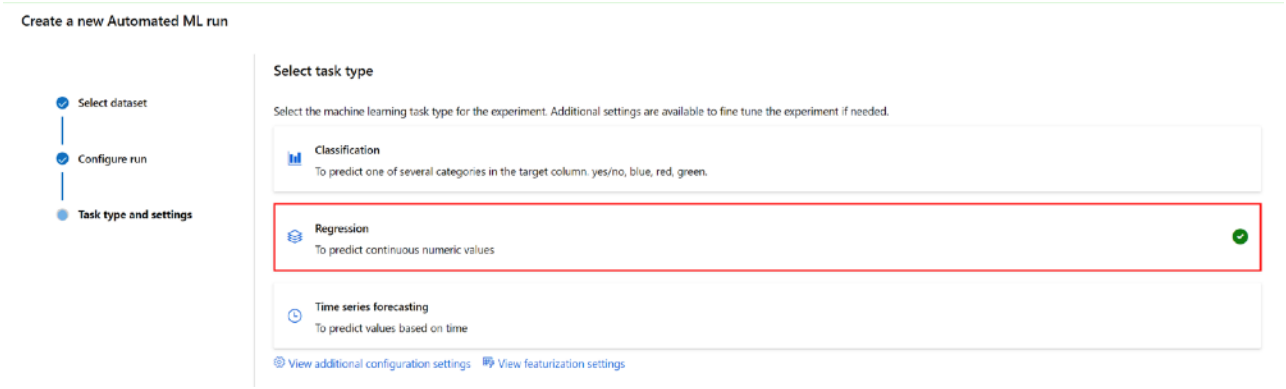
10. Now you should be able to select the newly created dataset for your experiment. Select the `training-formatted-dataset` dataset and select **Next** to move to the experiment run details page.



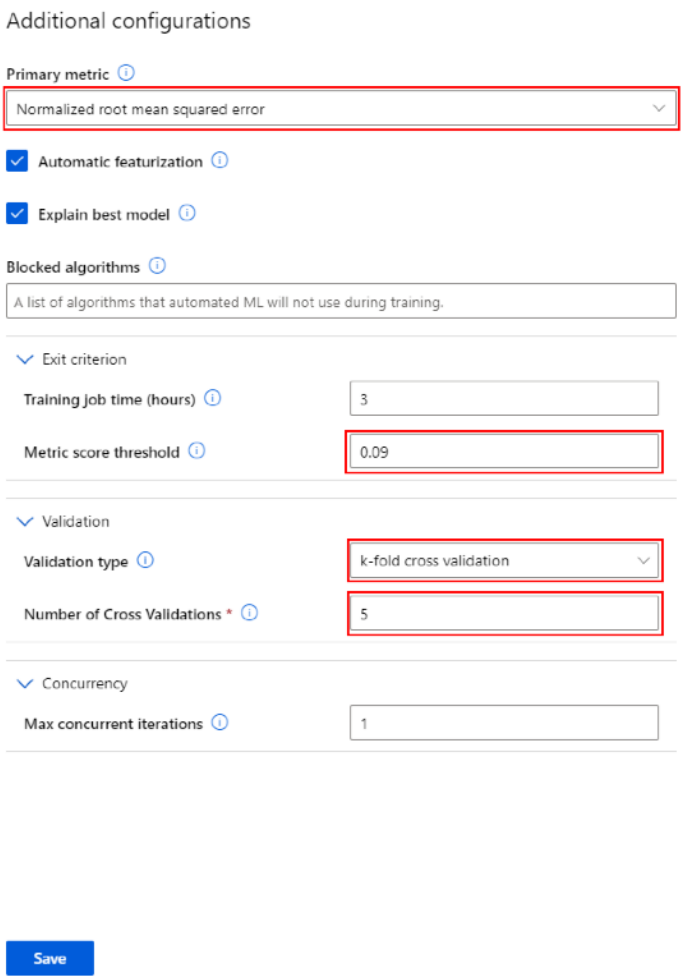
11. You will now configure the Auto ML run basic settings by providing the following values for the experiment name, target column and training compute:
    - Experiment name: **automlregression**
    - Target column: select **Survival_In_Days**
    - Select training compute target: : select **qs-compute**



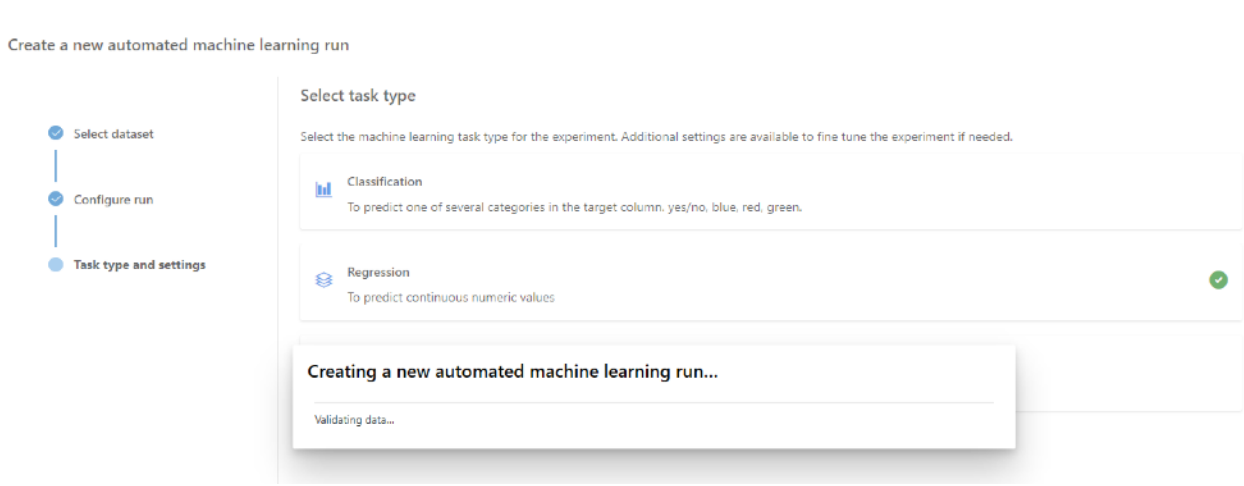
12. Select **Next** and select **Regression** in the `Task type and settings` page.



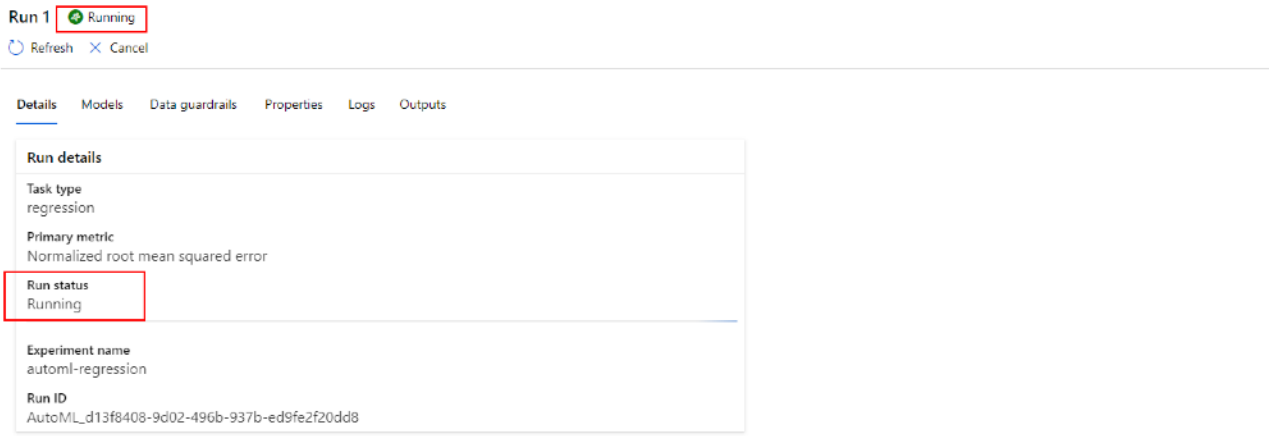
13. Select **View additional configuration settings** to open the advanced settings section. Provide the following settings:
    - Primary metric: **Normalized root mean squared error**
    - Exit criterion > Metric score threshold: **0.09**
    - Validation > Validation type: **k-fold cross validation**
    - Validation > Number of Cross Validations: **5**
    - Concurrency > Max concurrent iterations: **1**



14. Select **Save** and then **Finish** to begin the automated machine learning process.

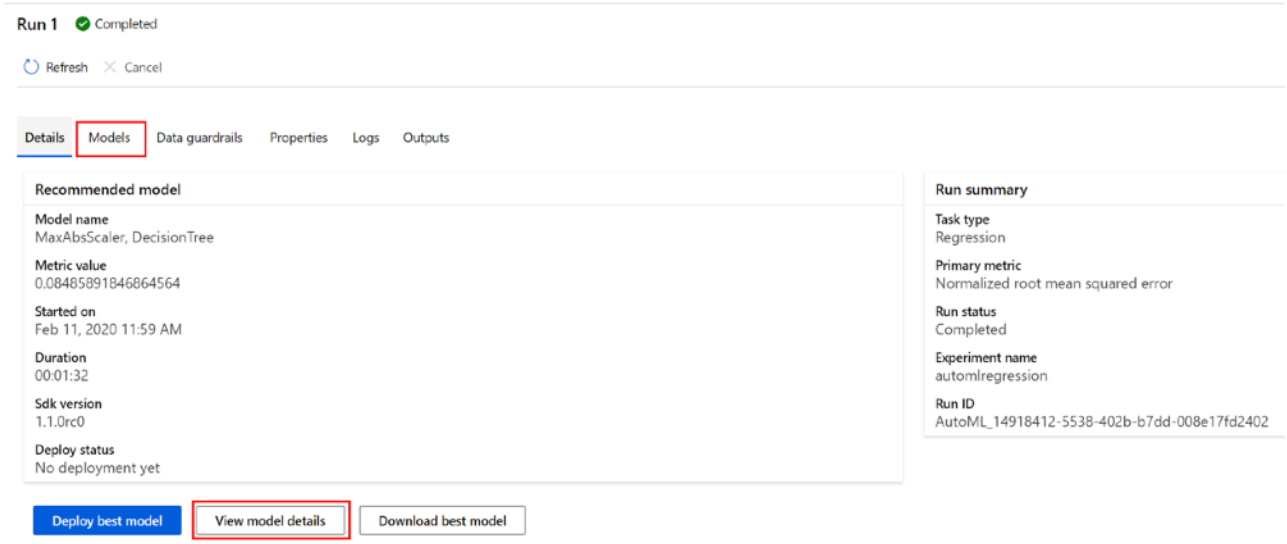


15. Wait until the `Run status` becomes **Running** in the `Run Detail page`.

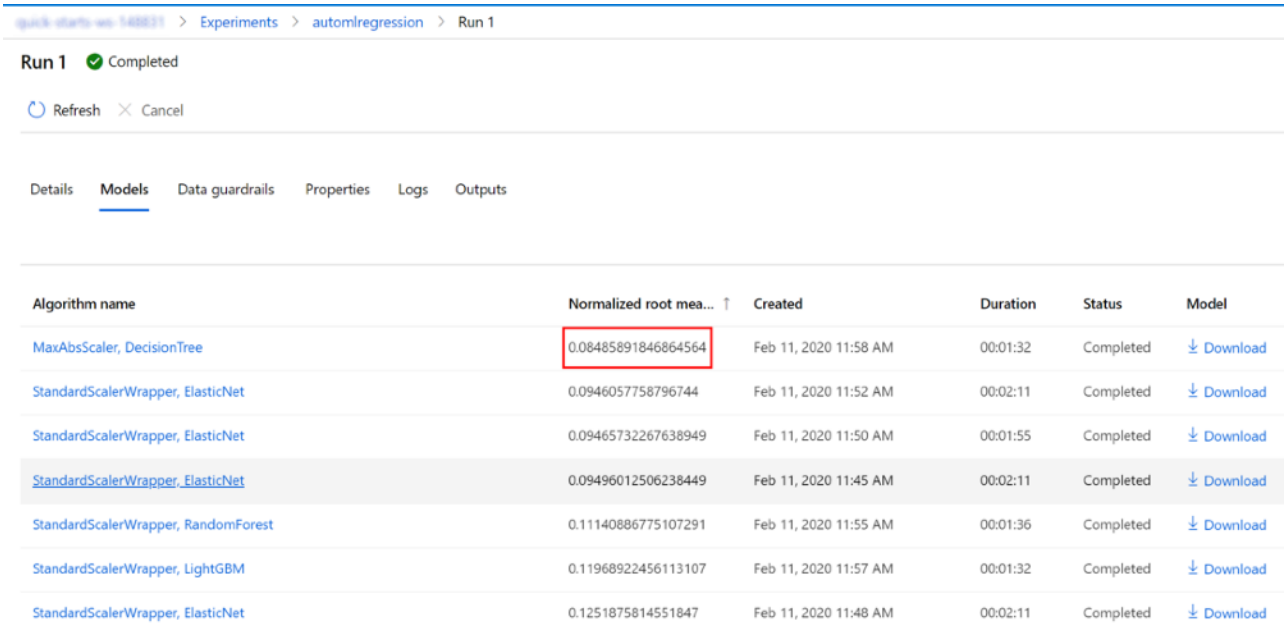


Task 2: Review the experiment run results

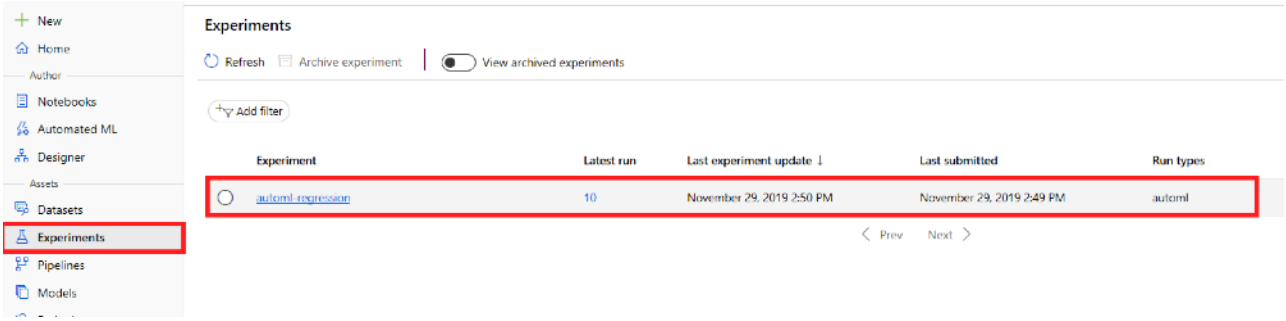
1. The experiment will run for about 15 minutes. While it runs and once it completes, you should check the `Models` tab on the `Run Detail` page to observe the model performance for the primary metric for different runs.



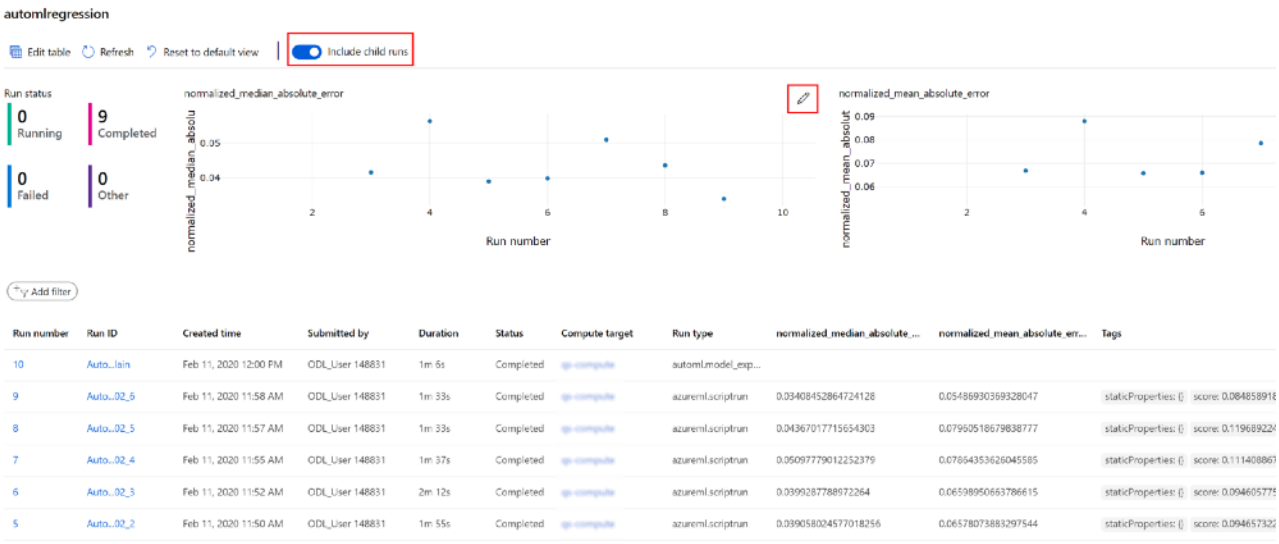
2. In the models list, notice at the top the iteration with the best **normalized root mean square error** score. Note that the normalized root mean square error measures the error between the predicted value and actual value. In this case, the model with the lowest normalized root mean square error is the best model.



3. Select **Experiments** on the left navigation pane and select the experiment `automlregression` to see the list of available runs.



4. Select the option to **Include child runs** to be able to examine model performance for the primary metric of different runs. By default, the left chart describes the `normalized_median_absolute_error` value for each run. Select the pen icon on the right corner of the `normalized_median_absolute_error` chart to configure the `normalized_root_mean_square_error` metric representation.



Next Steps

Congratulations! You have trained a simple time-series forecasting model using automated machine learning in the visual interface. 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.

Lab 18 - Managing a compute instance

Machine learning requires several tools to prepare data, and train and deploy models. Most of the work usually takes place within web-based, interactive notebooks, such as Jupyter notebooks. Although notebooks are lightweight and easily run in a web browser, you still need a server to to host them.
So, the setup process for most users is to install several applications and libraries on a machine, configure the environment settings, then loaed any additional resources to begin working within notebooks or integrated development environments (IDEs). All this setup takes time, and there is sometimes a fair amount of troubleshooting involved to make sure you have the right combination of software versions that are compatible with one another.

What if you could use a ready-made environment that is pre-optimized for your machine learning development?

Azure Machine Learning compute instance provides this type of environment for you, and is fully managed, meaning you don't have to worry about setup and applying patches and updates to the underlying virtual machine. Plus, since it is cloud-based, you can run it from anywhere and from any machine. All you need to do is specify the type of virtual machine, including GPUs and I/O-optimized options, then you have what you need to start working.

The managed services, such as computer instance and compute cluster, can be used as a training compute target to scale out training resources to handle larger data sets. When you are ready to run your experiments and build your models, you need to specify a compute target. Compute targets are compute resources where you run your experiments or host your service deployment. The target may be your local machine or a cloud-based resource. This is another example of where managed services like compute instance and computer cluster really shine.

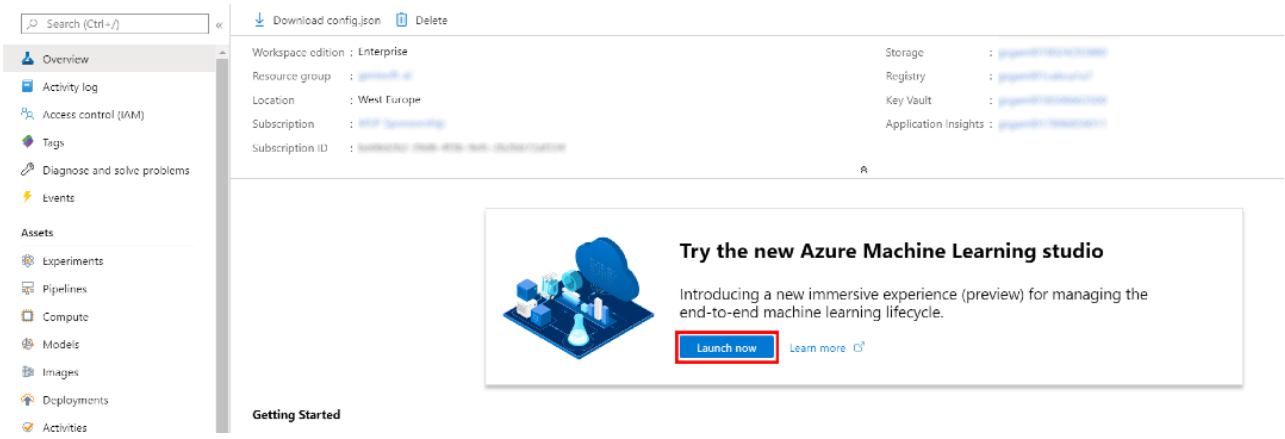
A managed compute resource is created and managed by Azure Machine Learning. This compute is optimized for machine learning workloads.
Azure Machine Learning compute clusters and compute instances are the only managed computes. Additional managed compute resources may be added in the future.

Overview

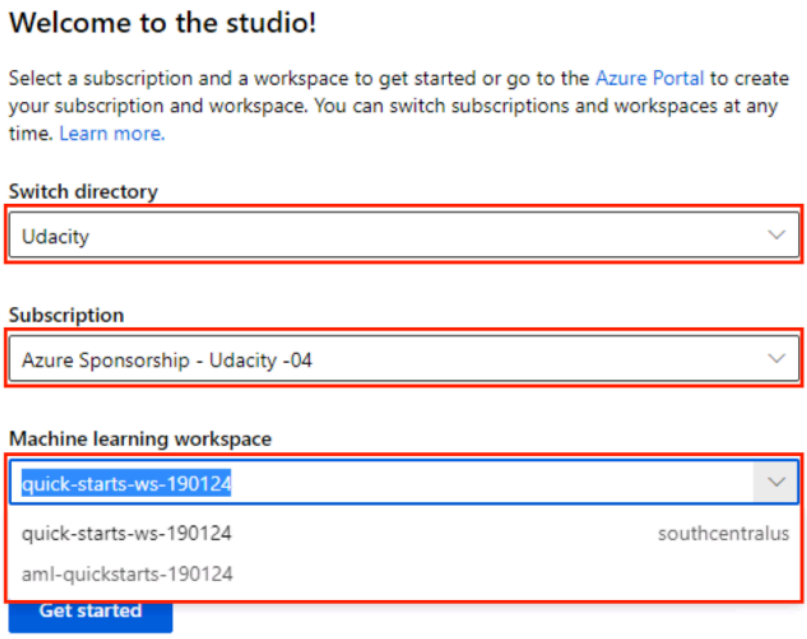
In this lab, you will explore different actions you can take to manage a compute instance in Azure Machine Learning Studio.

Exercise 1: Create New Compute Instance

1. In Azure portal, open the available machine learning workspace.
2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

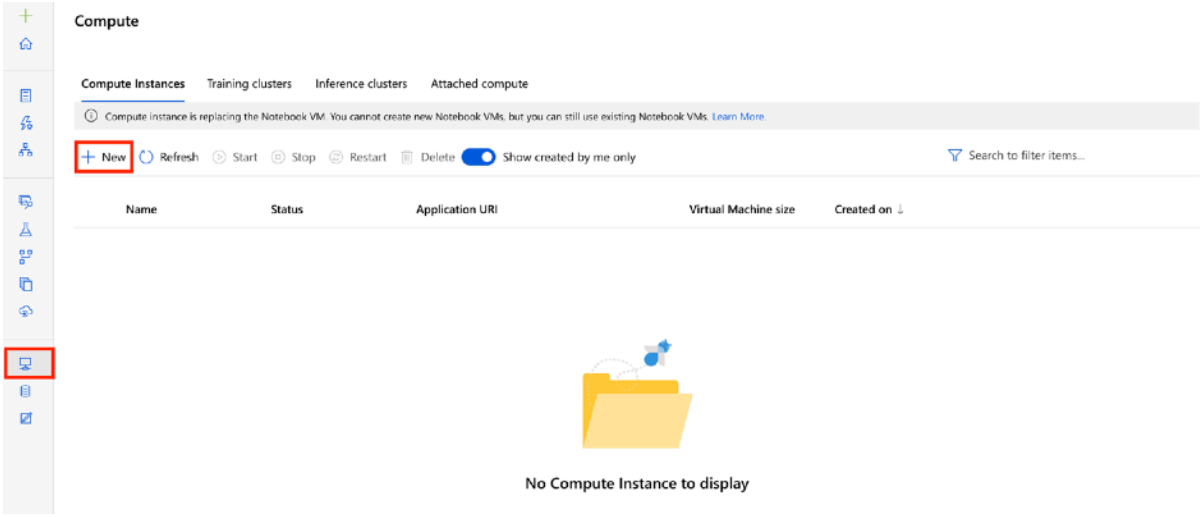


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:

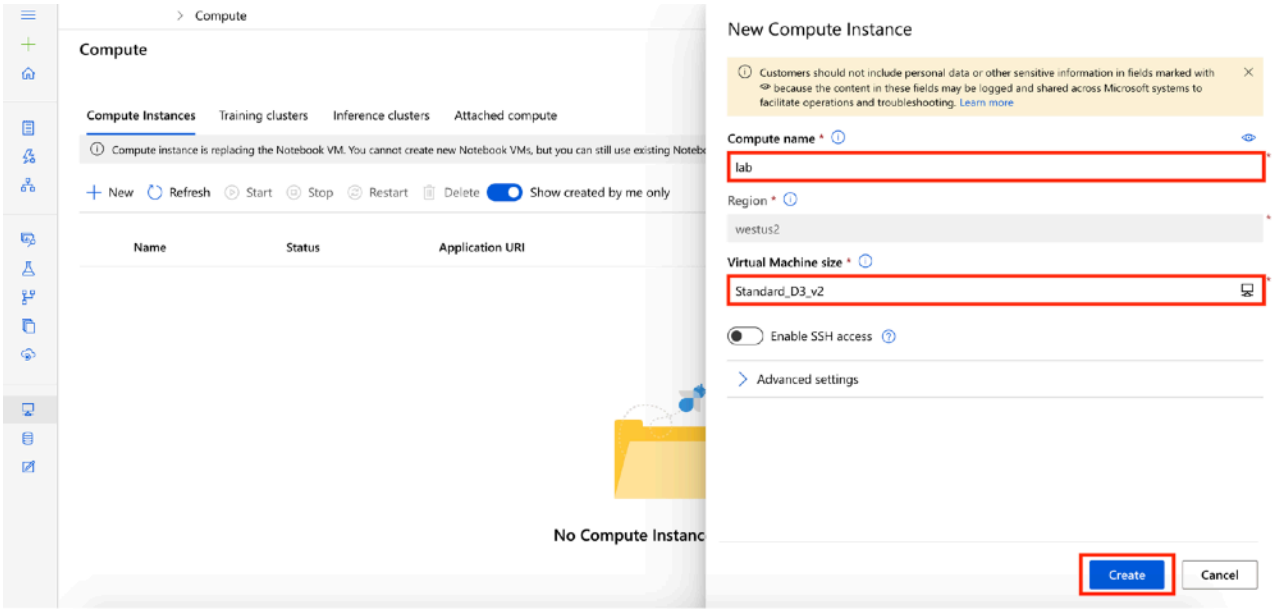


For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

4. From the studio, navigate to **Compute**, then select **+New**.



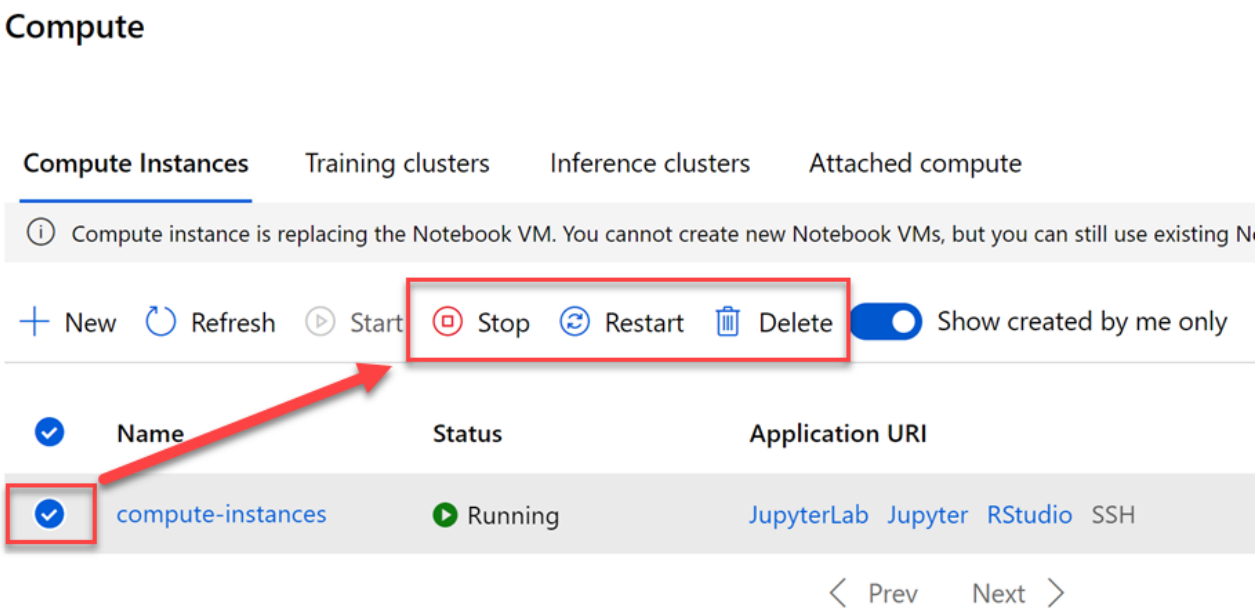
5. In the `New Compute Instance` pane, provide the following information and then select **Create**.
   - Compute name: `provide an unique name`
   - Virtual Machine size: `Standard_D3_v2`



6. It will take couple of minutes for your compute instance to be ready. Wait for your compute instance to be in status `Running`.

Exercise 2: Explore Compute Instances

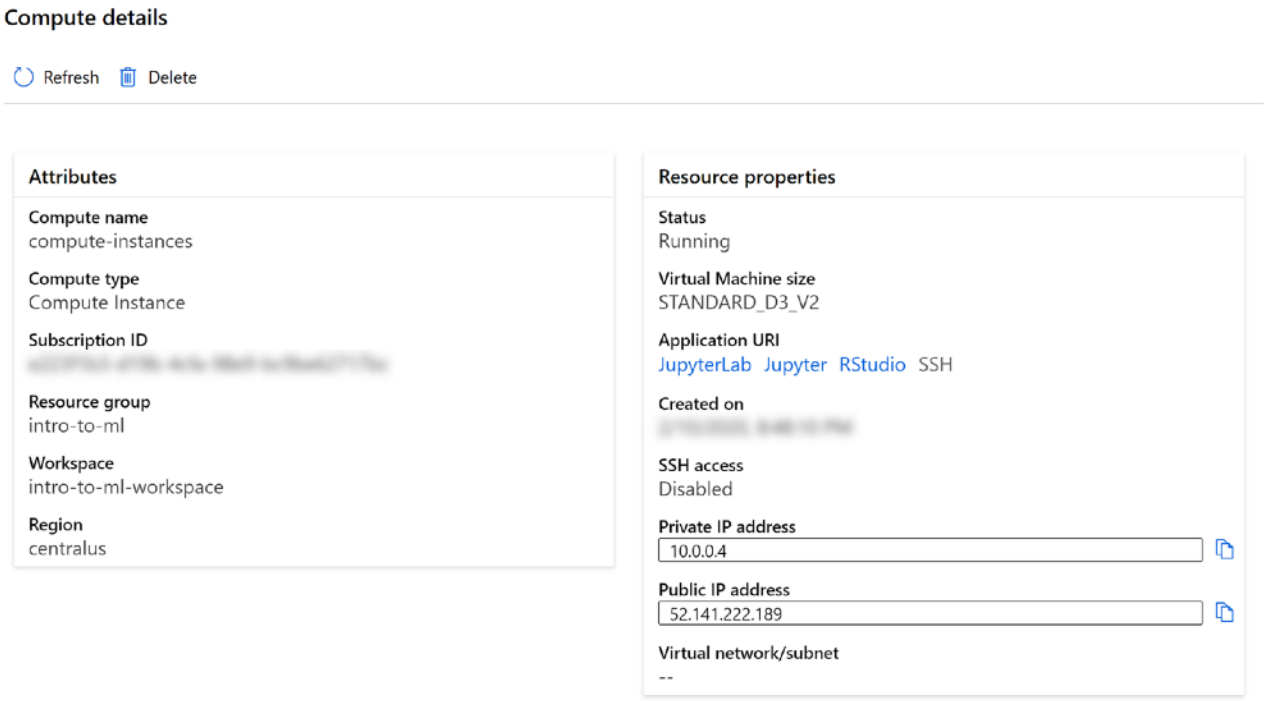
1. Select the radio button next to the name of your compute instance. This will select the instance, as indicated by a checkmark. Selecting your instance in this way enables the toolbar options above that enable you to Stop, Restart, or Delete the instance.



There are different scenarios in which you will want to perform these actions. Here are the actions you can take on a selected compute instance, and what they do:

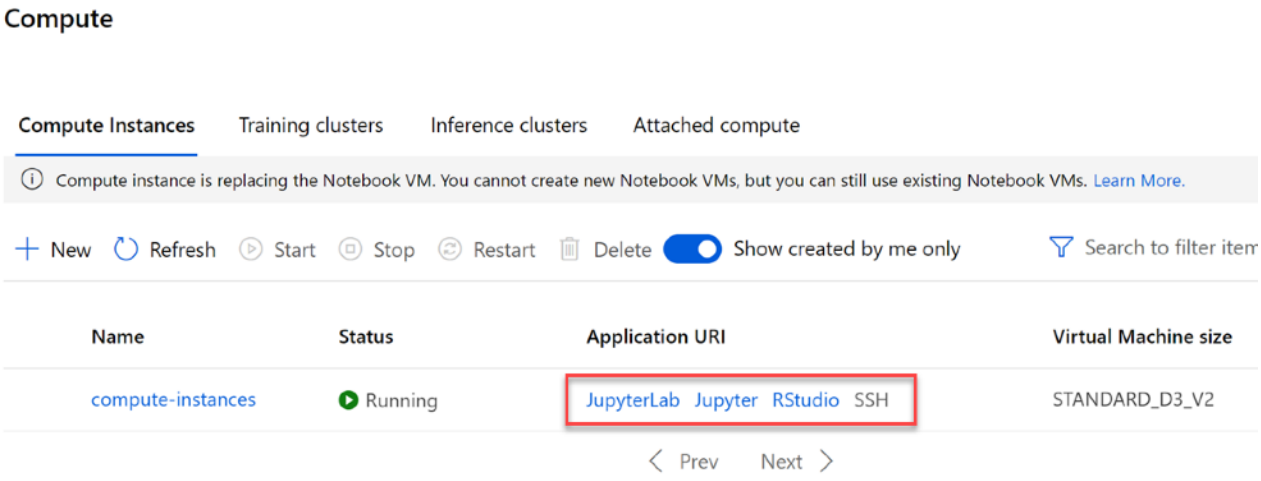
- **Stop**: Since the compute instance runs on a virtual machine (VM), you pay for the instance as long as it is running. Naturally, it needs to run to perform compute tasks, but when you are done using it, be sure to stop it with this option to prevent unnecessary costs.
- **Restart**: Restarting an instance is sometimes necessary after installing certain libraries or extensions. There may be times, however, when the compute instance stops functioning as expected. When this happens, try restarting it before taking further action.
- **Delete**: You can create and delete instances as you see fit. The good news is, all notebooks and R scripts are stored in the default storage account of your workspace in Azure file share, within the "User files" directory. This central storage allows all compute instances in the same workspace to access the same files so you don't lose them when you delete an instance you no longer need.

2. Select the name of your instance. This opens the **Compute details** blade, revealing useful information about your compute instance.



3. The **Attributes** describe the resource details of the compute instance, including the name, type, Azure subscription, the resource group to which it belongs, the Azure Machine Learning workspace that manages it, and the Azure region to which it is deployed. If you need to execute scripts that require details about your compute instance, this is where you can find most of what you need.
The **Resource properties** show the status and configuration of the compute instance, including links to its applications and public and private endpoints. In this screenshot, you will see that SSH access is disabled. You cannot enable SSH access after creating a compute instance. You can only enable this option at the time of creation. SSH access allows you to securely connect to the VM from a terminal or command window. Use the public IP address to connect via SSH or an integrated development environment (IDE) like Visual Studio Code.

4. Navigate back to **Compute**. The compute instance comes preconfigured with tools and environments that enable you to author, train, and deploy models in a fully integrated notebook experience. You access these environments through the **Application URI** links located in the resource properties (as seen in the previous step), and next to each compute instance in the list.



5. Select each of the application links to sign in to the related environment. You may be prompted to select your user account for each application.

Next Steps

Congratulations! You have completed the introduction to managing a compute instance lab. 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.

Lab 19 - Train a machine learning model from a managed notebook environment

So far, the Managed Services for Azure Machine Learning lesson has covered **compute instance** and the benefits it provides through its fully managed environment containing everythng you need to run Azure Machine Learning. Now it is time to gain some hands-on experience by putting a compute instance to work.

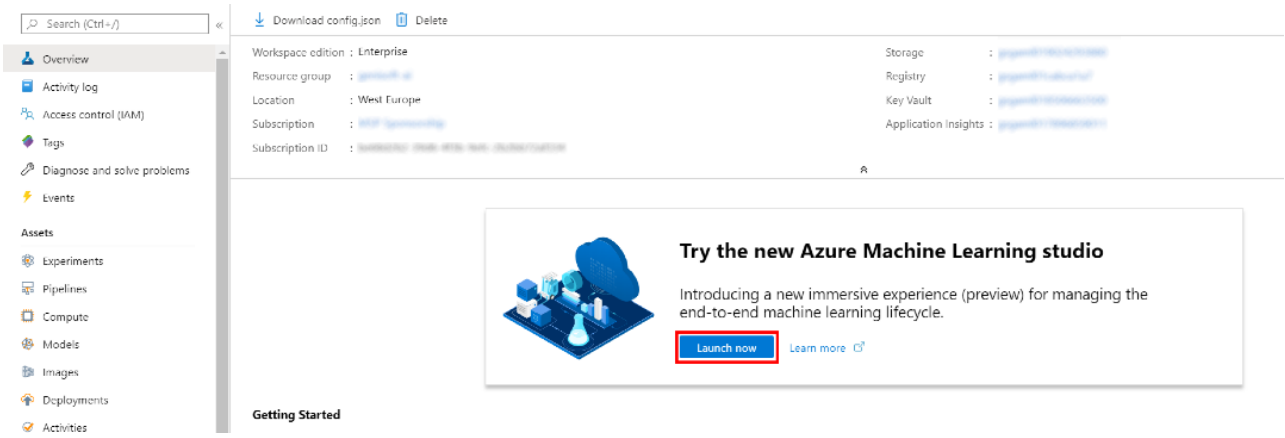
Overview

In this lab, you learn the foundational design patterns in Azure Machine Learning, and train a simple scikit-learn model based on the diabetes data set. After completing this lab, you will have the practical knowledge of the SDK to scale up to developing more-complex experiments and workflows.
In this tutorial, you learn the following tasks:

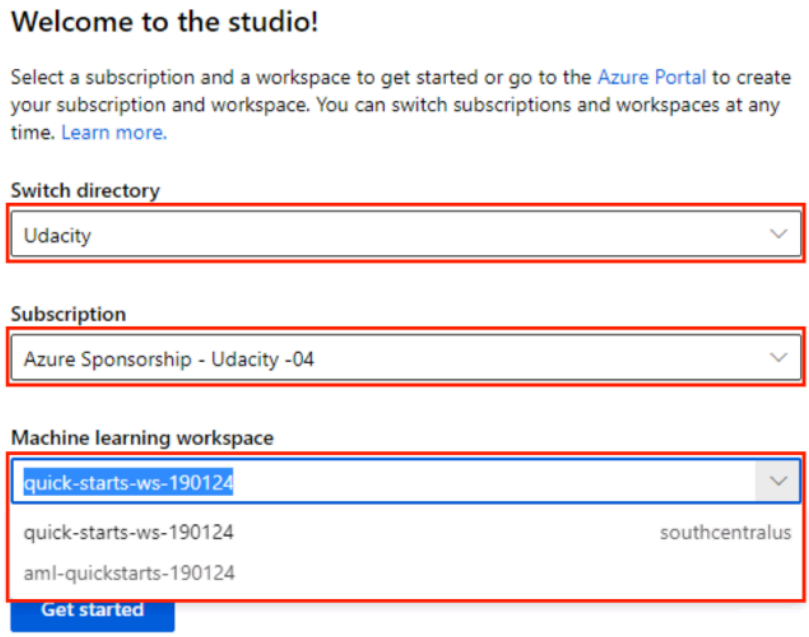
- Connect your workspace and create an experiment
- Load data and train a scikit-learn model

Exercise 1: Run the Notebook for this Lab

1. In Azure portal, open the available machine learning workspace.
2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

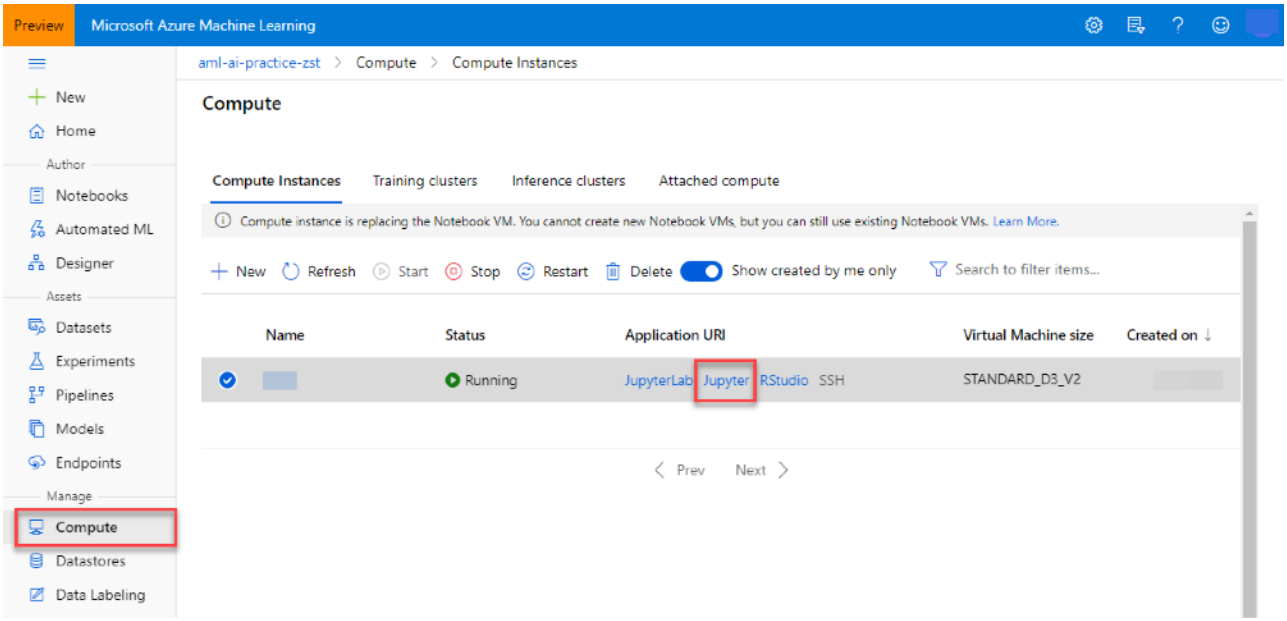


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:

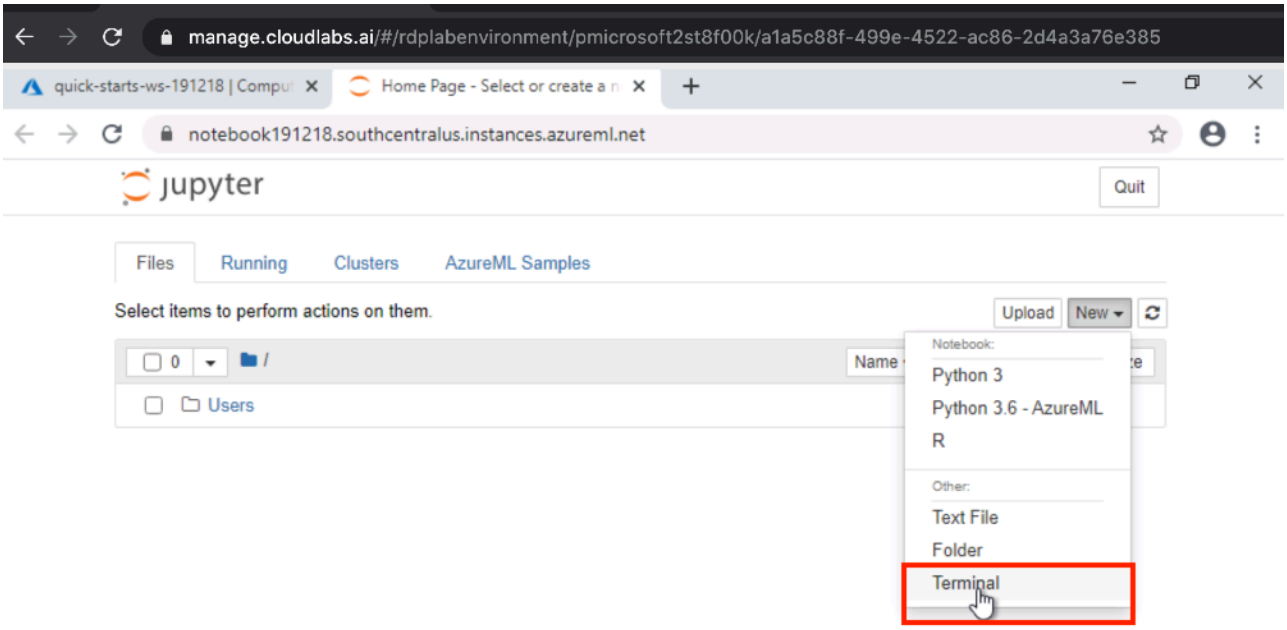


For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

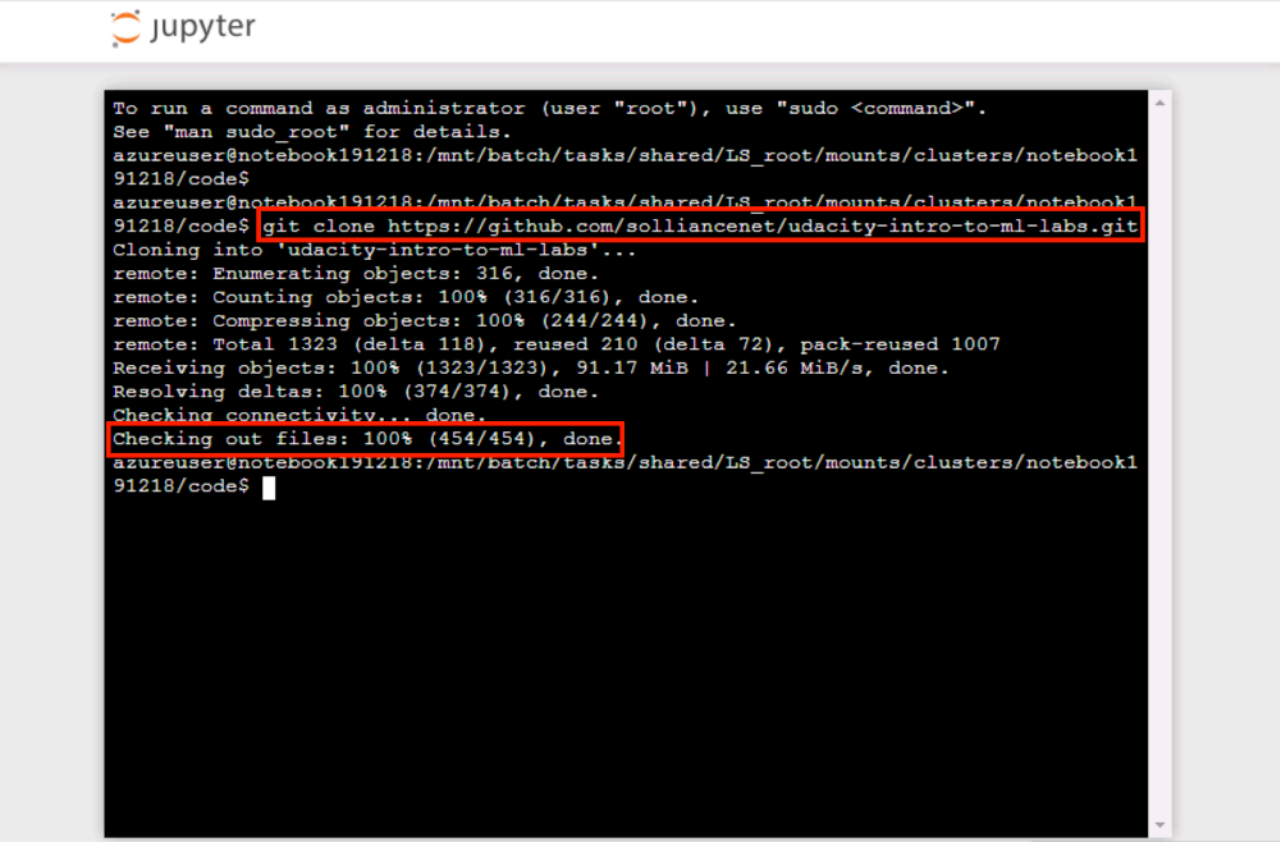
4. From the studio, navigate to **Compute**. Next, for the available Compute Instance, under Application URI select `Jupyter`. Be sure to select `Jupyter` and not `JupterLab`.



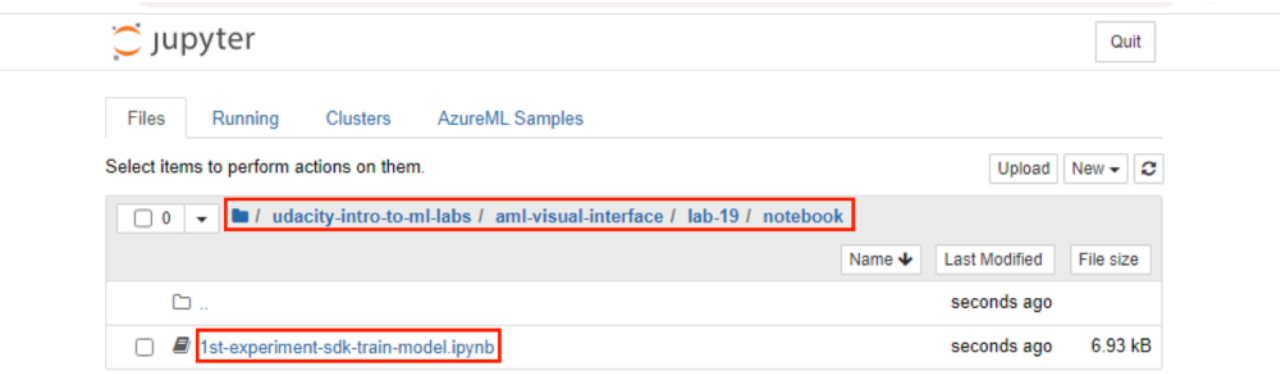
5. From within the Jupyter interface, select **New, Terminal**.



6. In the new terminal window run the following command and wait for it to finish: `git clone https://github.com/solliancenet/udacity-intro-to-ml-labs.git`

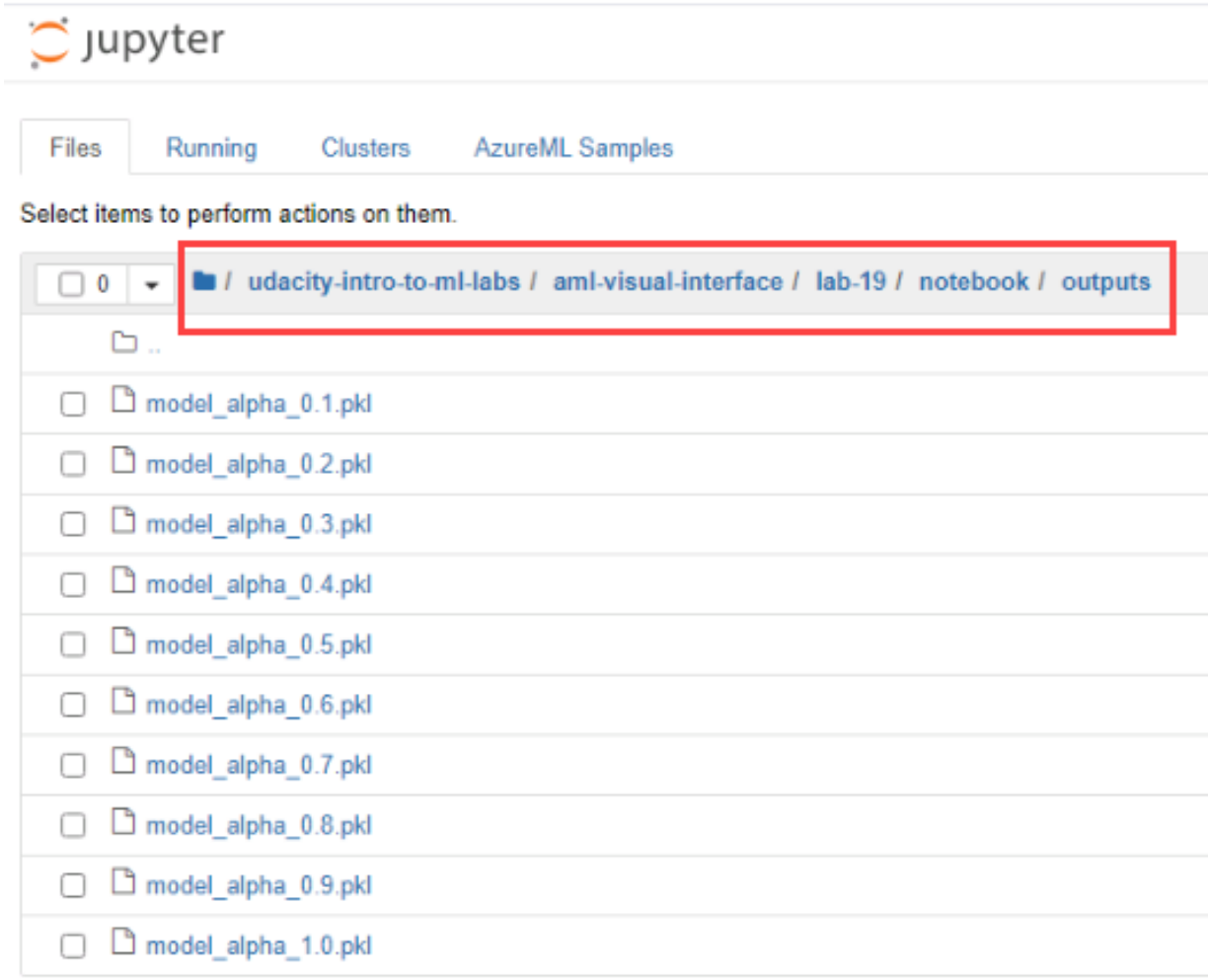


7. From within the Jupyter interface, navigate to directory `udacity-intro-to-ml-labs/aml-visual-interface/lab-19/notebook` and open `1st-experiment-sdk-train-model.ipynb`. This is the Python notebook you will step through executing in this lab.



8. Follow the instructions within the notebook to complete the lab.

9. After completing the notebook, navigate back to the **Notebook** folder, then select the newly created **outputs** sub-folder. Here you will see the trained models (`*.pkl` files) generated by the last cell you executed. In addition, the serialized model is uploaded to each run. This allows you to download the model file from the run in the portal as an alternative to downloading them from this folder.



Next Steps

Congratulations! You have just learned how to use the Jupyter application on a compute instance to train a model. You can now return to the Udacity portal to continue with the lesson.

Lab 20 - Explore experiments and runs

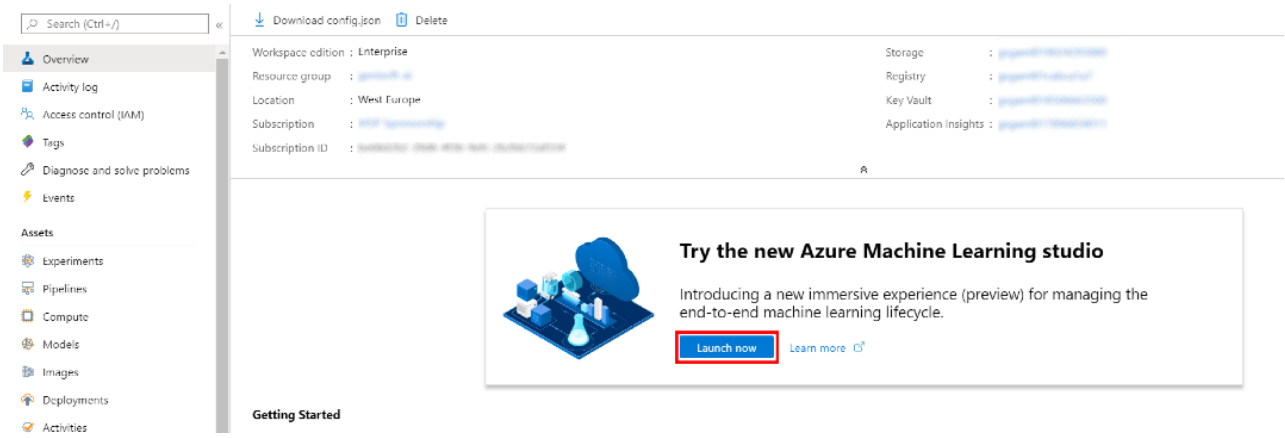
In the previous lab (19), you executed a Jupyter notebook that trained a model through a series of 10 different runs, each with a different alpha hyperparameter applied. These runs were created within the experiment you created at the beginning of the notebook. Because of this, Azure Machine Learning logged the details so you can review the result of each run and see how the alpha value is different between the them.

Overview

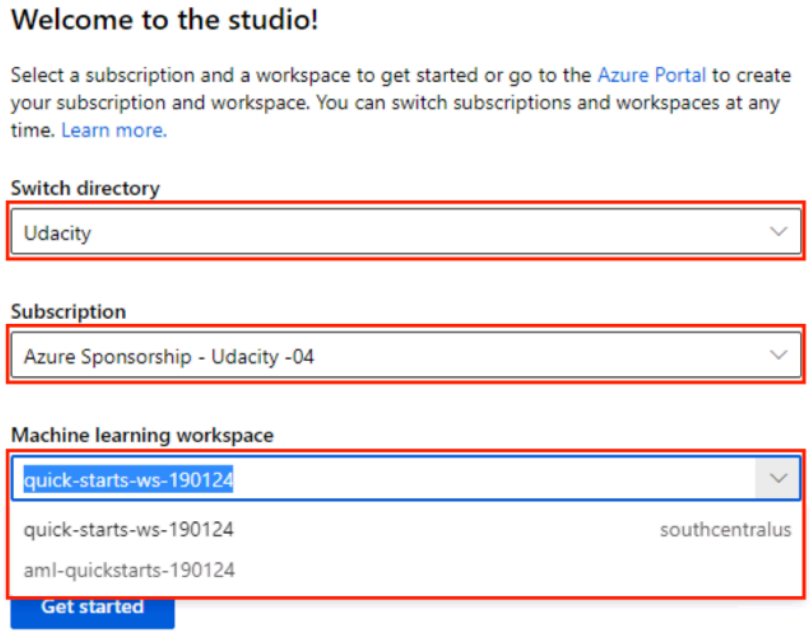
In this lab, you view the experiments and runs executed by a notebook. In the first part of the lab, you will use a notebook to create and run the experiments. In the second part of the lab, you will navigate to the **Experiments** blade in Azure Machine Learning Studio. Here you see all the individual runs in the experiment. Any custom-logged values (alpha_value and rmse, in this case) become fields for each run, and also become available for the charts and tiles at the top of the experiment page. To add a logged metric to a chart or tile, hover over it, click the edit button, and find your custom-logged metric.
When training models at scale over hundreds and thousands of separate runs, this page makes it easy to see every model you trained, specifically how they were trained, and how your unique metrics have changed over time.

Exercise 1: Run the Notebook for this Lab

1. In Azure portal, open the available machine learning workspace.
2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

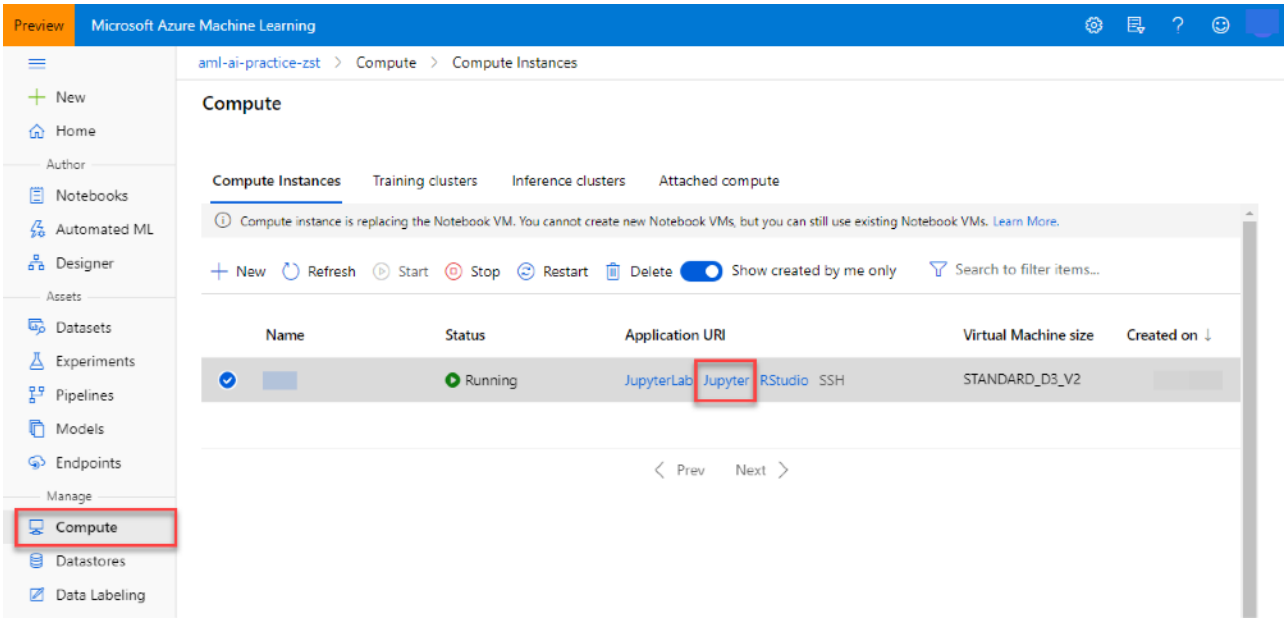


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:

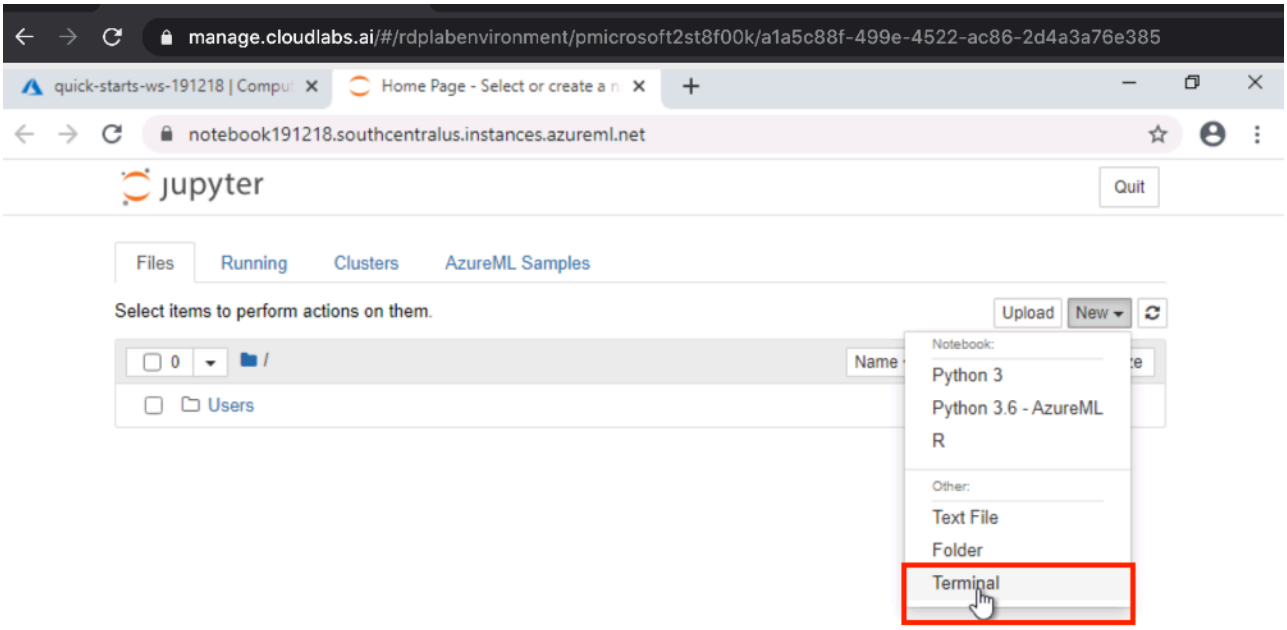


For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

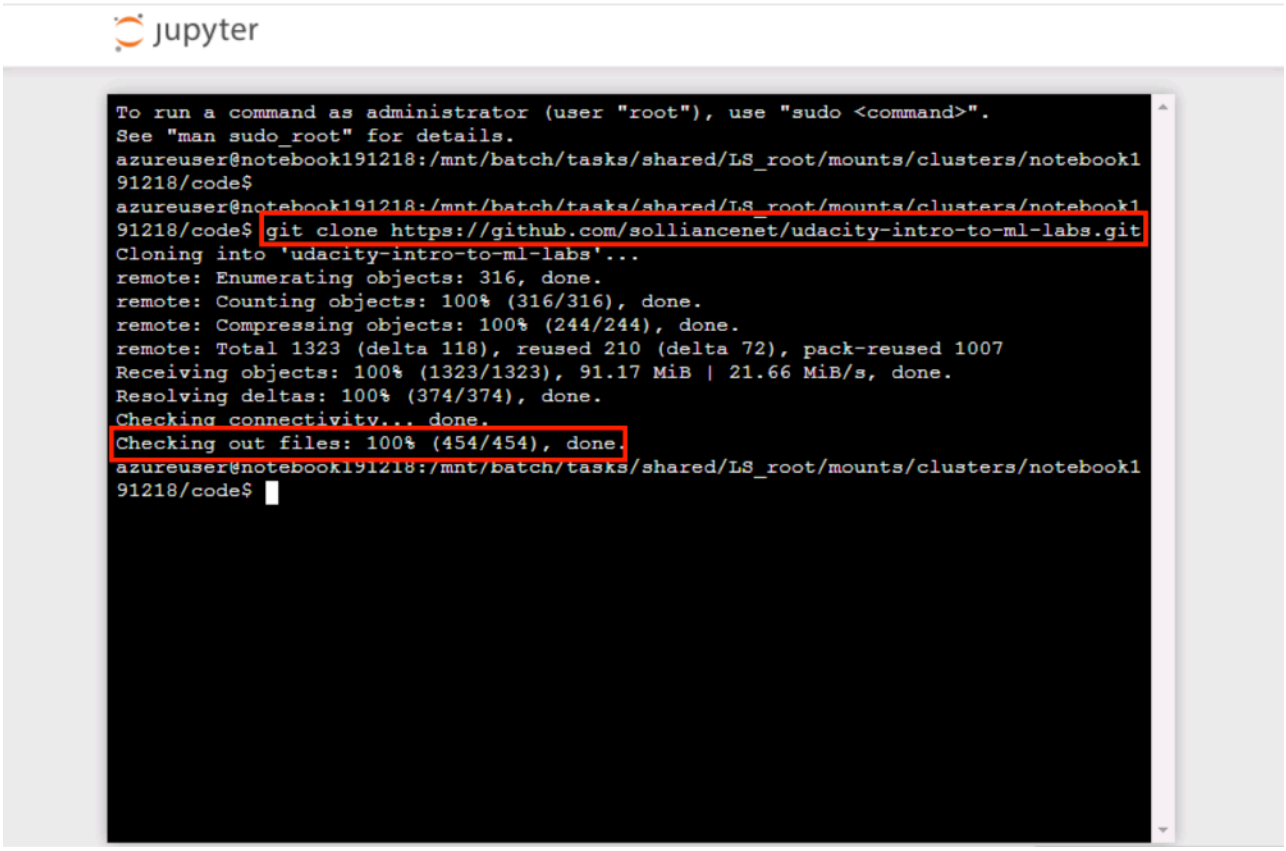
4. From the studio, navigate to **Compute**. Next, for the available Compute Instance, under Application URI select `Jupyter`. Be sure to select `Jupyter` and not `JupterLab`.



5. From within the Jupyter interface, select **New, Terminal**.



6. In the new terminal window run the following command and wait for it to finish: `git clone https://github.com/solliancenet/udacity-intro-to-ml-labs.git\`

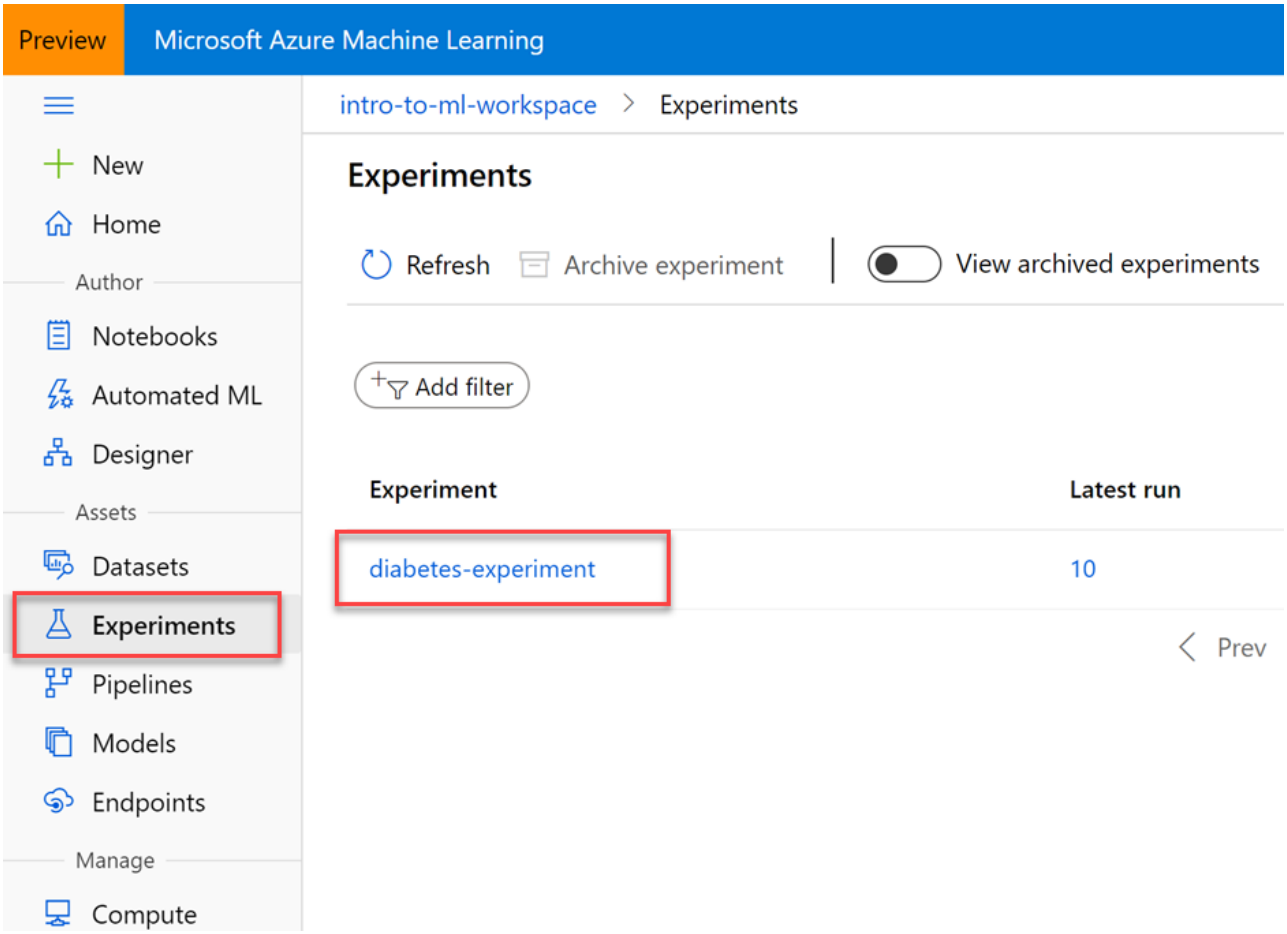


7. From within the Jupyter interface, navigate to directory `udacity-intro-to-ml-labs/aml-visual-interface/lab-20/notebook` and open `1st-experiment-sdk-train-model.ipynb`. This is the Python notebook you will step through executing in this lab.

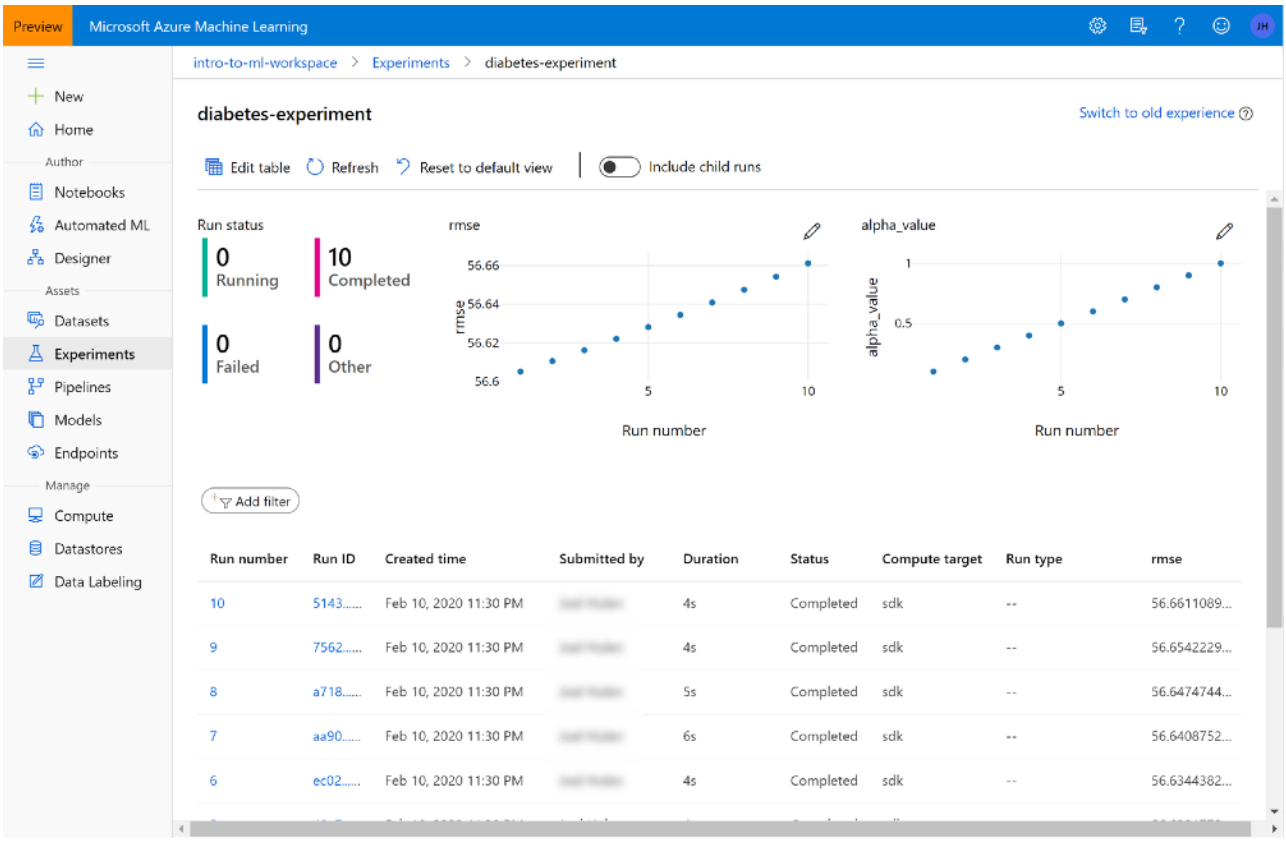
8. Follow the instructions within the notebook to complete the exercise.

Exercise 2: Open Experiments in the portal

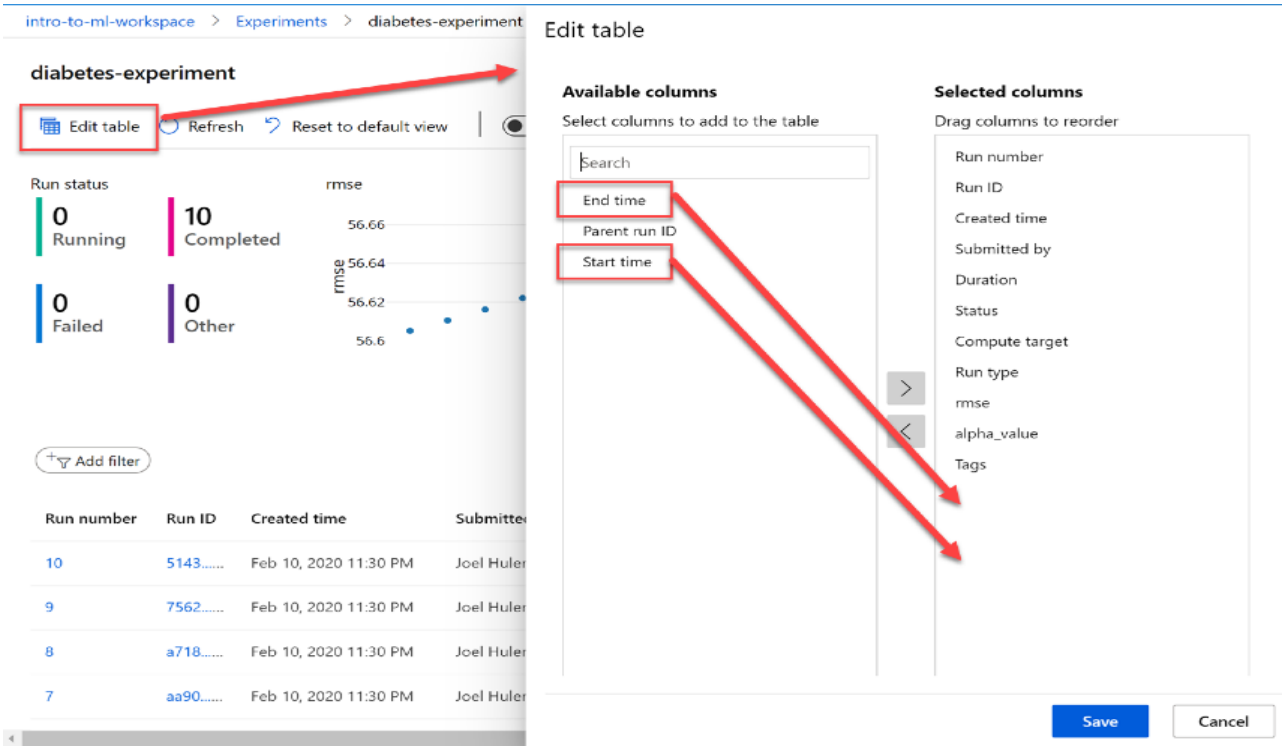
1. Within Azure Machine Learning Studio, select **Experiments** in the left-hand menu, then select the **diabetes-experiment** submitted by the notebook you executed in the previous lab (19).



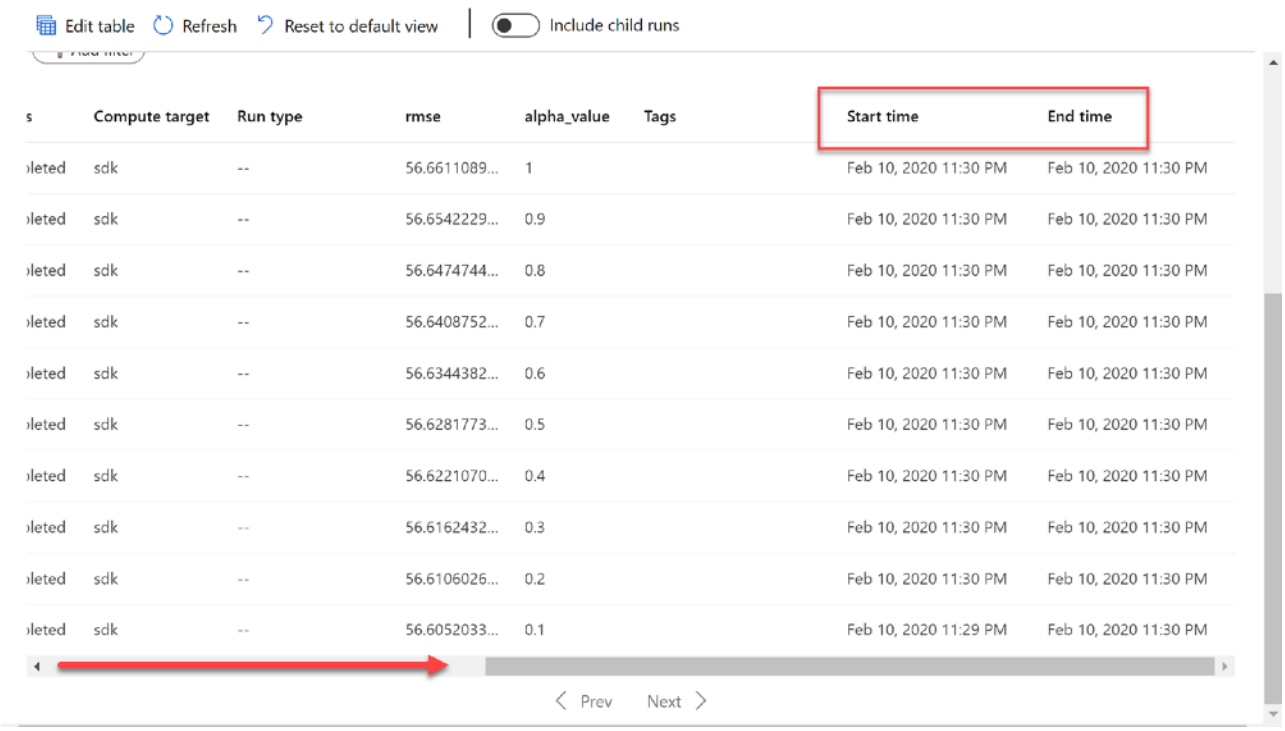
2. Here you can view details about the experiment and each of its runs, which created a new version of the model.



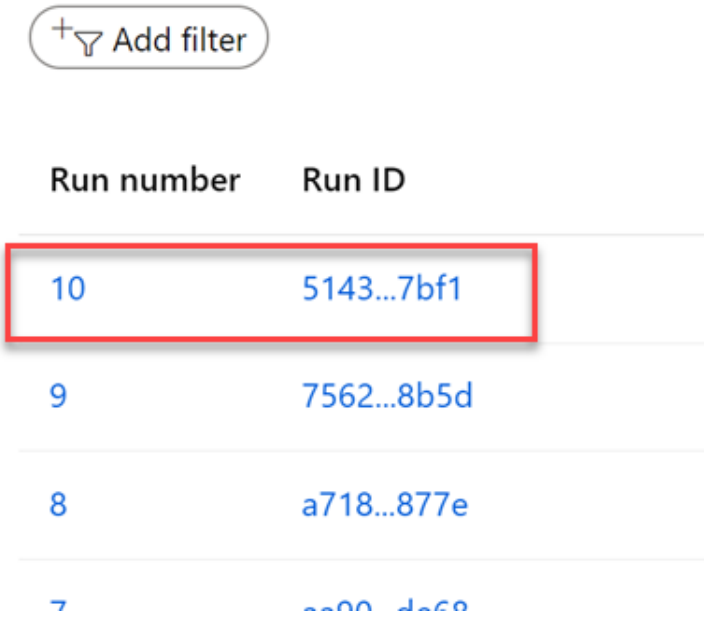
3. Select **Edit table** in the top toolbar. In the Edit table dialog that appears, add the **End time** and **Start time** columns to the Selected columns list, then select **Save**.



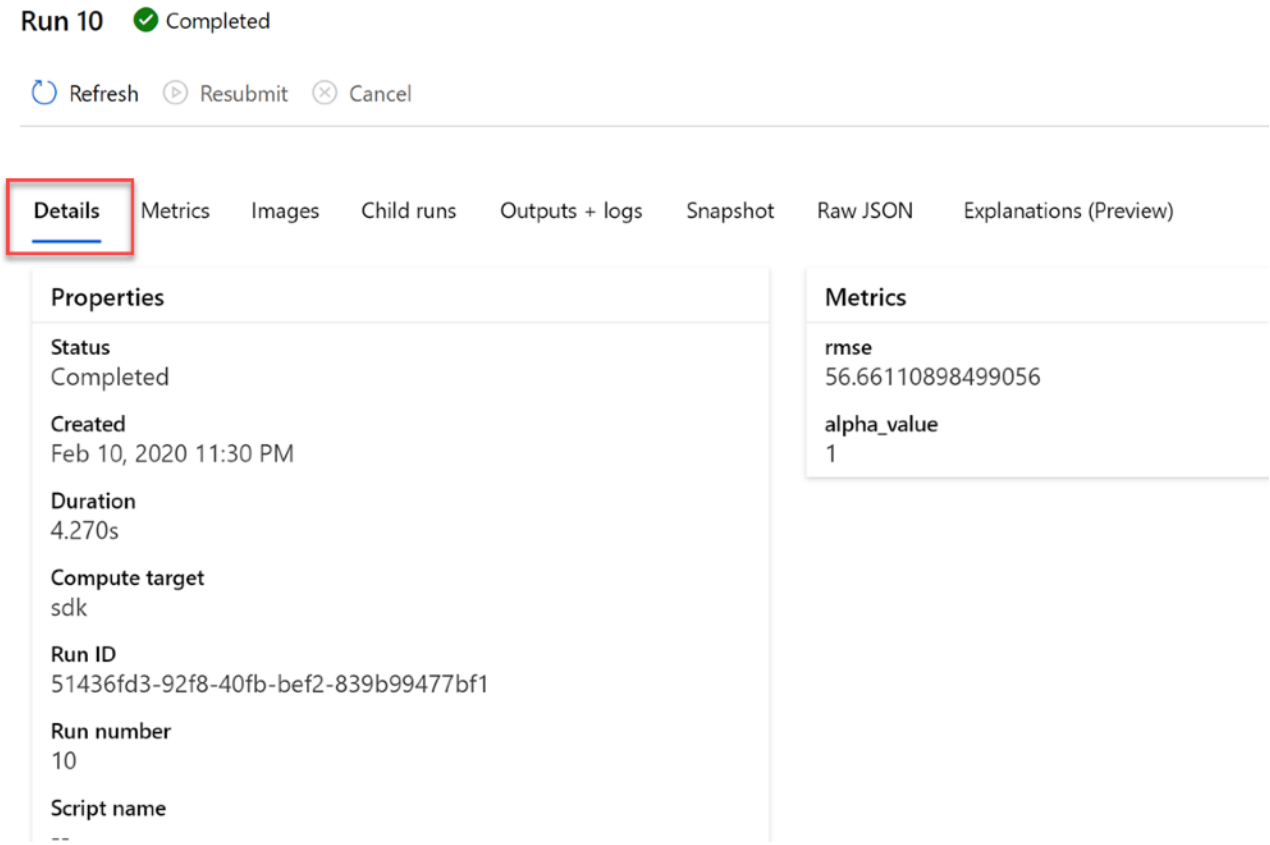
Depending on your screen resolution, you might need to scroll down the table to see the bottom horizontal scrollbar. When you scroll all the way to the right, you will see the new columns you added.



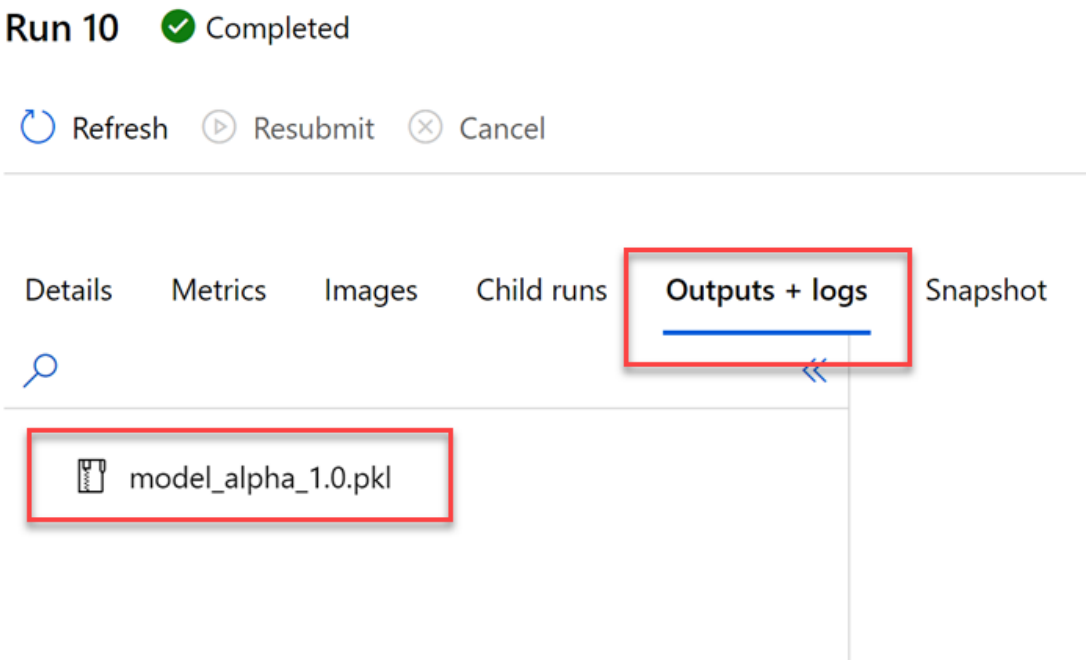
4. Select either the **Run number** or the **Run ID** of one of the runs to view its details. Both links on a run display the same dialog.



5. The **Details** tab shows you more detailed information about each run, including the run time and metrics.



6. Select the **Outputs + logs** tab. You see the `.pkl` file for the model that was uploaded to the run during each training iteration. This lets you download the model file rather than having to retrain it manually.



Next Steps

Congratulations! You have just learned how to use the Azure Machine Learning SDK to help you explain what influences the predictions a model makes. You can now return to the Udacity portal to continue with the lesson.

Lab 21 - Deploy a trained model as a webservice

In previous lessons, we spent much time talking about training a machine learning model, which is a multi-step process involving data preparation, feature engineering, training, evaluation, and model selection. The model training process can be very compute-intensive, with training times spanning across many hours, days, or weeks depending on the amount of data, type of algorithm used, and other factors. A trained model, on the other hand, is used to make decisions on new data quickly. In other words, it infers things about new data it is given based on its training. Making these decisions on new data on-demand is called real-time inferencing.

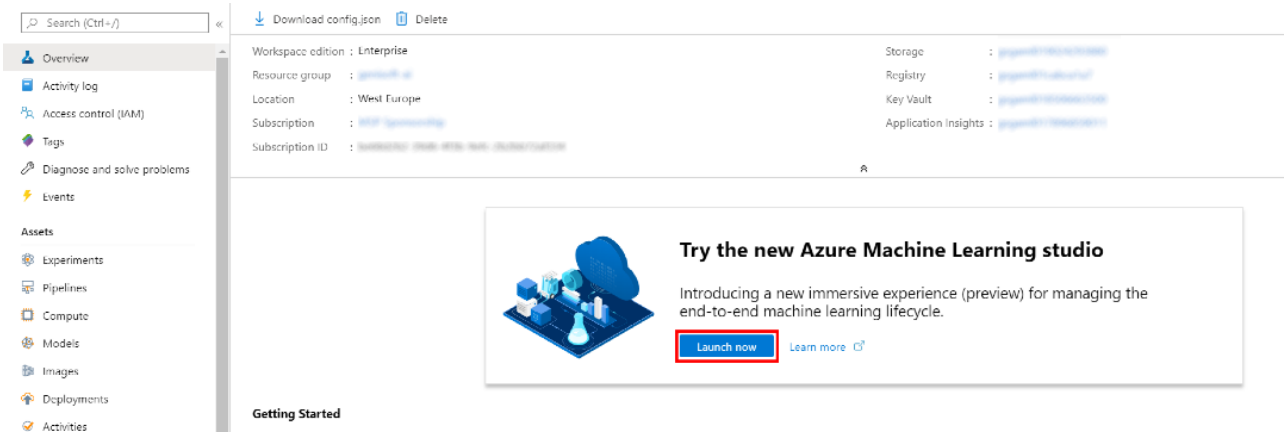
Overview

In this lab, you learn how to deploy a trained model that can be used as a webservice, hosted on an Azure Kubernetes Service (AKS) cluster. This process is what enables you to use your model for real-time inferencing.
The Azure Machine Learning designer simplifies the process by enabling you to train and deploy your model without writing any code.

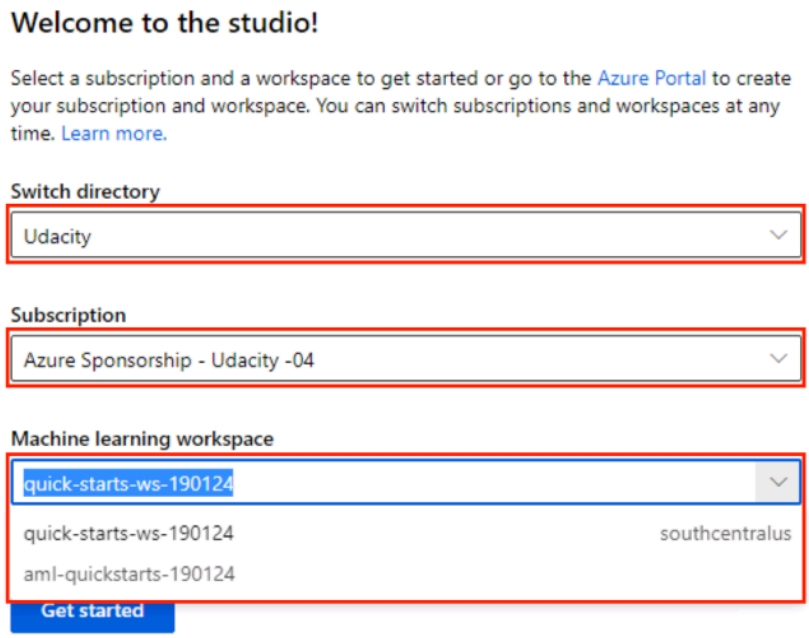
Exercise 1: Open a sample training pipeline

Task 1: Open the 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.

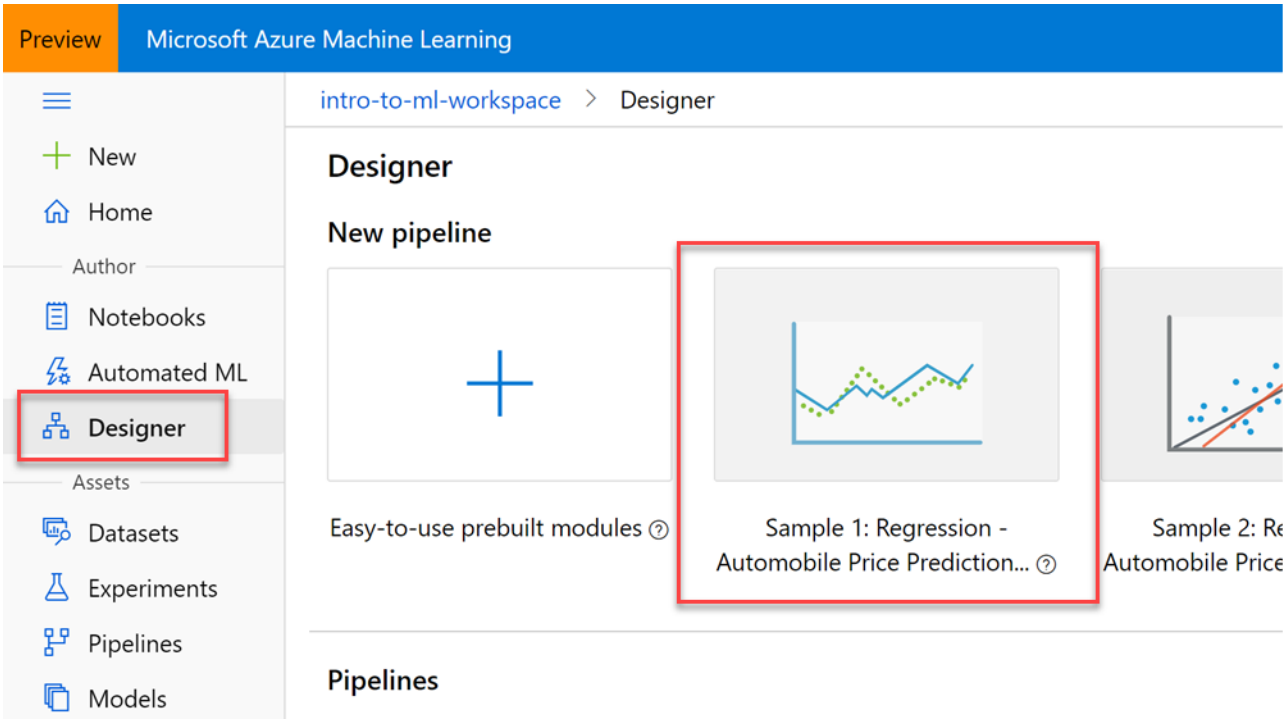


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



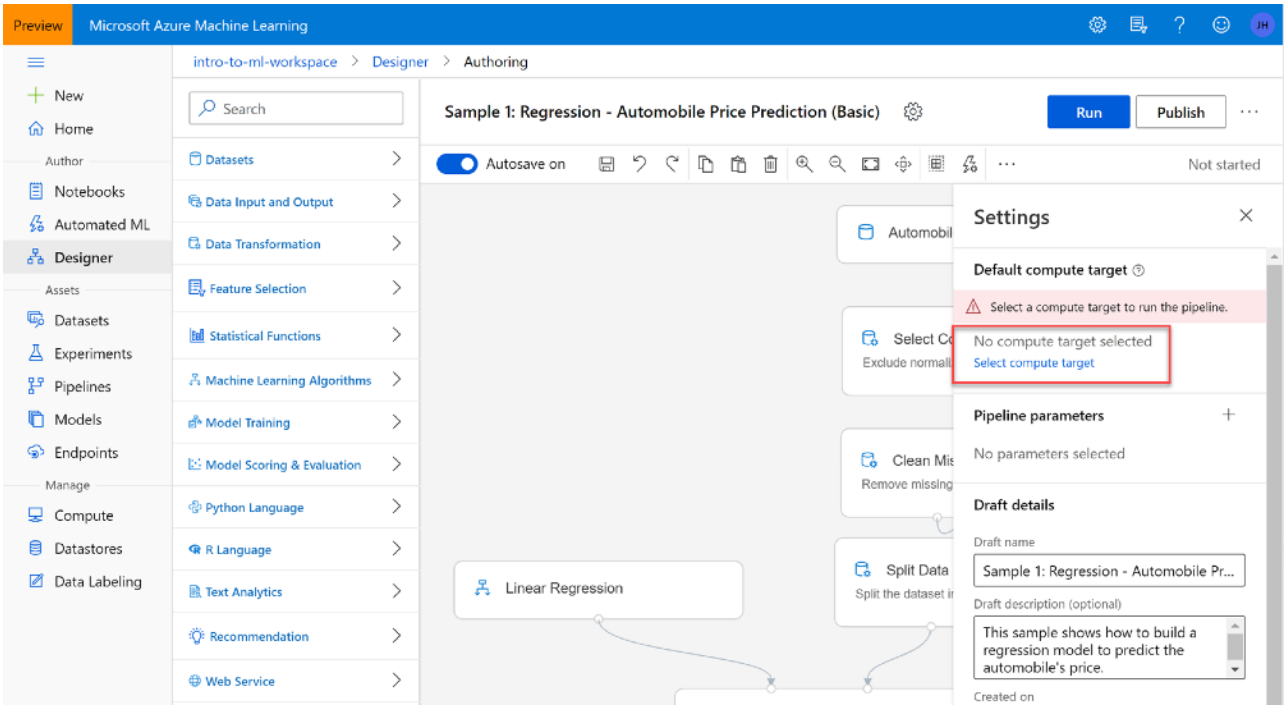
For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

4. From the studio, select **Designer** in the left-hand menu. Next, select **Sample 1: Regression - Automobile Price Prediction (Basic)** under the **New pipeline** section. This will open a `visual pipeline authoring editor`.

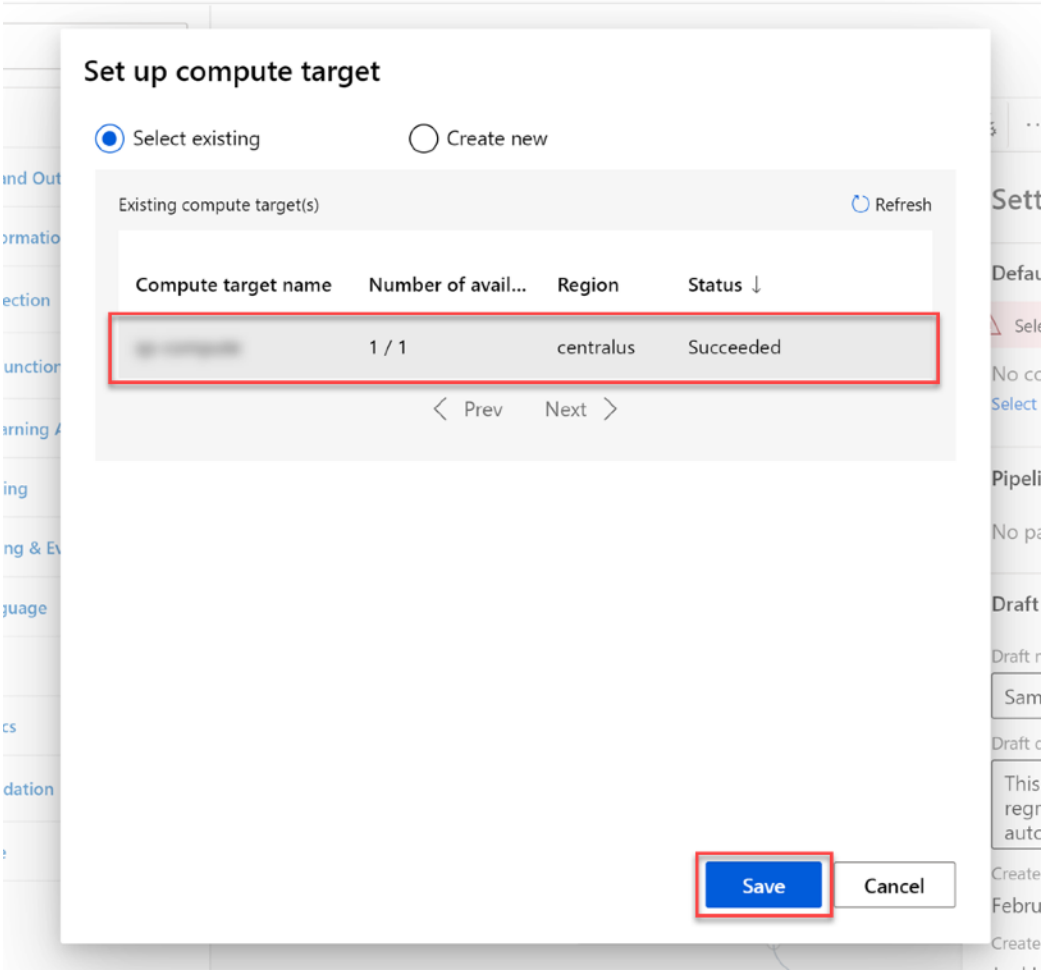


Task 2: Setup the compute target

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



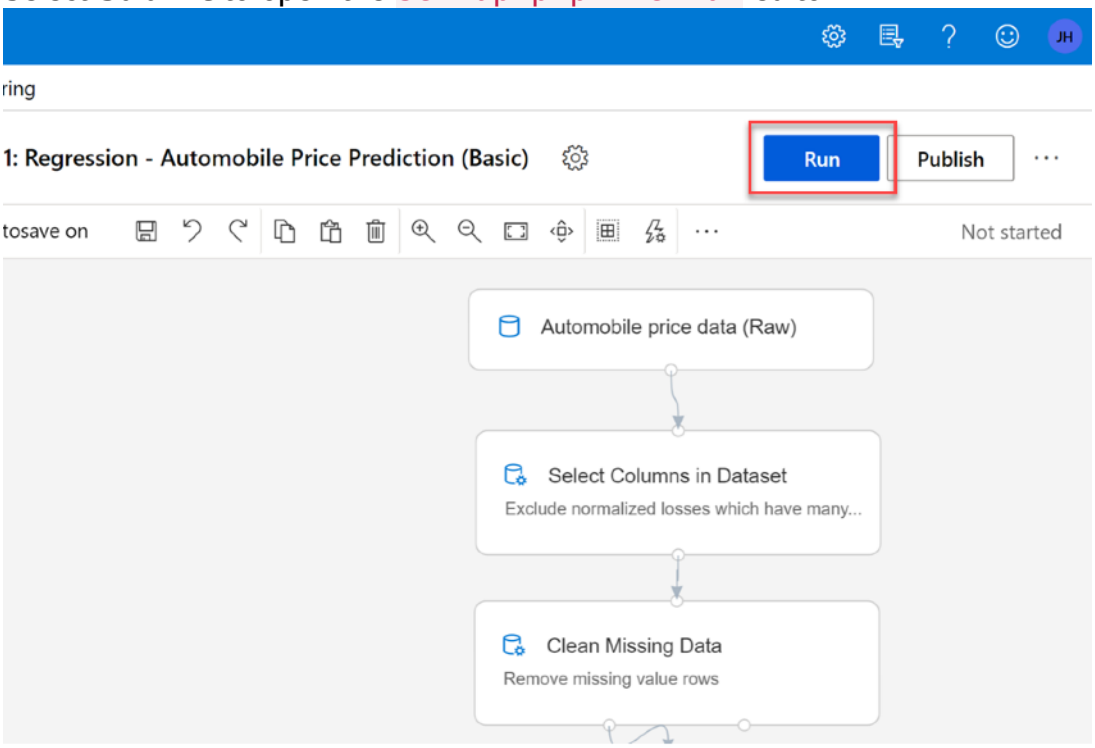
2. In the `Set up compute target` editor, select the existing compute target, then select **Save**.



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.

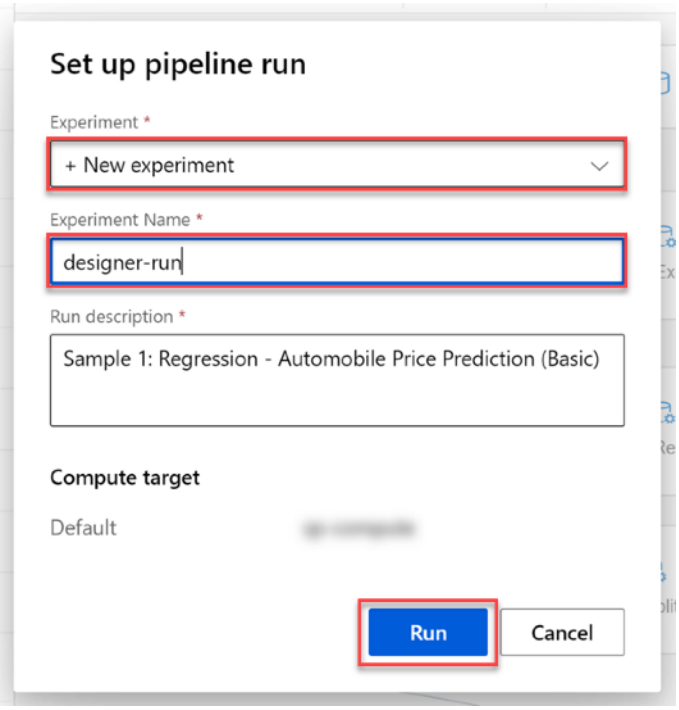
Task 3: Create a new experiment and submit the pipeline

1. Select **Submit** to open the `Set up pipline 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`: **designer-run**, and then select **Submit**.

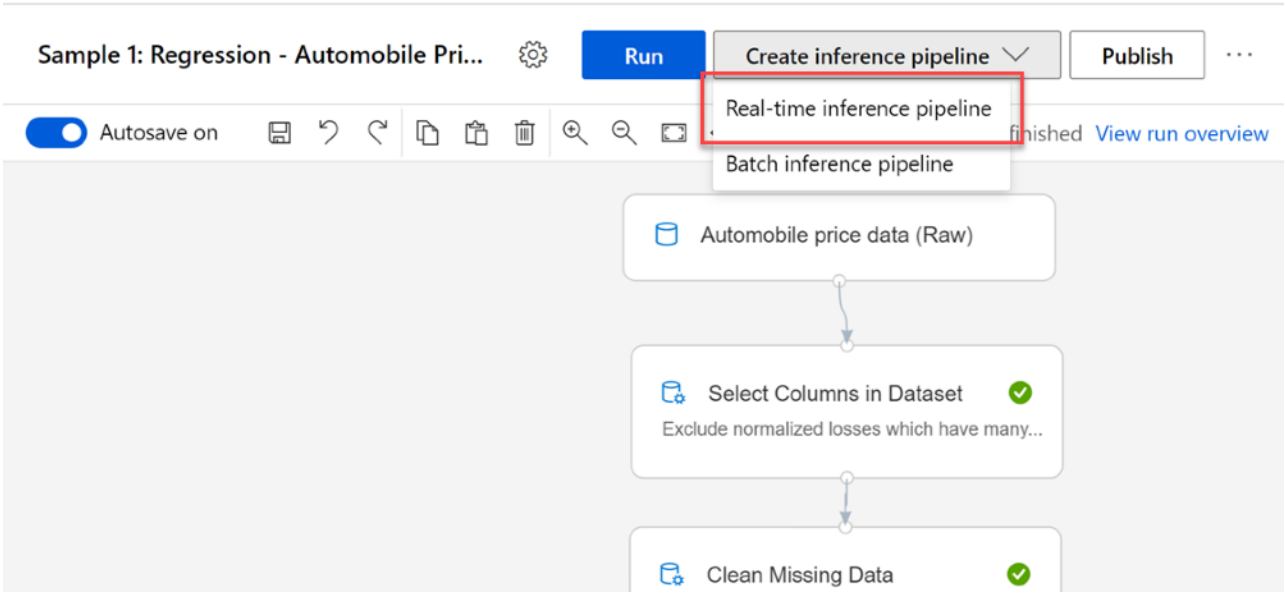


3. Wait for the pipeline run to complete. It will take around **10 minutes** to complete the run.

Exercise 2: Real-time inference pipeline

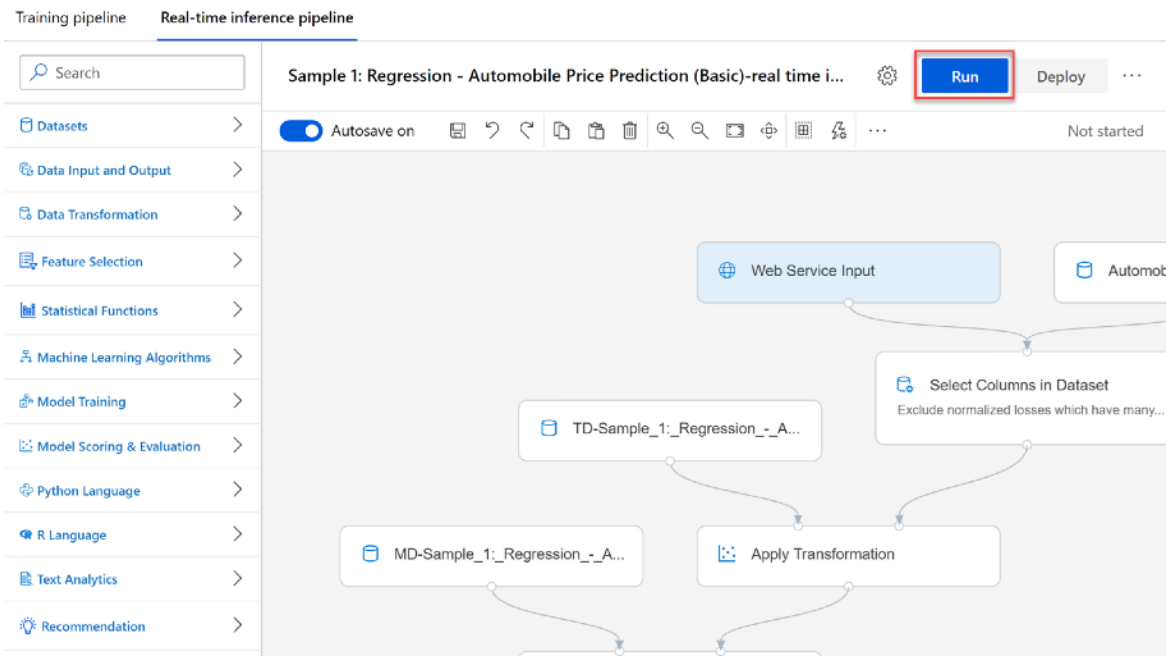
Task 1: Create pipeline

1. Select **Create inference pipeline**, then select **Real-time inference pipeline** from the list to create a new inference pipeline.



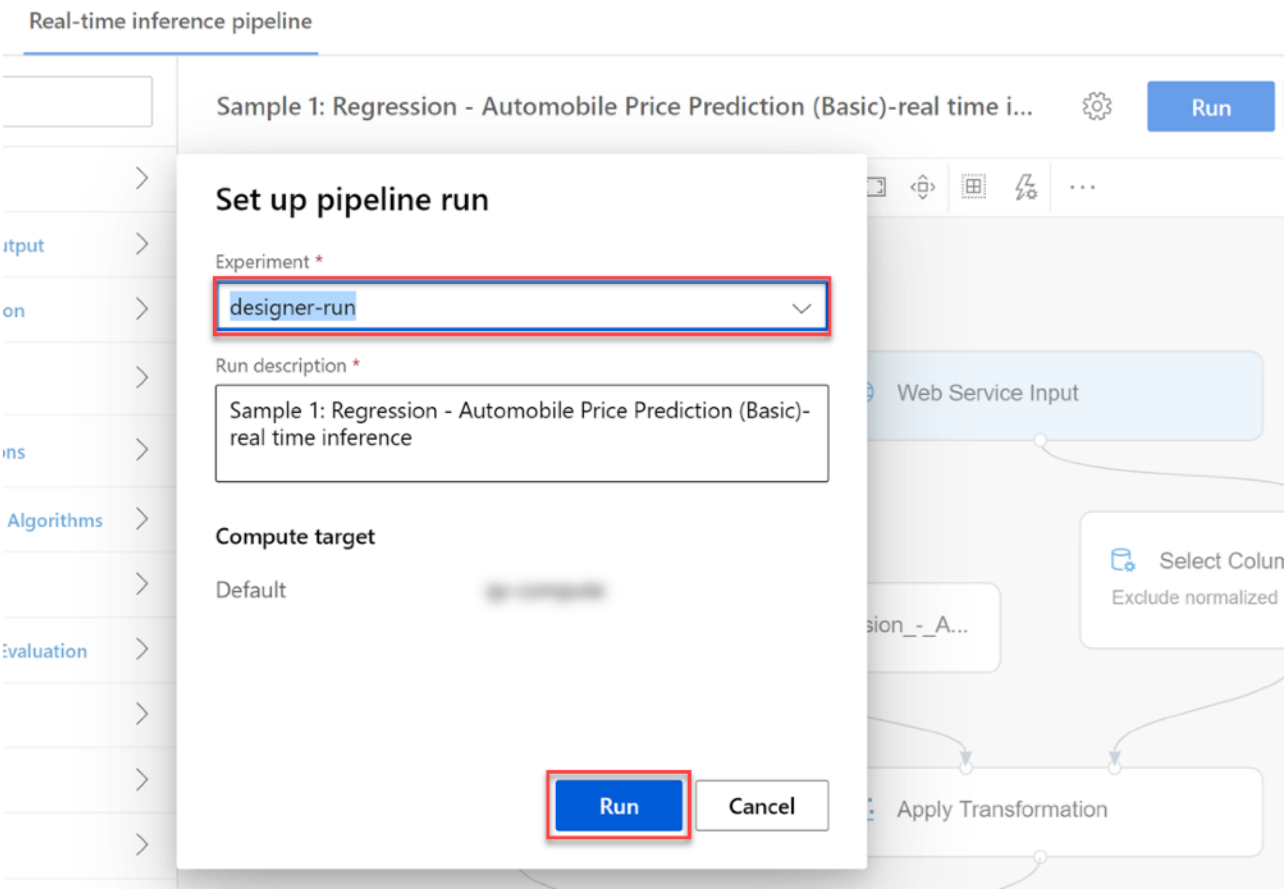
Task 2: Submit the pipeline

1. Select **Submit** to open the `Set up 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 **Select existing**, then select the experiment you created in an earlier step: **designer-run**. Select **Submit** to start the pipeline.

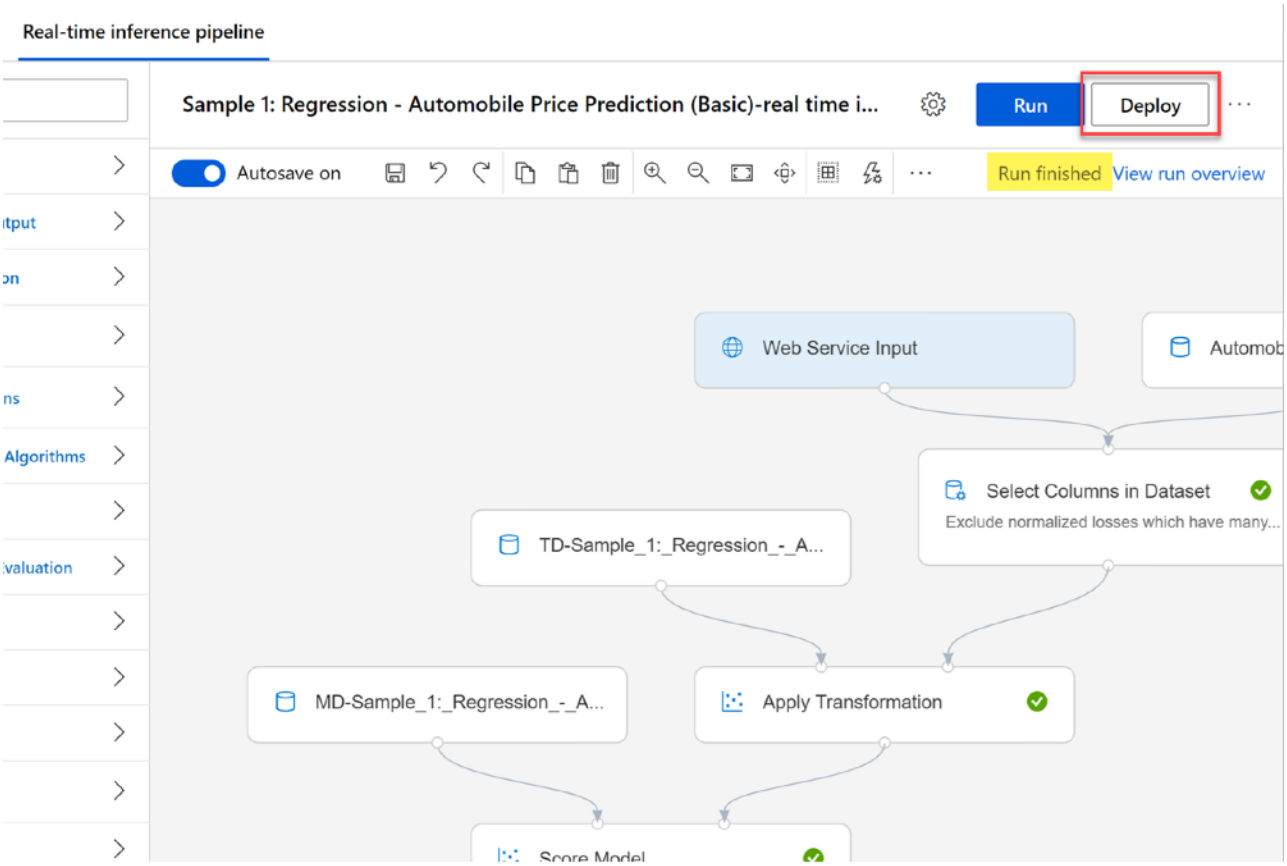


3. Wait for pipeline run to complete. It will take around **7 minutes** to complete the run.

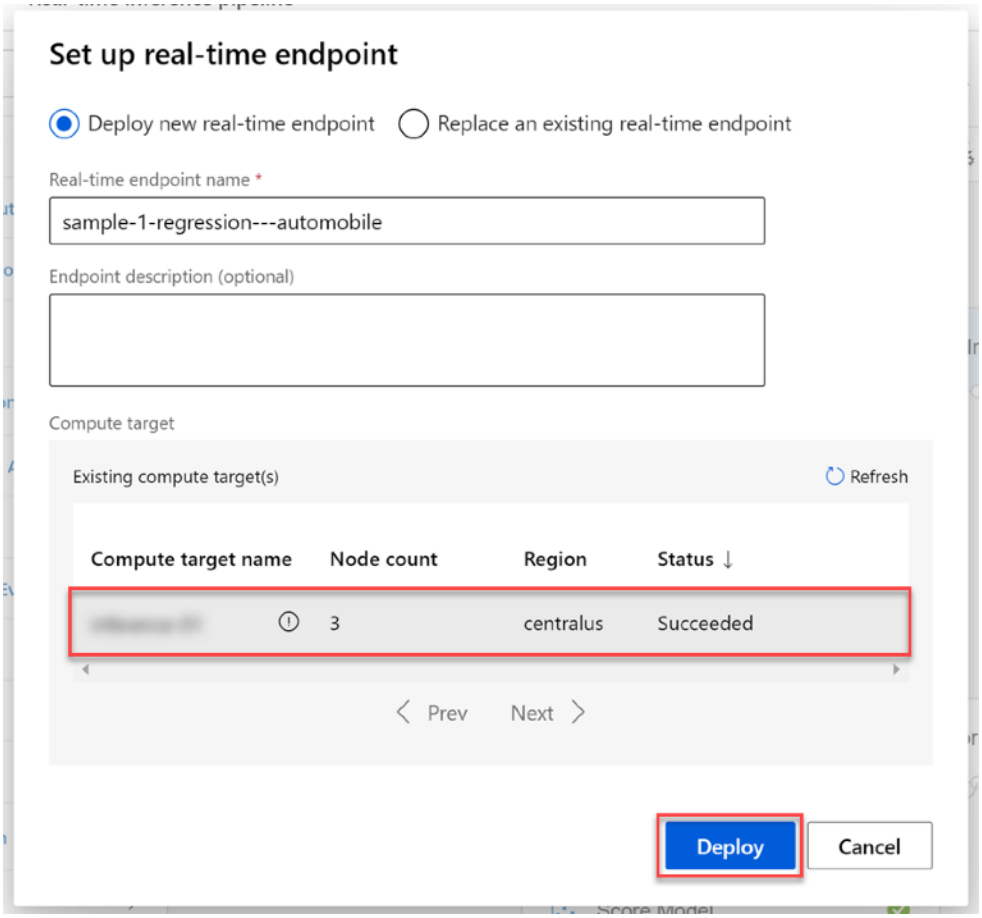
Exercise 3: Deploy web service on Azure Kubernetes Service compute

Task 1: Deploy the web service

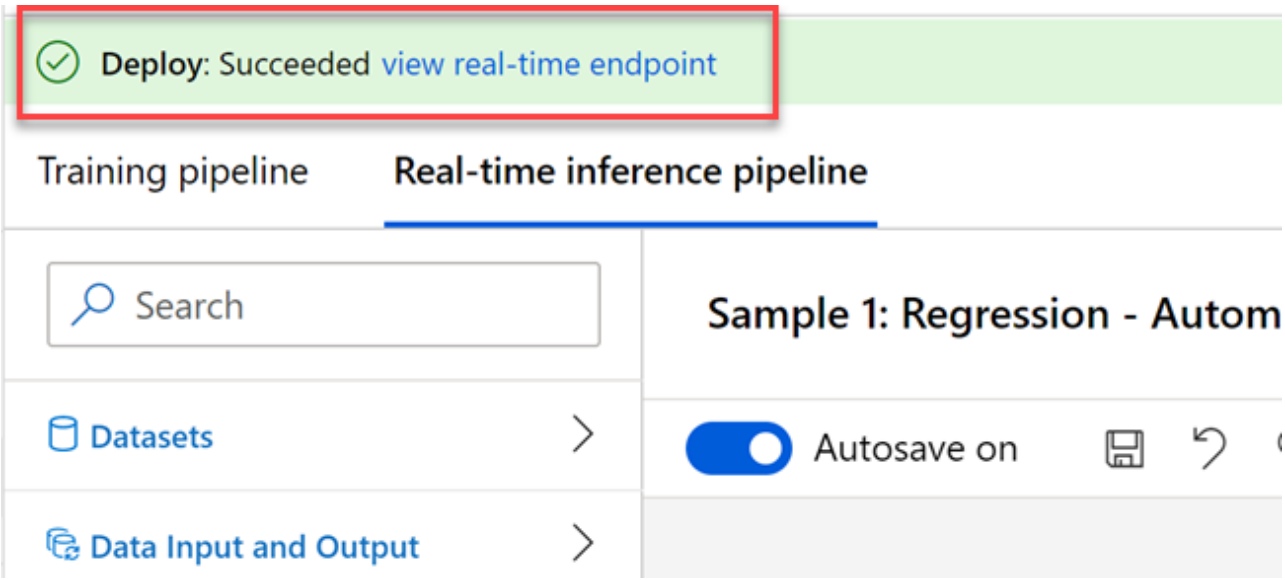
1. After the inference pipeline run is finished, select **Deploy** to open the `Set up real-time endpoint` editor.



2. In the `Set up real-time endpoint` editor, select your **existing compute target**, then select **Deploy**.

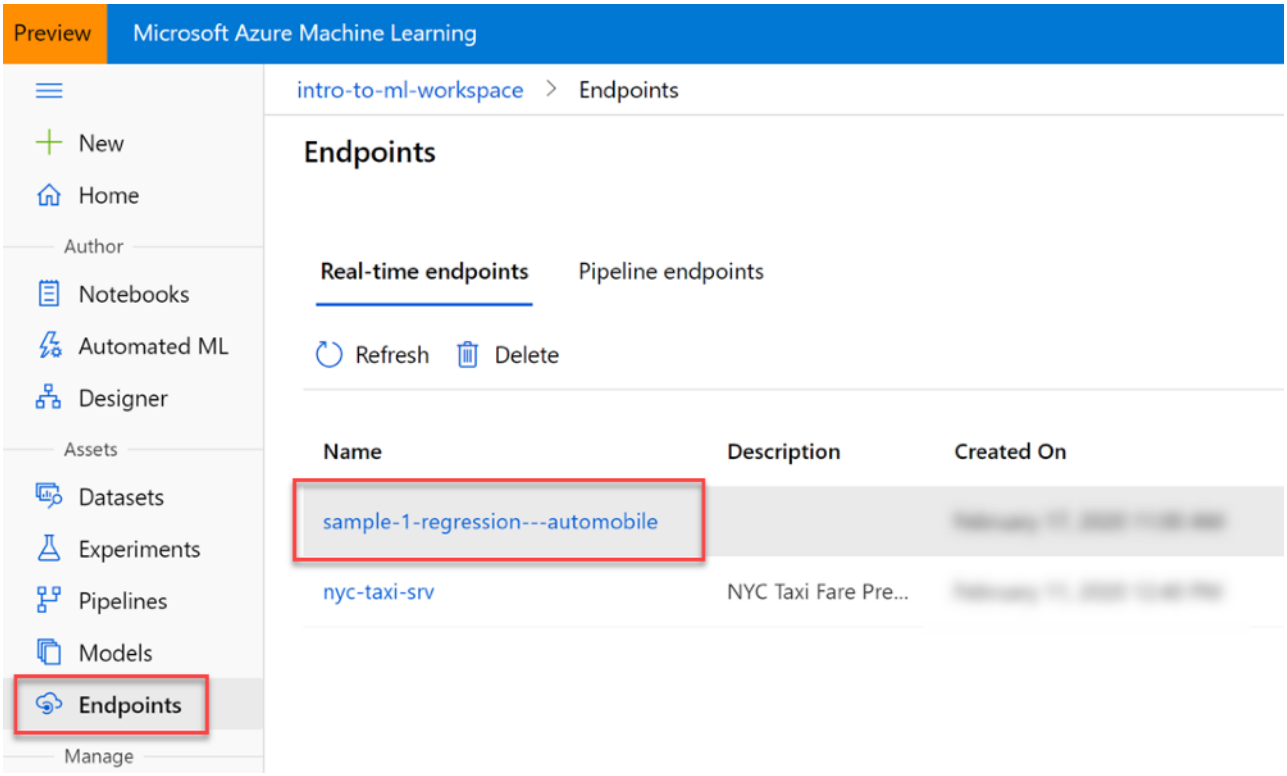


3. Wait for the deployment to complete. The status of the deployment can be observed above the `Pipeline Authoring Editor`.



Task 2: Review deployed web service

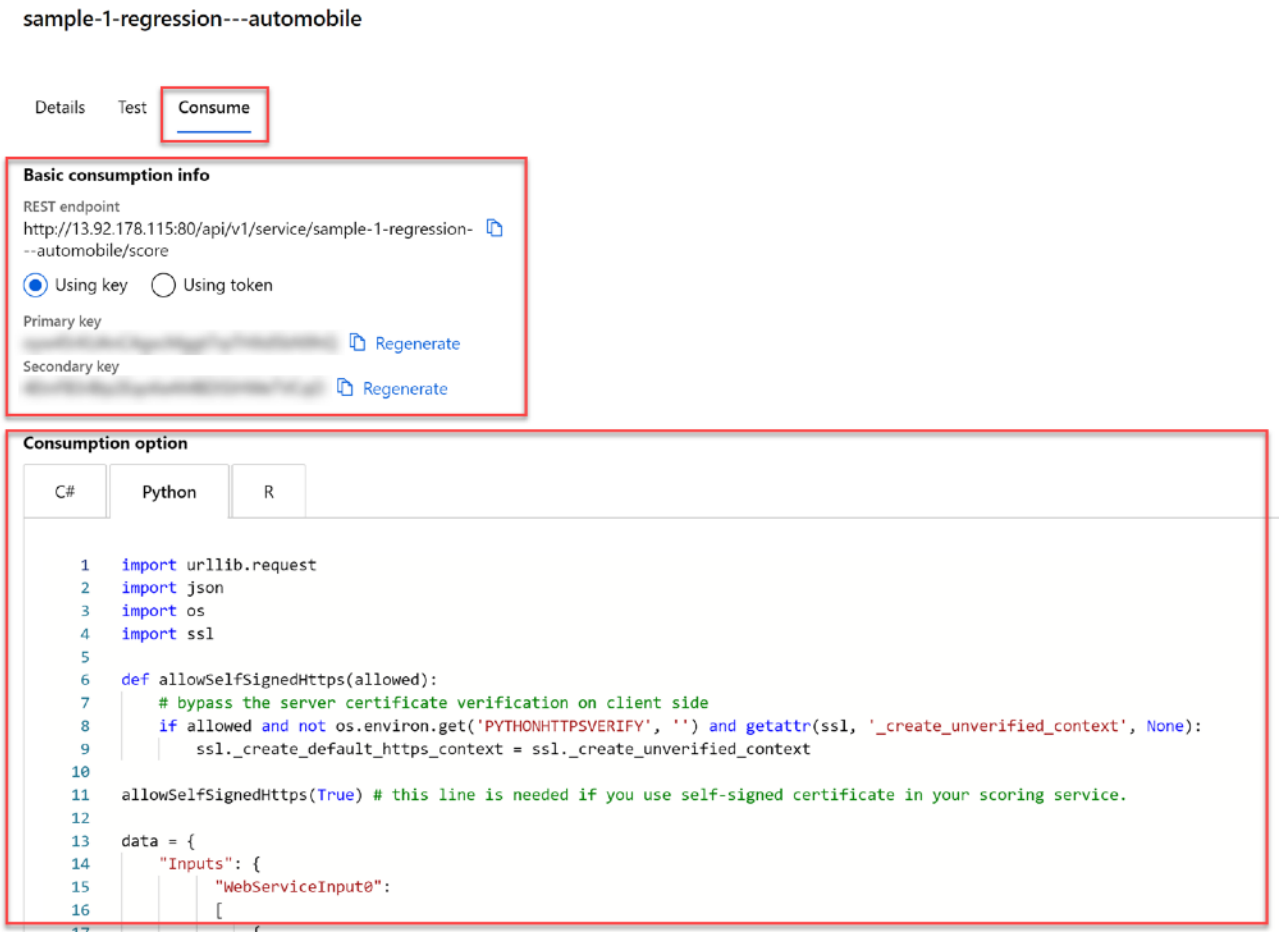
1. To view the deployed web service, select the **Endpoints** section in your Azure Portal Workspace.
2. Select the deployed web service: **sample-1-regression—-automobile** to open the deployment details page.



Note: you have to select the text of the service name to open the deployment details page

Task 3: Review how to consume the deployed web service

1. Select the **Consume** tab to observe the following information:
    1. `Basic consumption info` displays the **REST endpoint**, **Primary key**, and **Secondary key**.
    2. `Consumption option` shows code samples in **C#**, **Python**, and **R** on how to call the endpoint to consume the webservice.



Next Steps

Congratulations! You have just learned how to train and deploy a model to an Azure Kubernetes Service (AKS) cluster for real-time inferencing. You can now return to the Udacity portal to continue with the lesson.

Lab 22 - Training and deploying a model from a notebook running in a Compute Instance

So far, the Managed Services for Azure Machine Learning lesson has covered **compute instance** and the benefits it provides through its fully managed environment containing everything you need to run Azure Machine Learning.
The compute instance provides a comprehensive set of a capabilities that you can use directly within a python notebook or python code including:

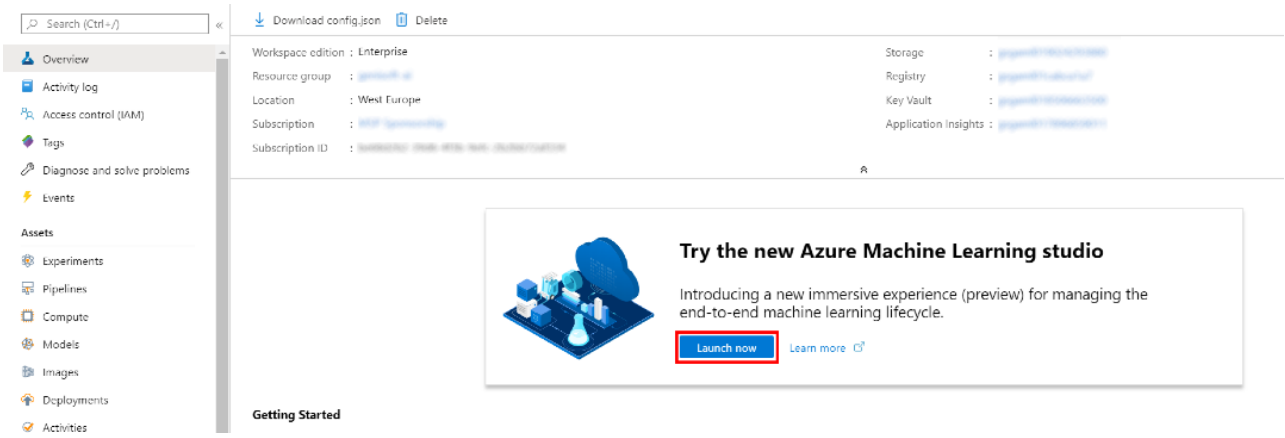
- Creating a **Workspace** that acts as the root object to organize all artifacts and resources used by Azure Machine Learning.
- Creating **Experiments** in your Workspace that capture versions of the trained model along with any desired model performance telemetry. Each time you train a model and evaluate its results, you can capture that run (model and telemetry) within an Experiment.
- Creating **Compute** resources that can be used to scale out model training, so that while your notebook may be running in a lightweight container in Azure Notebooks, your model training can actually occur on a powerful cluster that can provide large amounts of memory, CPU or GPU.
- Using **Automated Machine Learning (AutoML)** to automatically train multiple versions of a model using a mix of different ways to prepare the data and different algorithms and hyperparameters (algorithm settings) in search of the model that performs best according to a performance metric that you specify.
- Packaging a Docker **Image** that contains everything your trained model needs for scoring (prediction) in order to run as a web service.
- Deploying your Image to either Azure Kubernetes or Azure Container Instances, effectively hosting the **Web Service**.

Overview

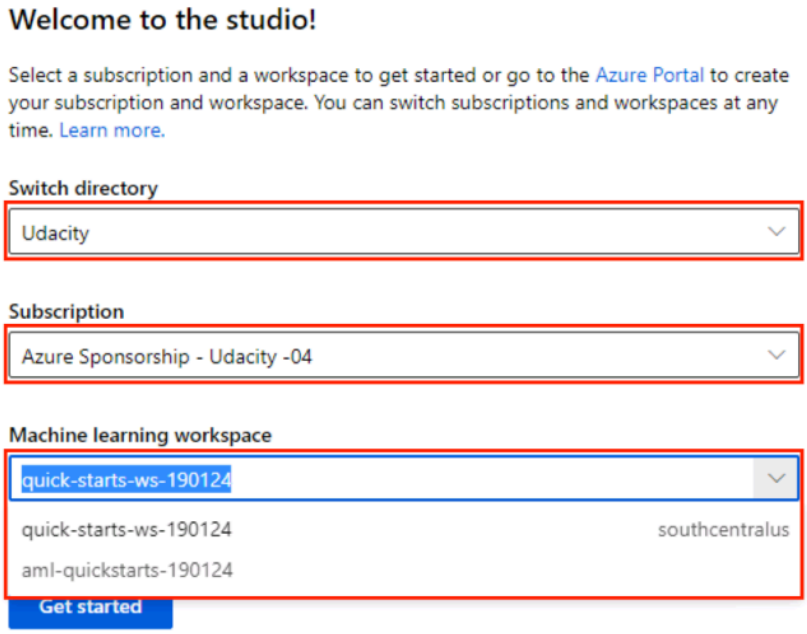
In this lab, you start with a model that was trained using Automated Machine Learning. Learn how to use the Azure ML Python SDK to register, package, and deploy the trained model to Azure Container Instances (ACI) as a scoring web service. Finally, test the deployed model (1) by make direct calls on service object, (2) by calling the service end point (Scoring URI) over http.

Exercise 1: Run the Notebook for this Lab

1. In Azure portal, open the available machine learning workspace.
2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

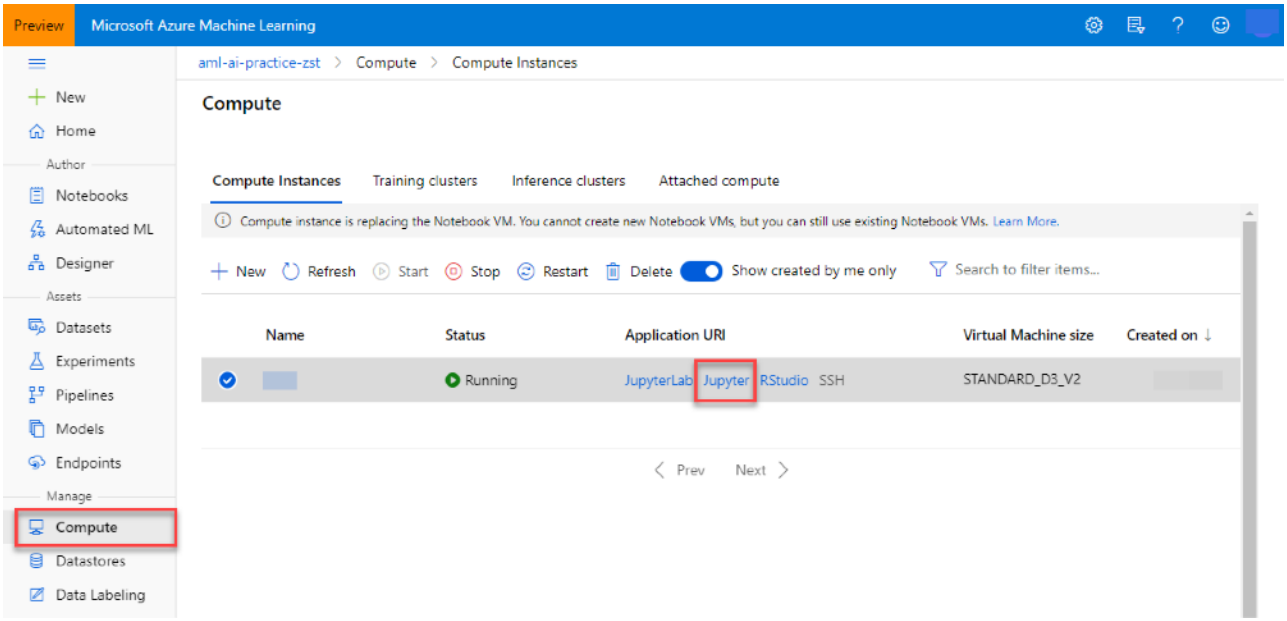


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:

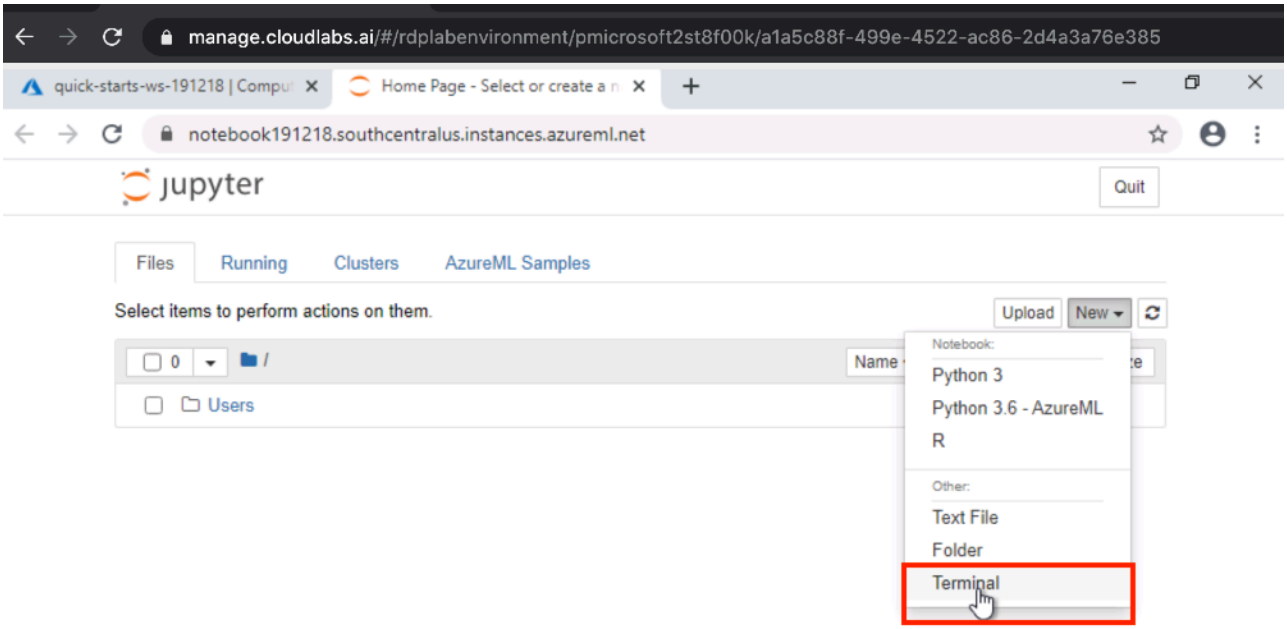


For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

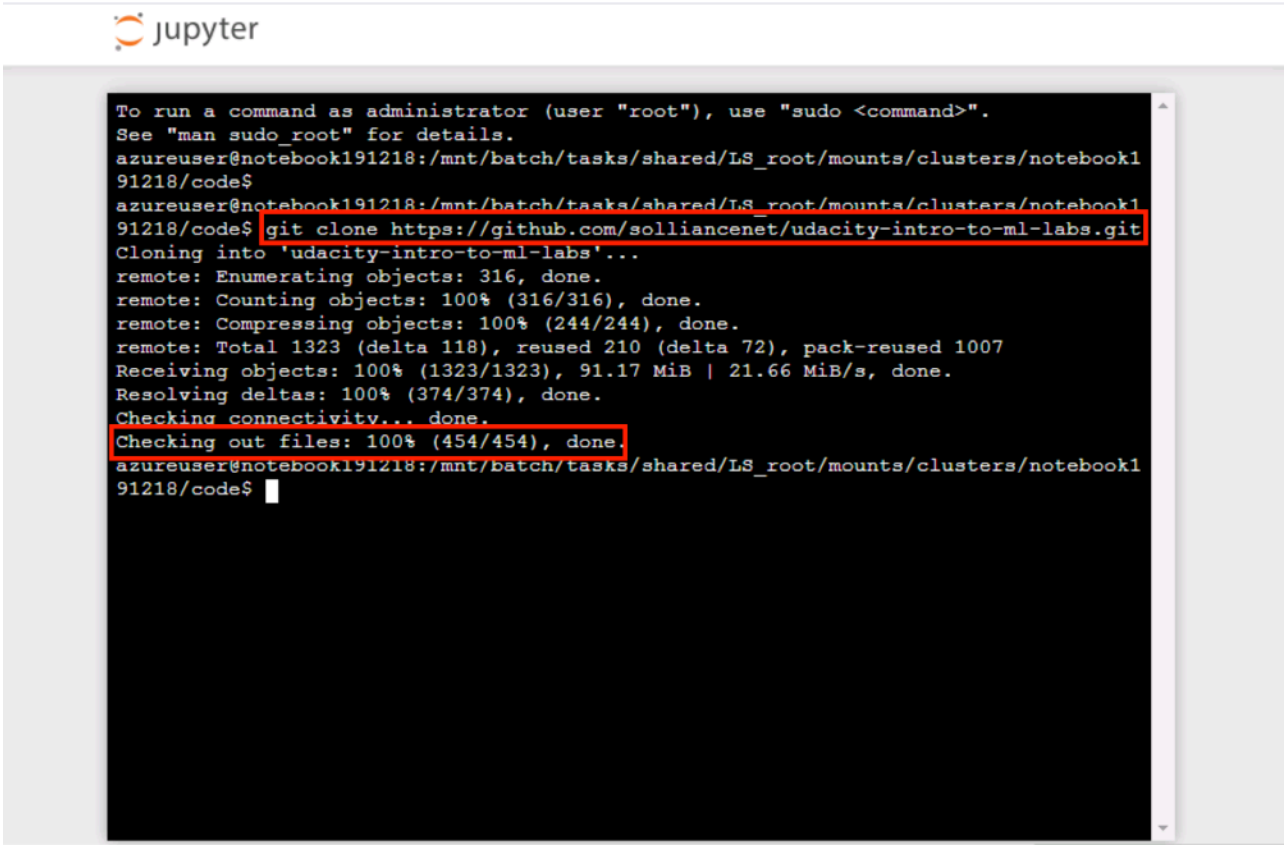
4. From the studio, navigate to **Compute**. Next, for the available Compute Instance, under Application URI select `Jupyter`. Be sure to select `Jupyter` and not `JupterLab`.



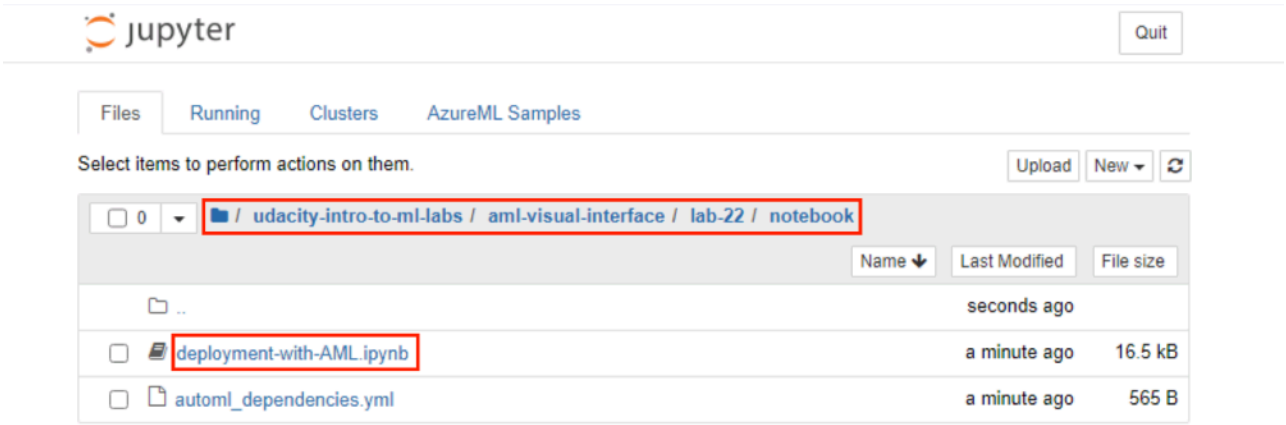
5. From within the Jupyter interface, select **New, Terminal**.



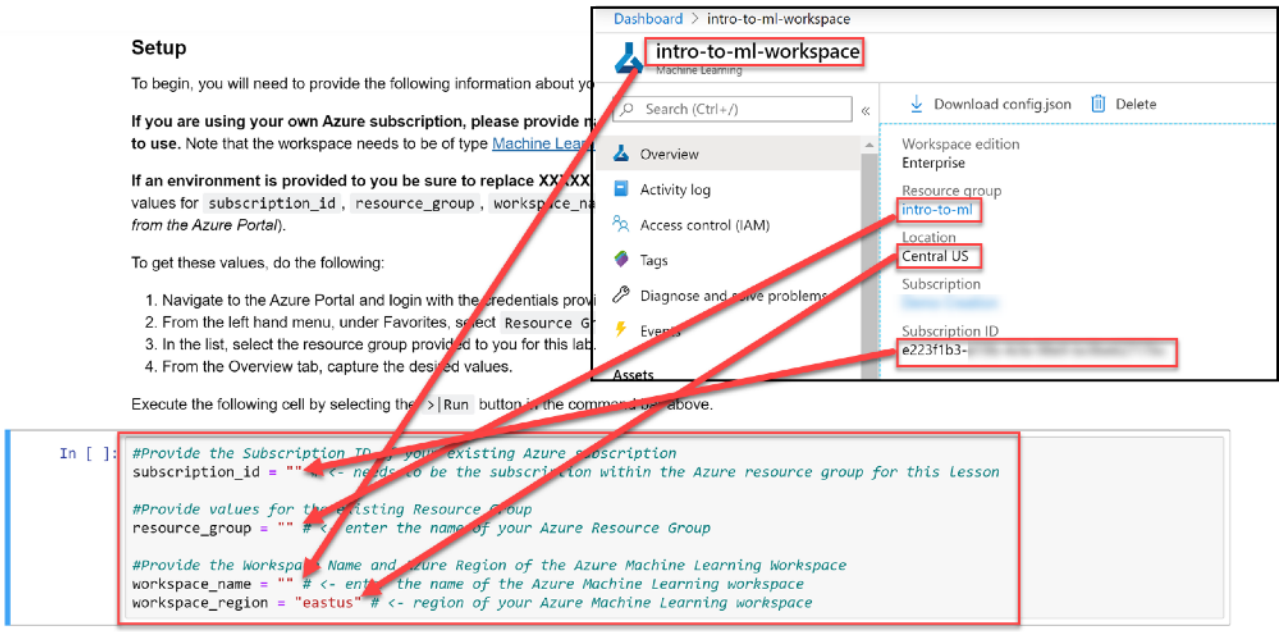
6. In the new terminal window run the following command and wait for it to finish: `git clone https://github.com/solliancenet/udacity-intro-to-ml-labs.git`



7. From within the Jupyter interface, navigate to directory `udacity-intro-to-ml-labs/aml-visual-interface/lab-22/notebook` and open `deployment-with-AML.ipynb`. This is the Python notebook you will step through executing in this lab.



8. In the Setup portion of the notebook, you will be asked to provide values for `subscription_id`, `resource_group`, `workspace_name`, and `workspace_region`. To find these, open your Azure Machine Learning workspace in the Azure portal and copy the values as shown:



9. Follow the instructions within the notebook to complete the lab.

Next Steps

Congratulations! You have just learned how to use the Jupyter application on a compute instance to deploy a trained model to Azure Container Instances (ACI) for real-time inferencing. You can now return to the Udacity portal to continue with the lesson.

Lab 23 - Explaining models: Model interpretability with Azure Machine Learning service

Machine learning interpretability is important in two phases of machine learning development cycle:

- During training: Model designers and evaluators require interpretability tools to explain the output of a model to stakeholders to build trust. They also need insights into the model so that they can debug the model and make decisions on whether the behavior matches their objectives. Finally, they need to ensure that the model is not biased.
- During inferencing: Predictions need to be explainable to the people who use your model. For example, why did the model deny a mortgage loan, or predict that an investment portfolio carries a higher risk?

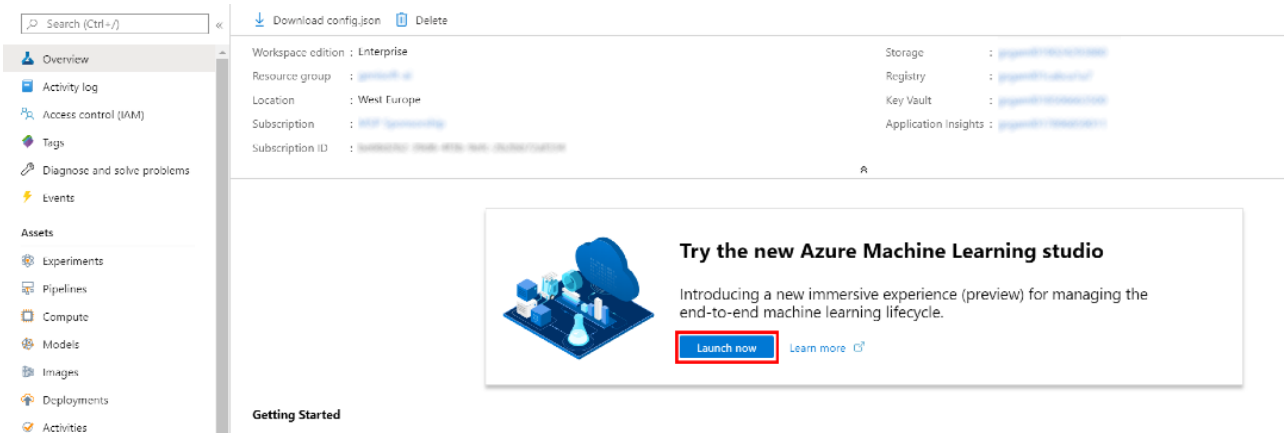
The Azure Machine Learning Interpretability Python SDK incorporates technologies developed by Microsoft and proven third-party libraries (for example, SHAP and LIME). The SDK creates a common API across the integrated libraries and integrates Azure Machine Learning services. Using this SDK, you can explain machine learning models globally on all data, or locally on a specific data point using the state-of-art technologies in an easy-to-use and scalable fashion.

Overview

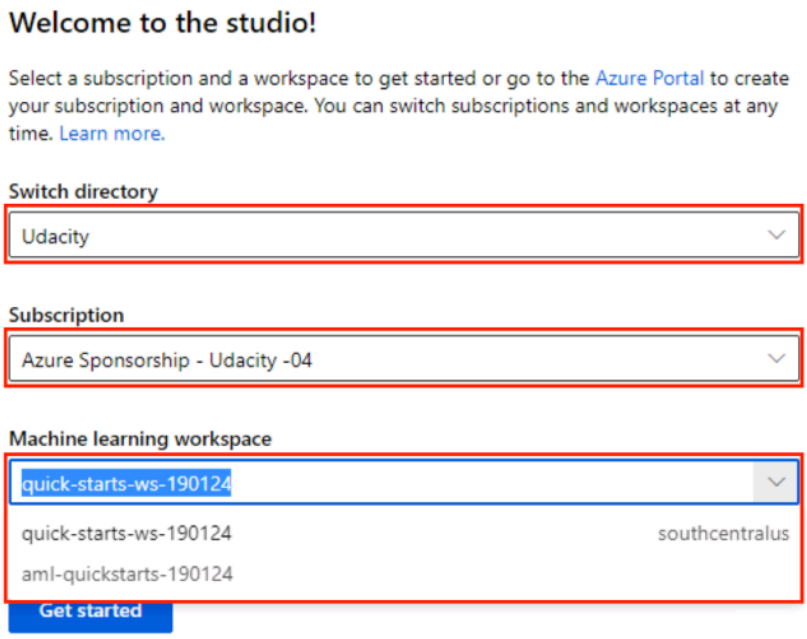
In this lab, we will be using a subset of NYC Taxi & Limousine Commission - green taxi trip records available from Azure Open Datasets. The data is enriched with holiday and weather data. We will use data transformations and the GradientBoostingRegressor algorithm from the scikit-learn library to train a regression model to predict taxi fares in New York City based on input features such as, number of passengers, trip distance, datetime, holiday information and weather information.
The primary goal of this quickstart is to explain the predictions made by our trained model with the various Azure Model Interpretability packages of the Azure Machine Learning Python SDK.

Exercise 1: Run the Notebook for this Lab

1. In Azure portal, open the available machine learning workspace.
2. Select **Launch now** under the **Try the new Azure Machine Learning studio** message.

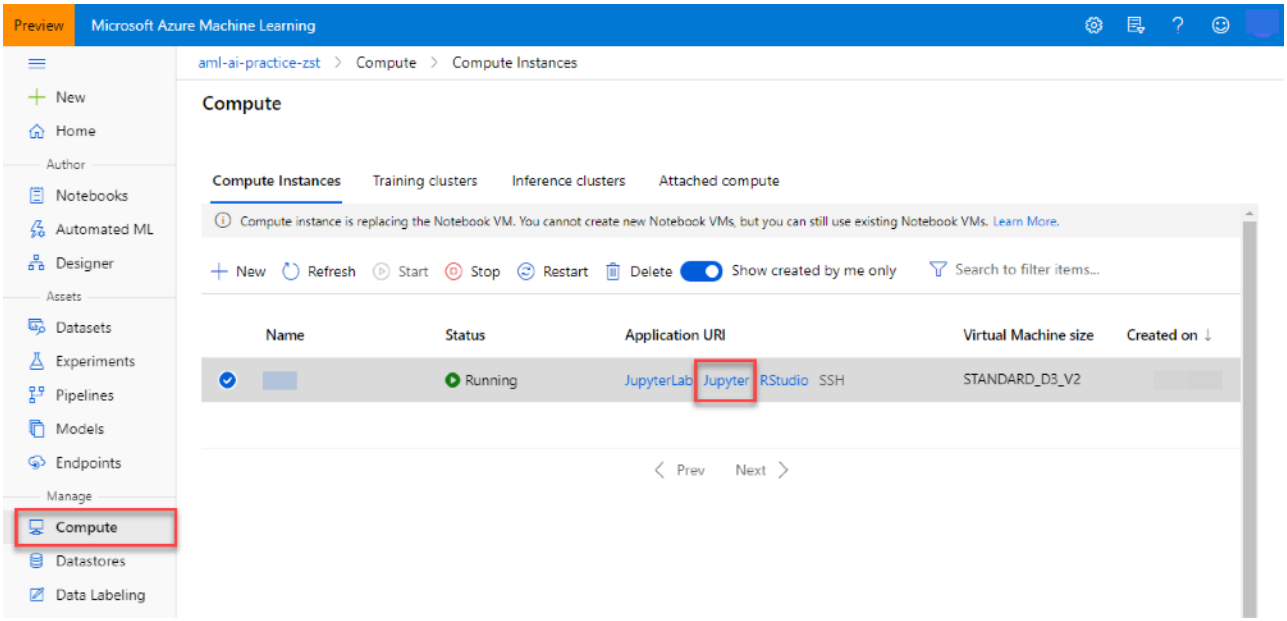


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:

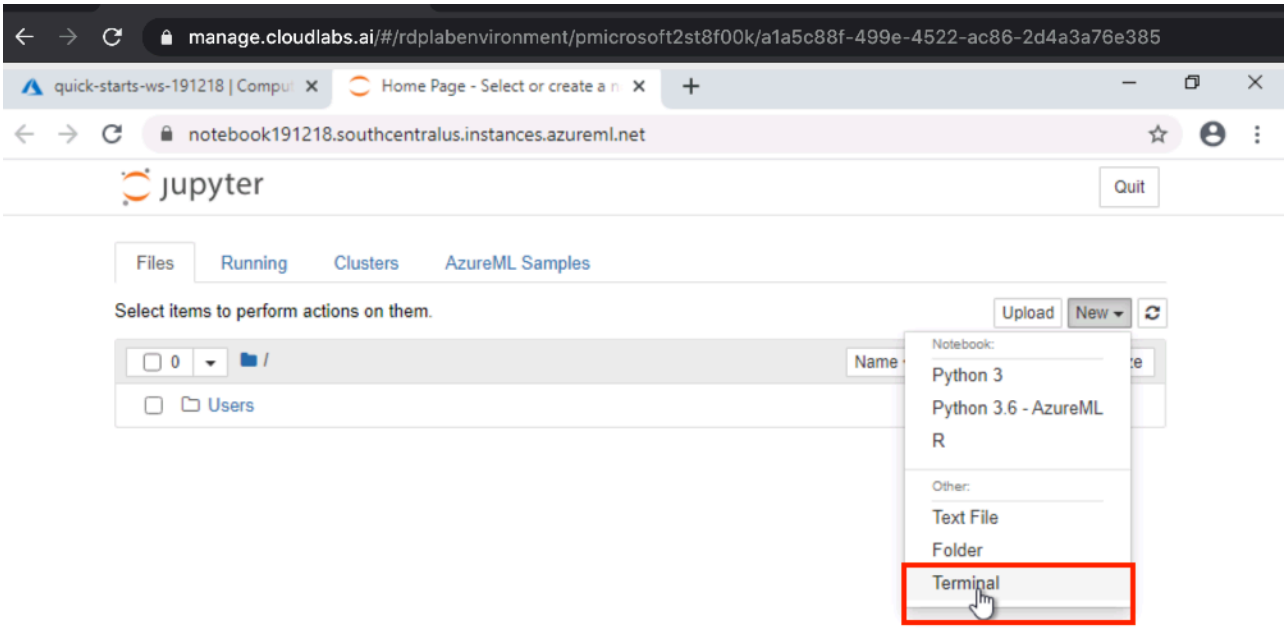


For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

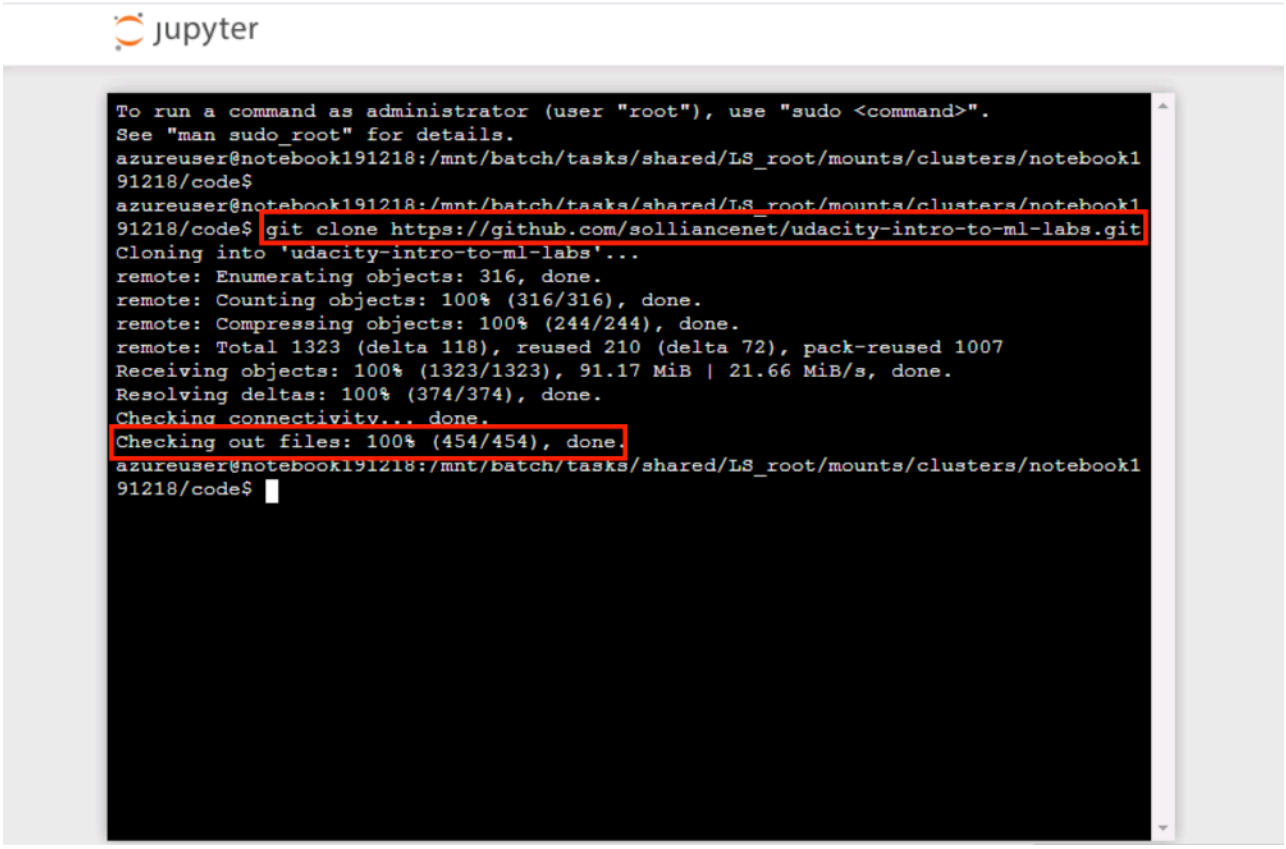
4. From the studio, navigate to **Compute**. Next, for the available Compute Instance, under Application URI select `Jupyter`. Be sure to select `Jupyter` and not `JupterLab`.



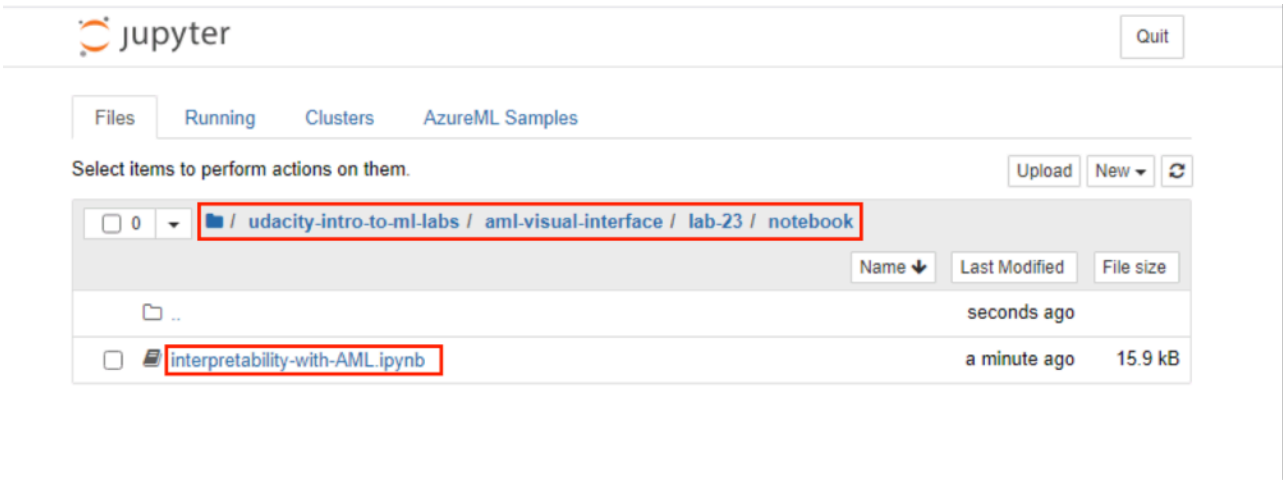
5. From within the Jupyter interface, select **New, Terminal**.



6. In the new terminal window run the following command and wait for it to finish: `git clone https://github.com/solliancenet/udacity-intro-to-ml-labs.git`



7. From within the Jupyter interface, navigate to directory `udacity-intro-to-ml-labs/aml-visual-interface/lab-23/notebook` and open `interpretability-with-AML.ipynb`. This is the Python notebook you will step through executing in this lab.



8. Follow the instructions within the notebook to complete the lab.

Next Steps

Congratulations! You have just learned how to use the Azure Machine Learning SDK to help you explain what influences the predictions a model makes. You can now return to the Udacity portal to continue with the lesson.