Table of Contents

Lesson 3: Model Training

Chapter 1: Lesson Overview

Before training a model, we first need to handle **data preparation**, so we will explore this topic first. More specifically, we will go over:

- Data importing and transformation
- The data management process, including:
- The use of *datastores* and *datasets*
- Versioning
- Feature engineering
- How to monitor for *data drift*

Next, we will introduce the basics of **model training**. We will cover:

- The core model training process
- Two of the fundamental machine learning models: *Classifier* and *regressor*
- The model evaluation process and relevant metrics

And finally, we'll conclude with an introduction to **ensemble learning** and **automated machine learning**, two core techniques used to make decisions based on multiple—rather than single—trained models.

Through Model training, we transform data into trained ML Models. We will start with Data Management, Data preparation and Data handling.

Proper data is the most important ingredient for successful ML Models. Feed proper, accurate, clean and high quality data. Then we will focus into Training process.

Usage of Multiple ML Models to improve the performance of predictions. This is, called ensemble learning

Chapter 2: Prelaunch Lab

Prelaunch your lab environment.

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

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

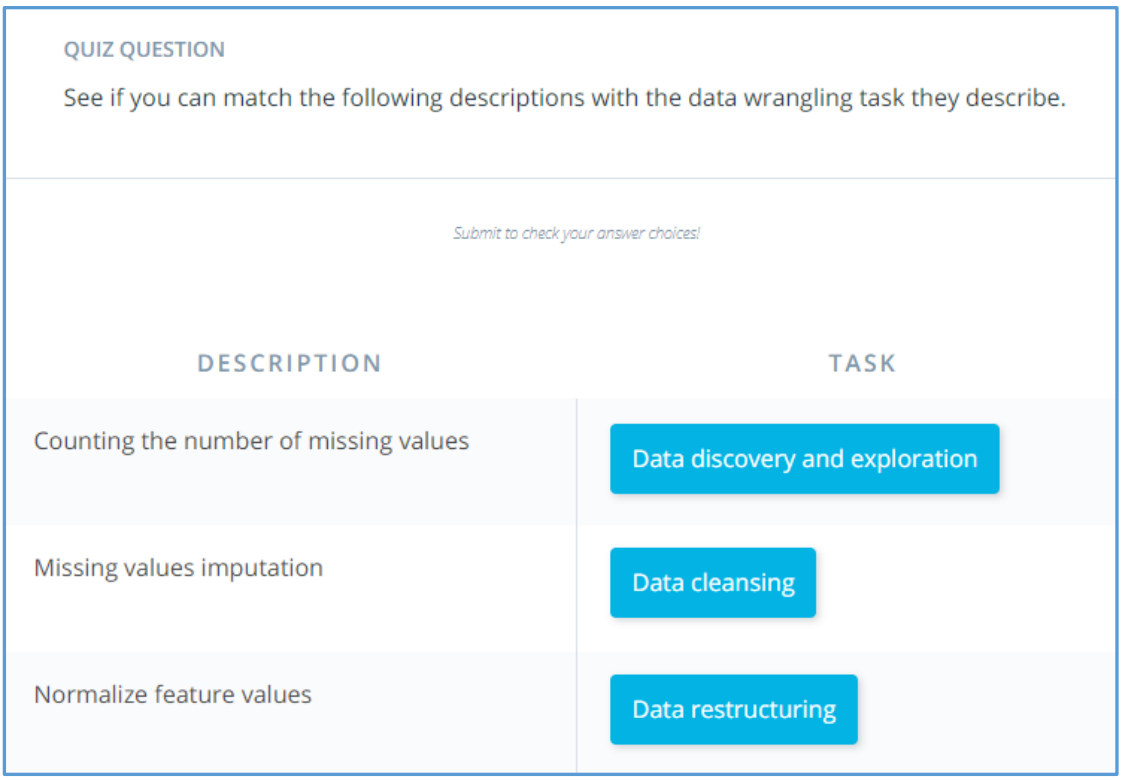
Chapter 3: Data Import and Transformation

Data wrangling is the process of cleaning and transforming data to make it more appropriate for data analysis. The process generally follows these main steps:

- **Data discovery & Exploration -** Explore the raw data and check the general quality of the dataset.
- **Data Transformation -** Transform the raw data, by restructuring, normalizing, and cleaning the data. For example, this could involve handling missing values and detecting errors.
- **Data Publishing -** Validate and publish the data.

Data wrangling is an *iterative* process where you do some data transformation then check the results and come back to the process to make improvements.

Noisy data can lead to undesirable results. Data wrangling is an important part of Data import and transformation process. Through Data wrangling, we eliminate all of these problems. This helps in cleaning, restructuring and potentially enriching the data to transform it into a format, which is much more suitable for the training process.



Chapter 4: Lab – Import Transform and Export Data

Lab Overview

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

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

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

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

While the use of datasets is, recommended to import data, you can also use the Import Data module from the designer. Data comes into the designer from, 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.

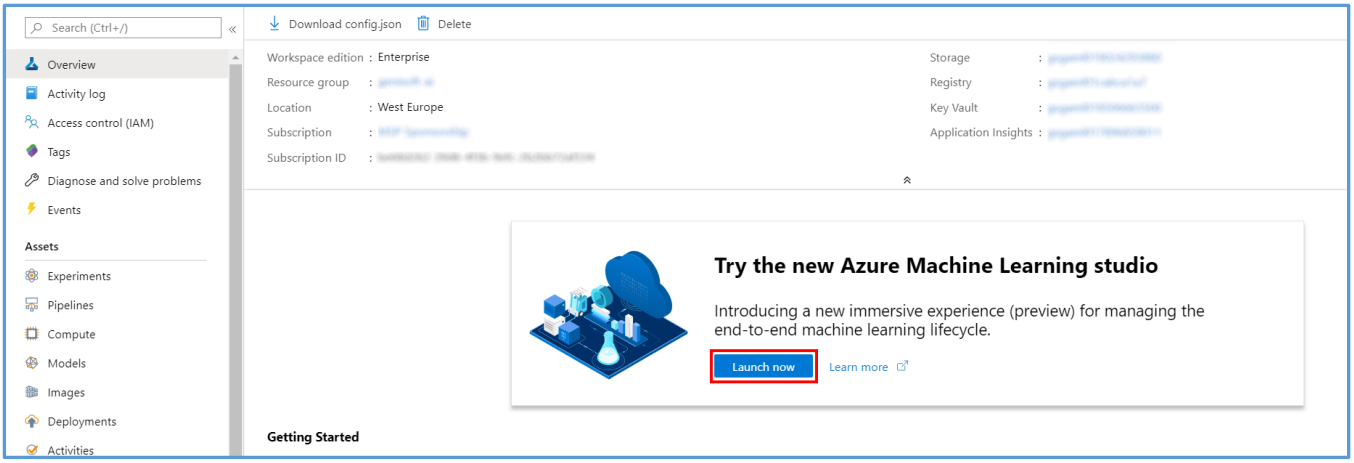
The following exercise focuses on the Import Data module to load data into a machine-learning pipeline from several datasets that will be, merged and restructured. We will be using some sample data from the UCI dataset repository to demonstrate how you can perform basic data import transformation steps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

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

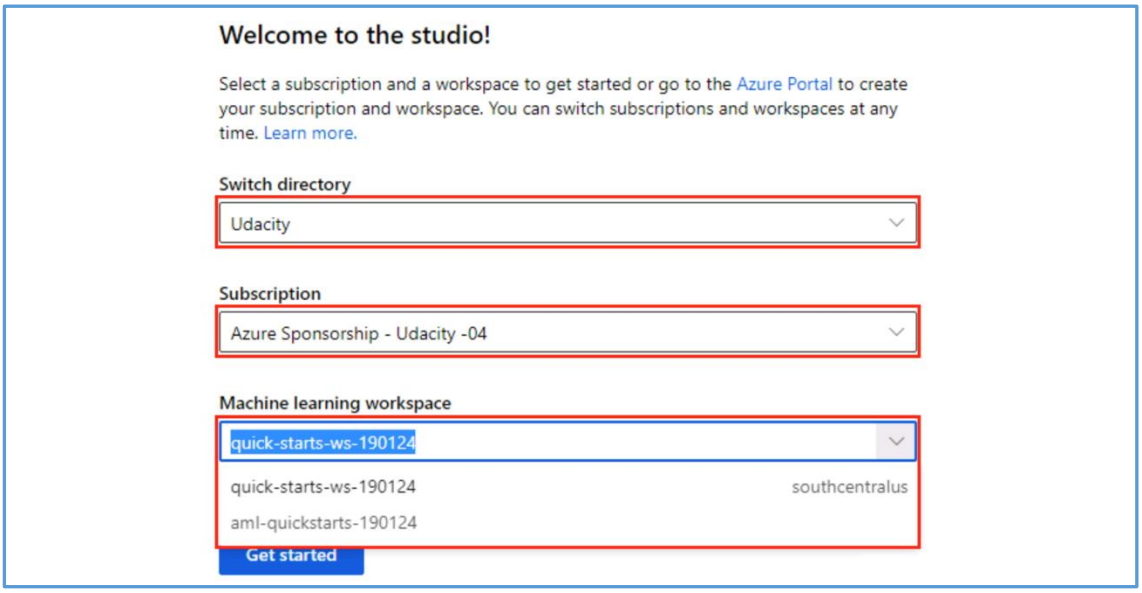
Task 1: Open Pipeline Authoring Editor

In Azure portal, open the available machine learning workspace.

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

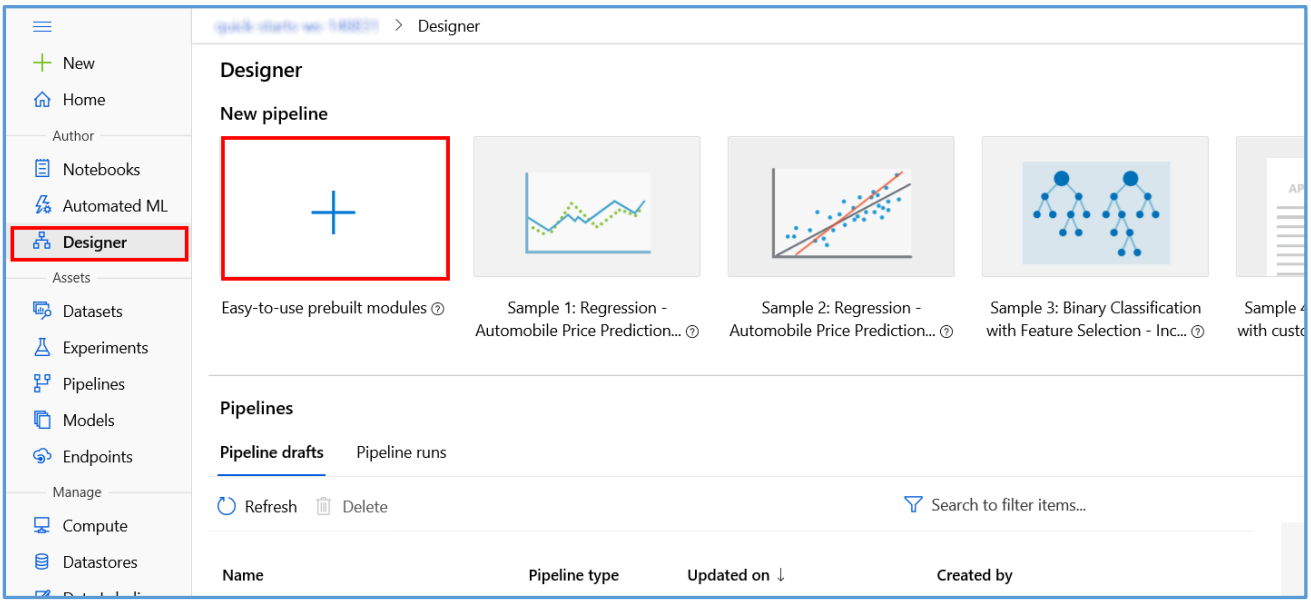


When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



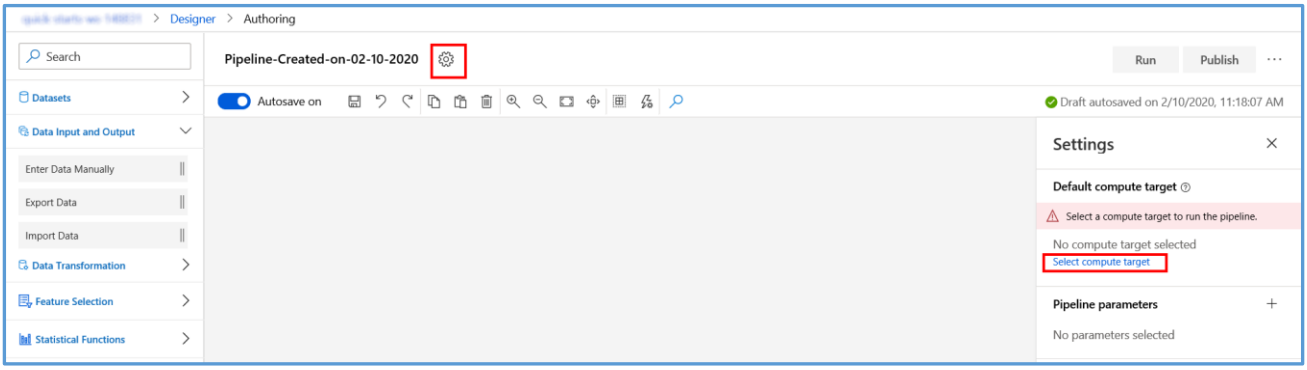
For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it does not matter which) and then click **Get started**.

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



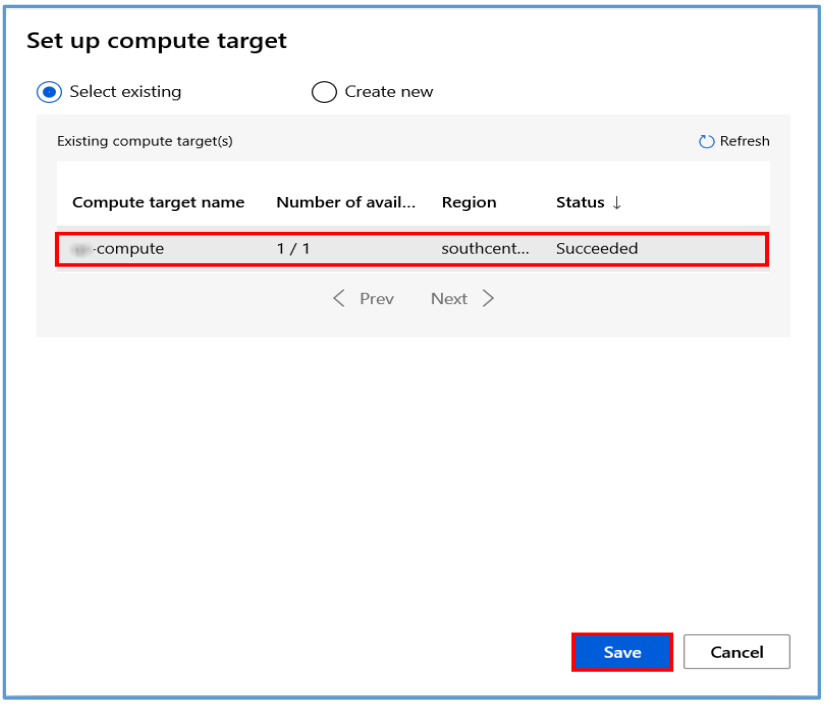
Task 2: Setup Compute Target

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



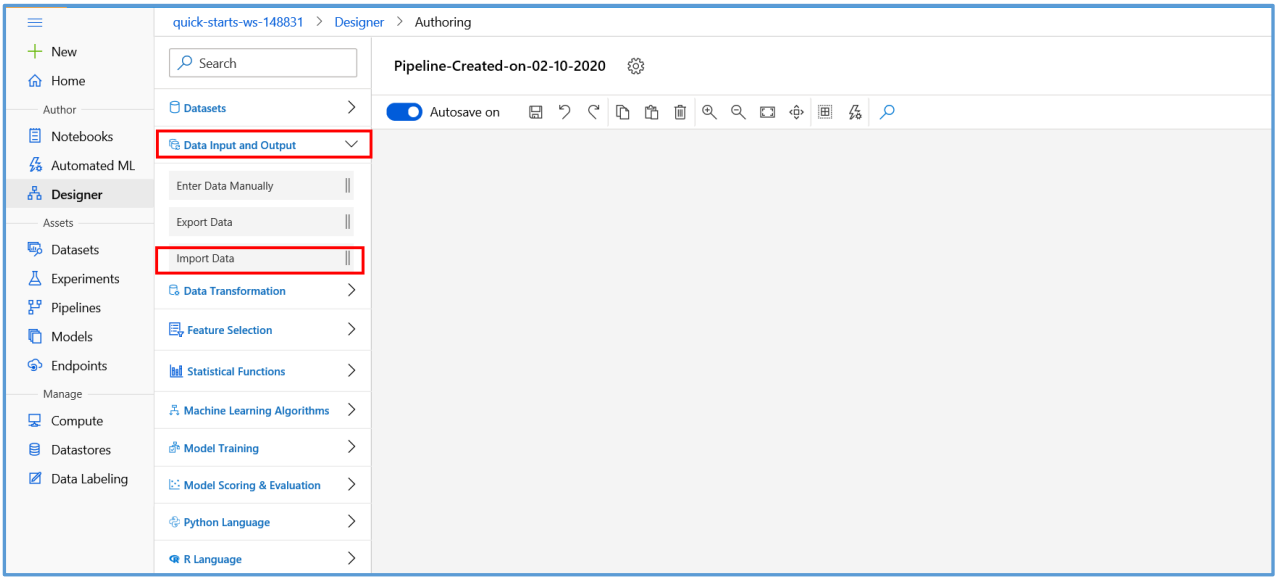
In the Set up compute target editor, select the existing compute target, and then select **Save**.

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.

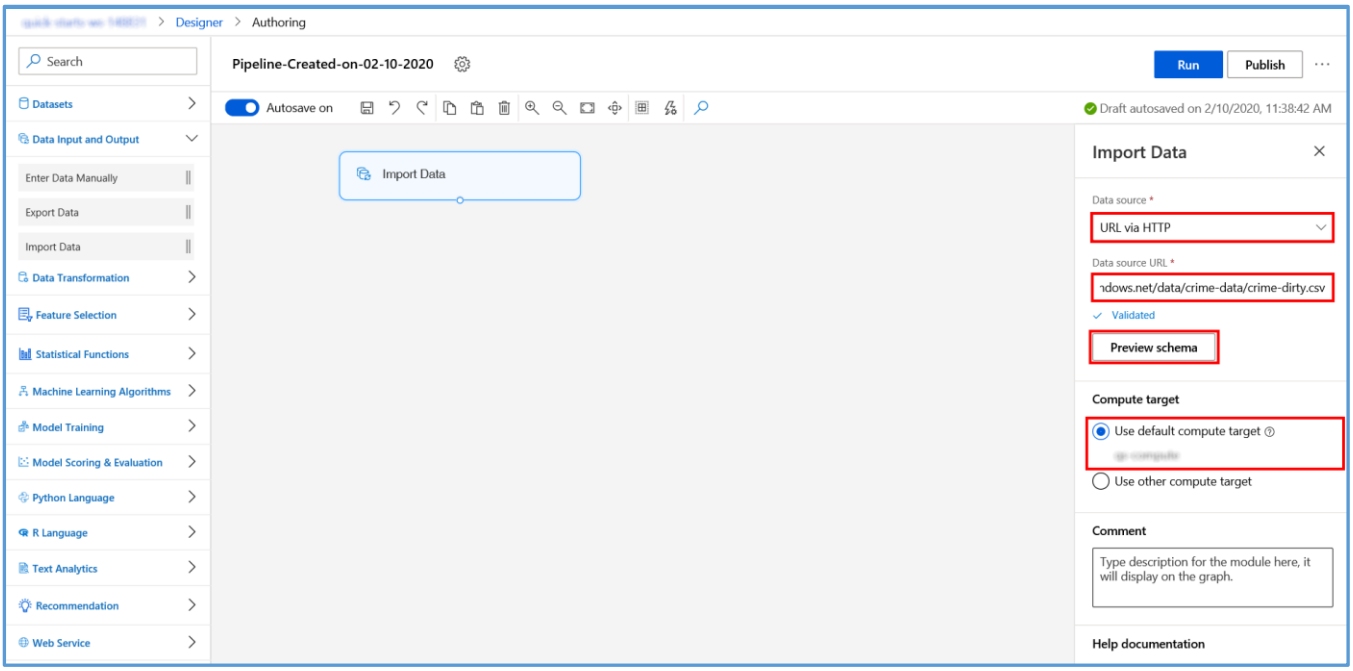


Task 3: Import data from Web URL

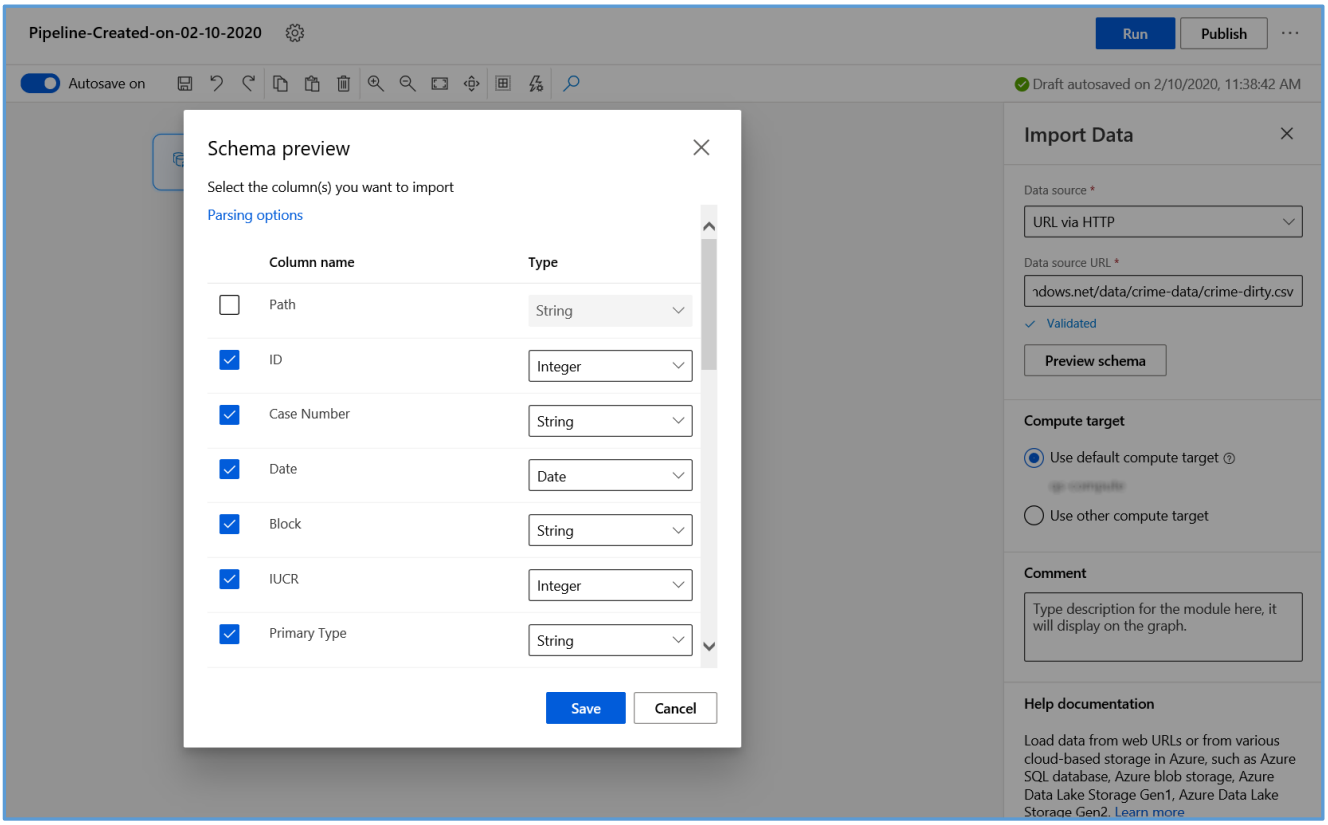
Select **Data Input and Output** section in the left navigation. Next, select **Import Data** and drag and drop the selected module on to the canvas.



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



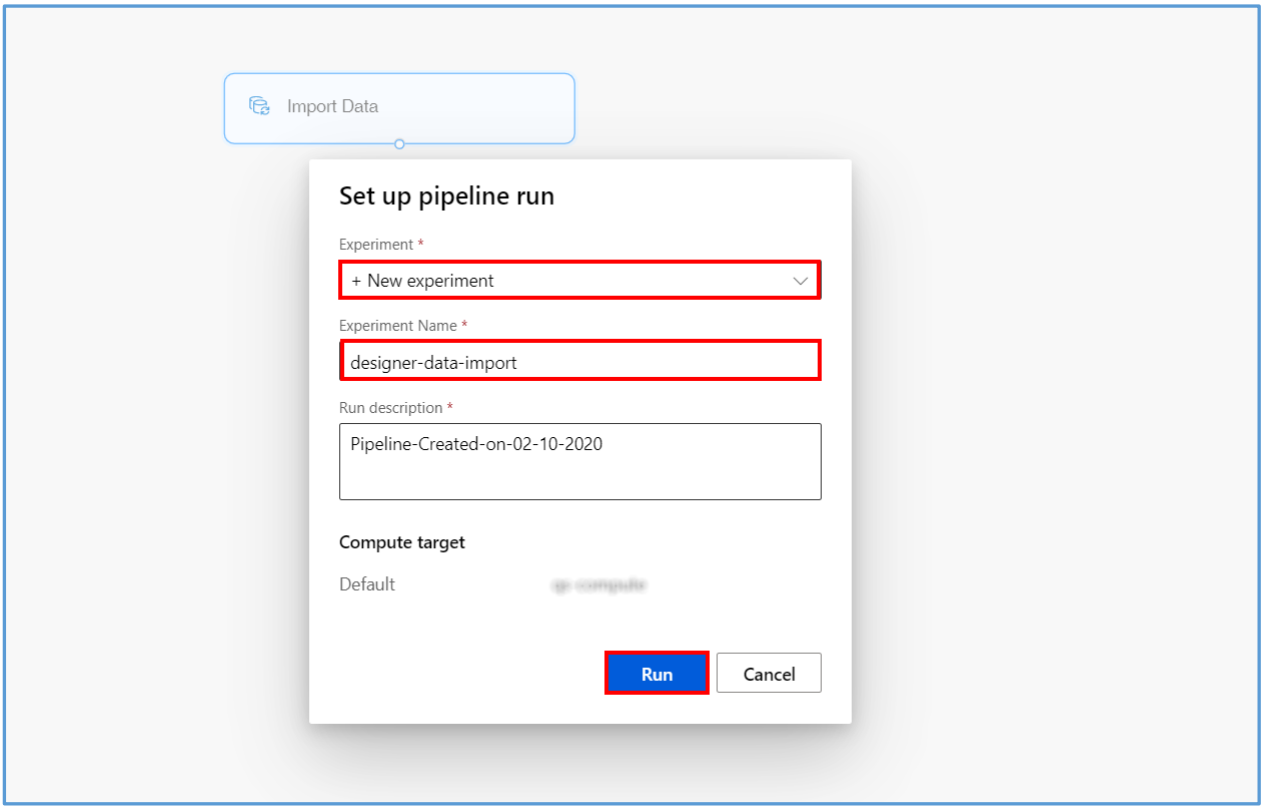
Select the **Preview schema** to filter the columns you want to include. You can also define advanced settings like Delimiter in Parsing options. Select **Save** to close the dialog.



Task 4: Create Experiment and Submit Pipeline

Back to the pipeline canvas; select **Submit** on the top right corner to open the Setup pipeline run editor.

In the Setup pipeline run editor, select **Experiment, Create new** and provide New experiment name: **designer-data-import**, and then select **Submit**.

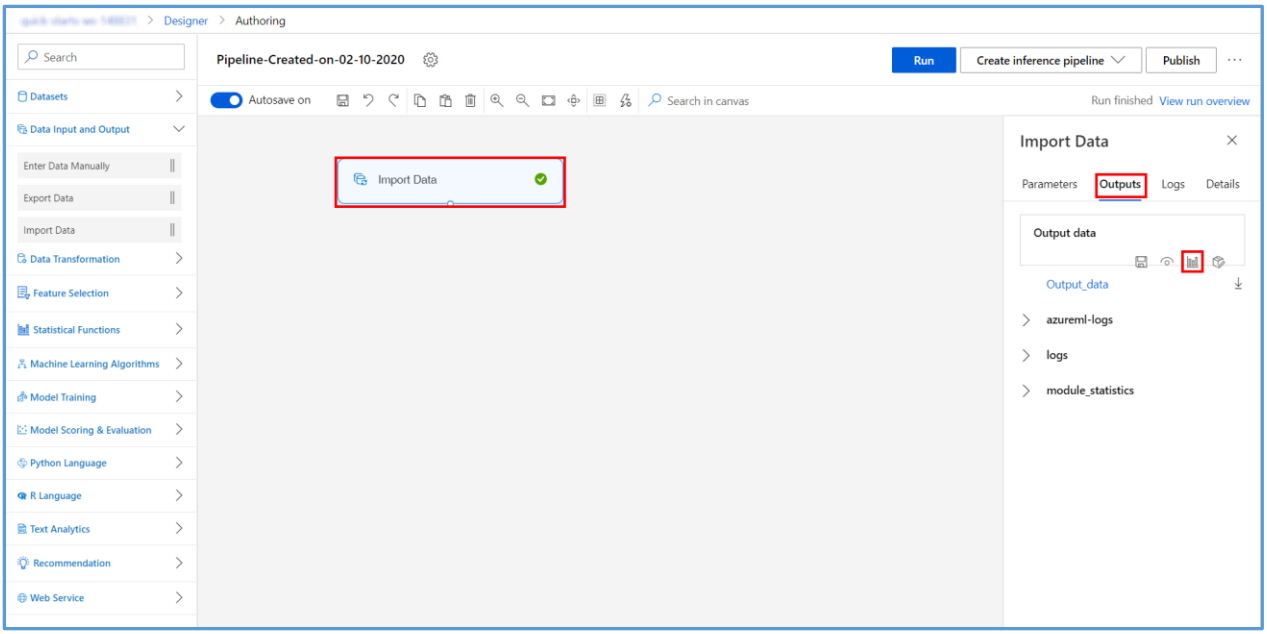


Please note that the button name in the UI is, changed from **Run** to **Submit**.

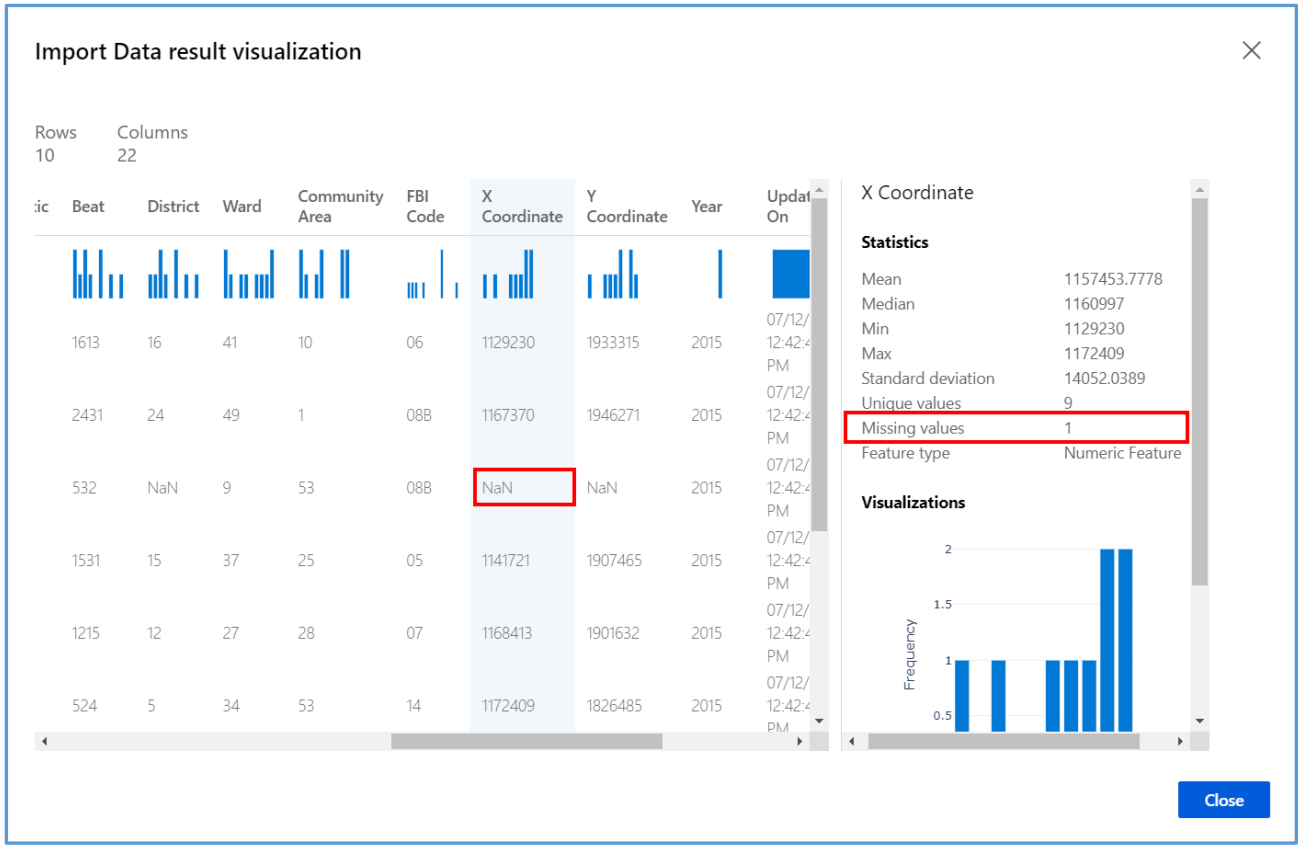
Wait for pipeline run to complete. It will take around **2 minutes** to complete the run.

Task 5: Visualize Import Data results

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.



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.



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

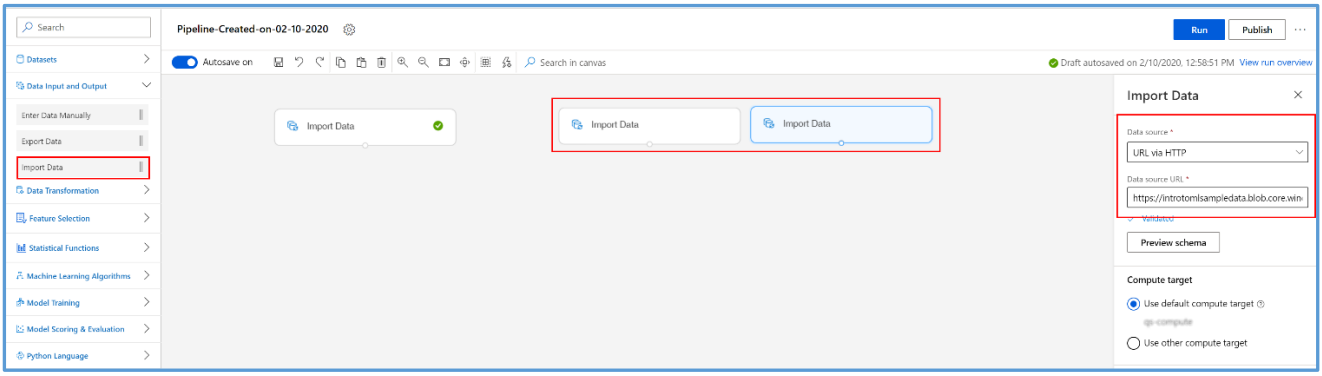
Select **Data Input and Output** section in the left navigation. Next, drag and drop two **Import Data** modules on to the canvas as demonstrated in the first exercise and fill in the Web URLs as follows:

for the first one, **Data source**

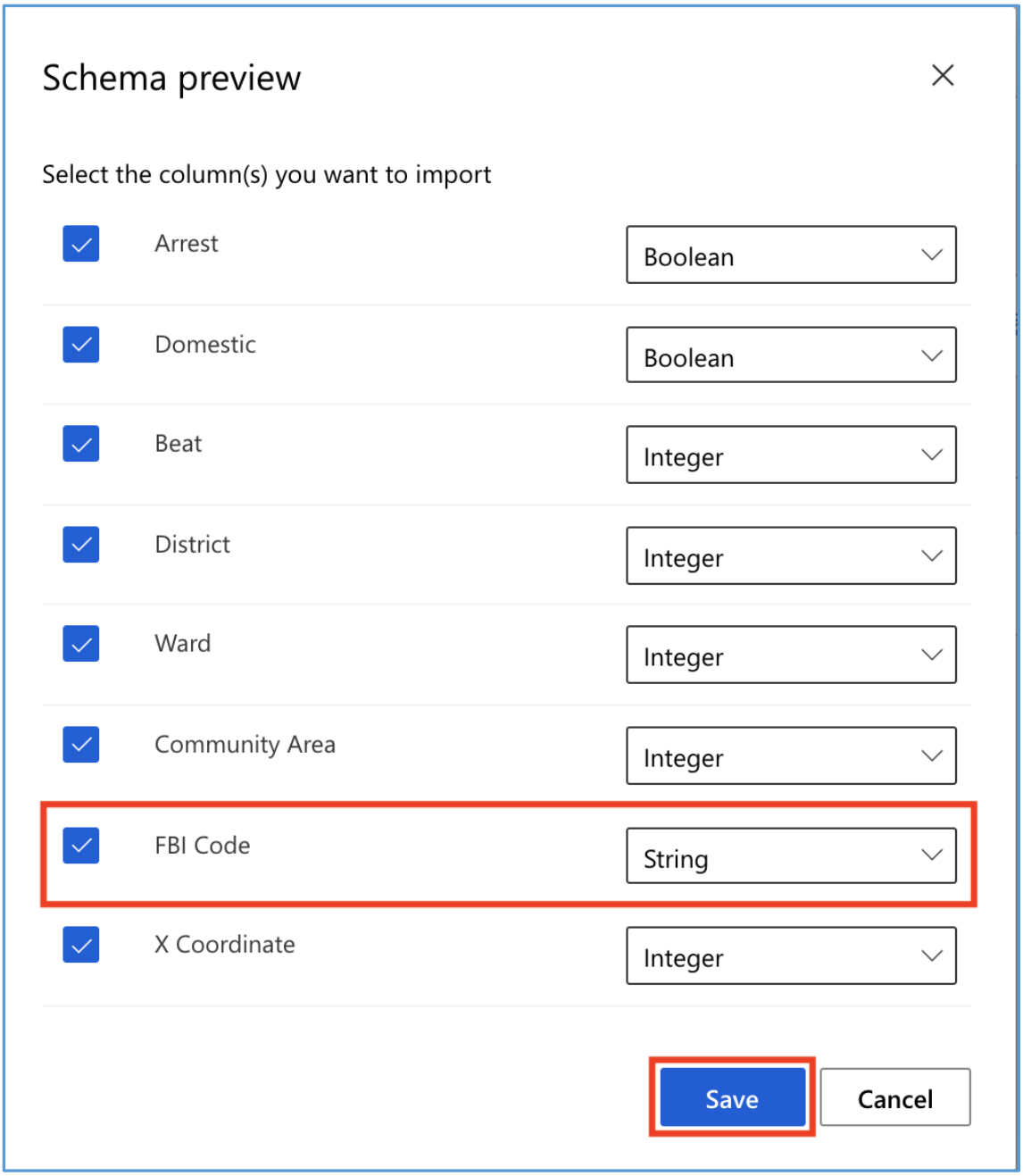
URL : https://introtomlsampledata.blob.core.windows.net/data/crime-data/crime-spring.csv

for the second one, **Data source**

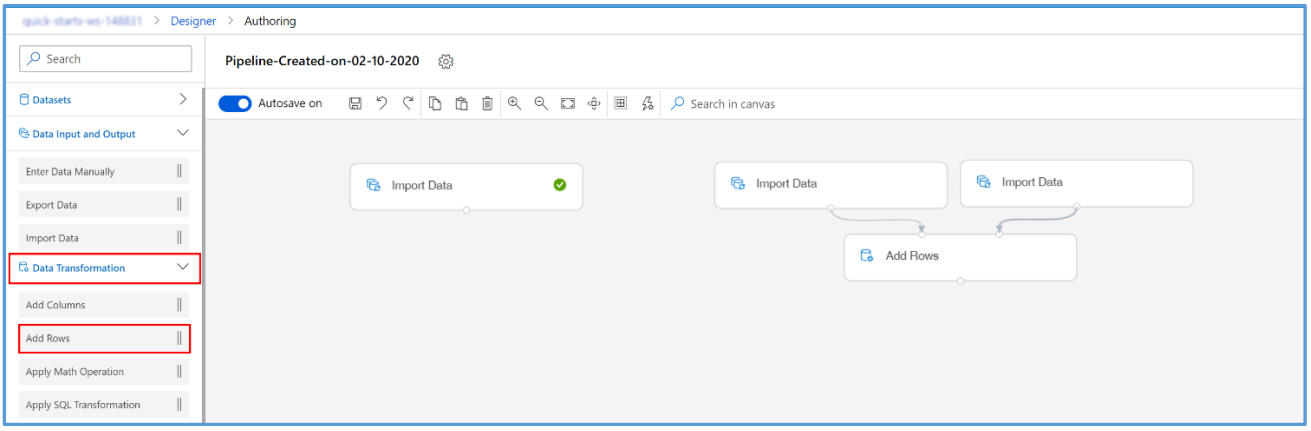
URL : https://introtomlsampledata.blob.core.windows.net/data/crime-data/crime-winter.csv



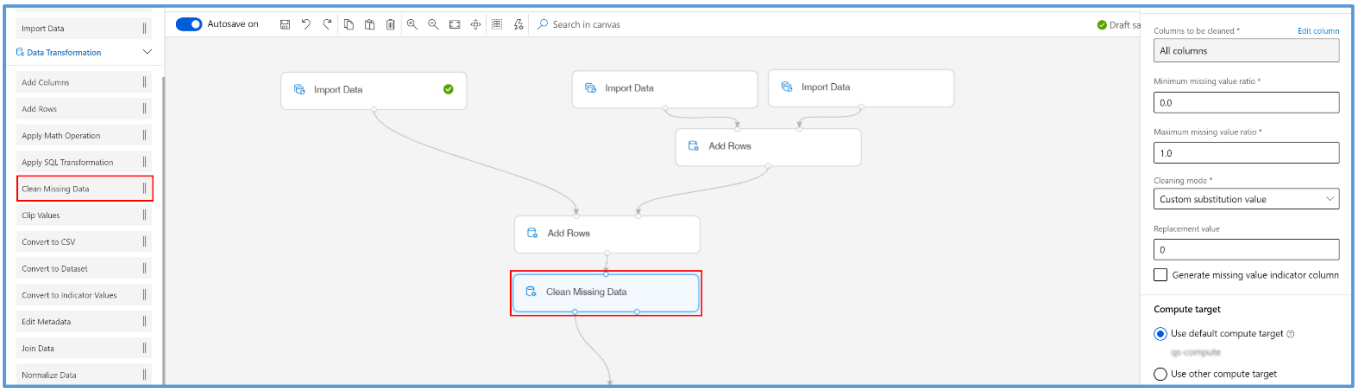
For each of the three Import Data modules, select **Preview schema** and ensure that the data type for FBI Code is of type String and then select **Save**.



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.

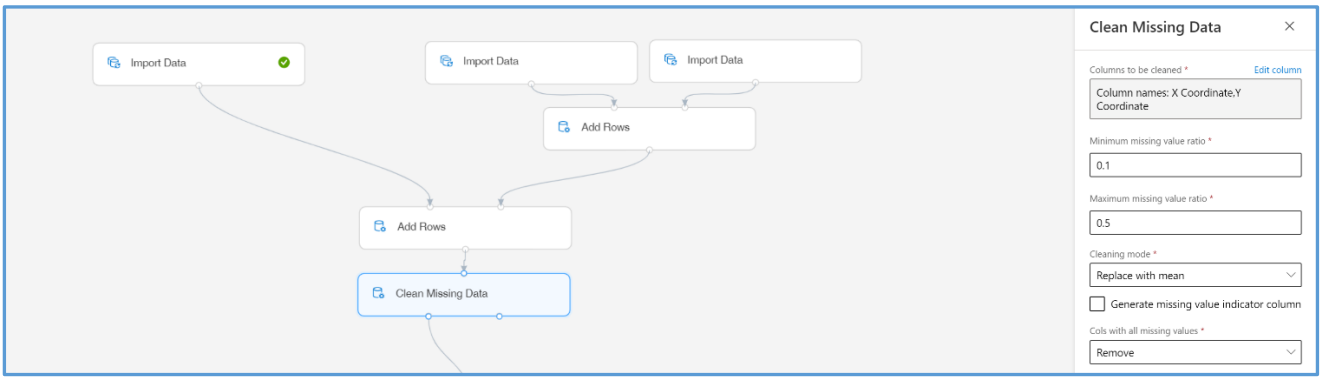


Repeat the same step and add a second **Add rows** module that connects the output from the first **Import data** module to the output of the first **Add rows** module.



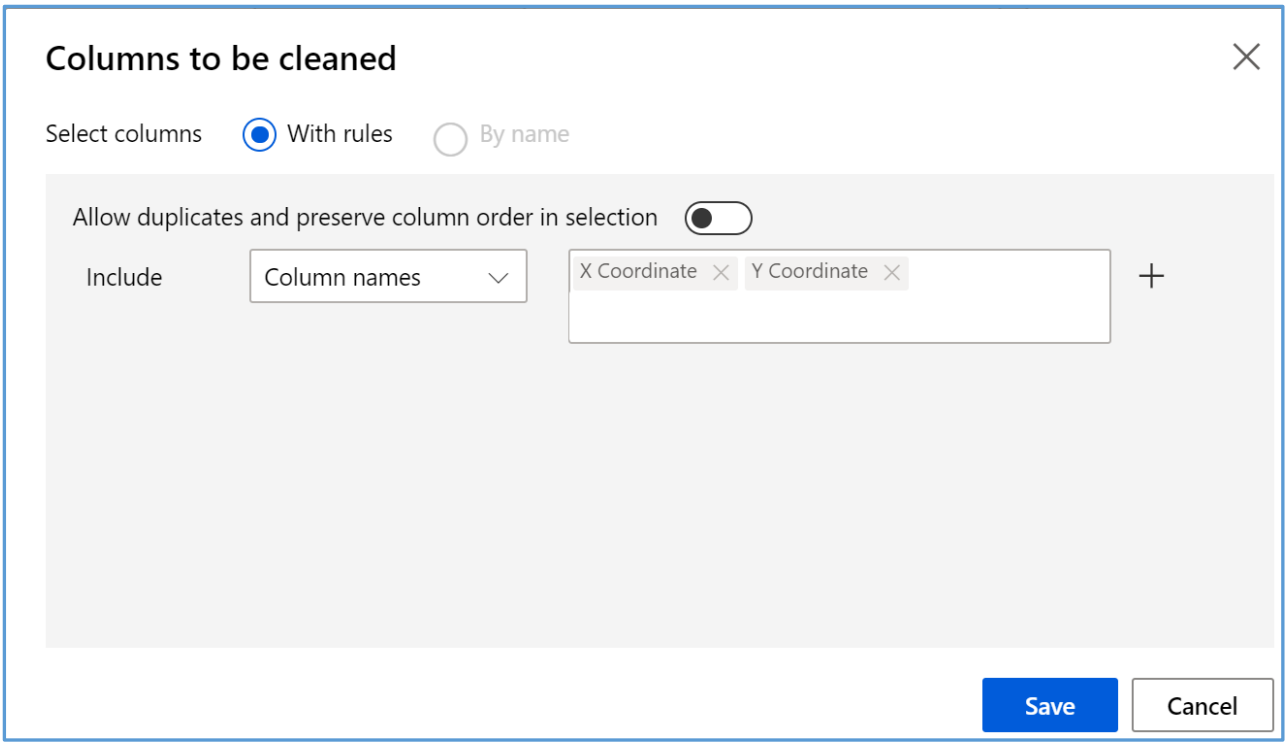
Task 2: Clean missing values

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



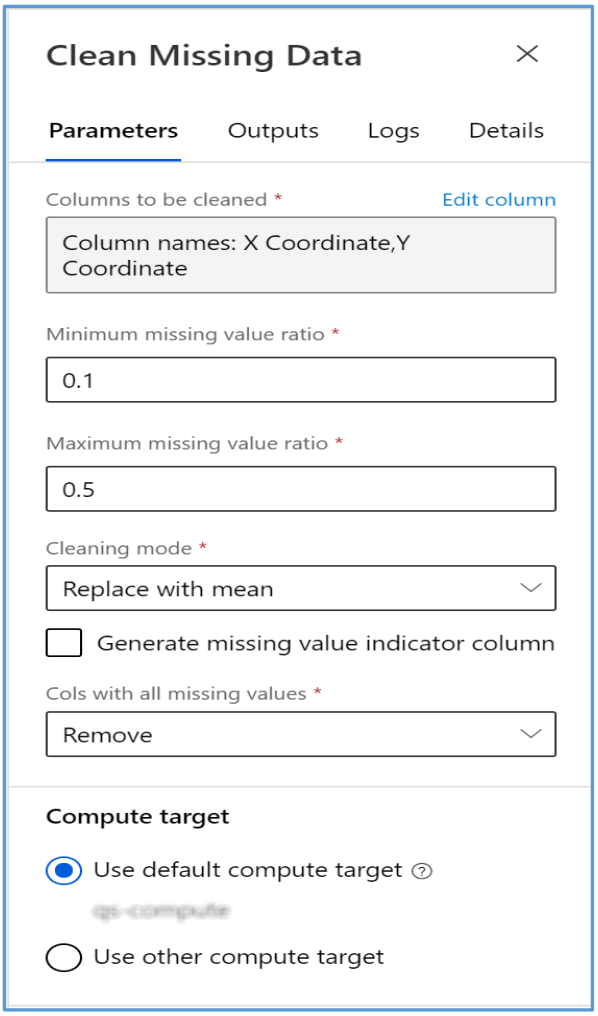
Select **Edit column** in the right pane to configure the list of columns to be, cleaned.

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.



Set the **Minimum missing value ratio** to `0.1` and the **Maximum missing value ratio** to `0.5`.

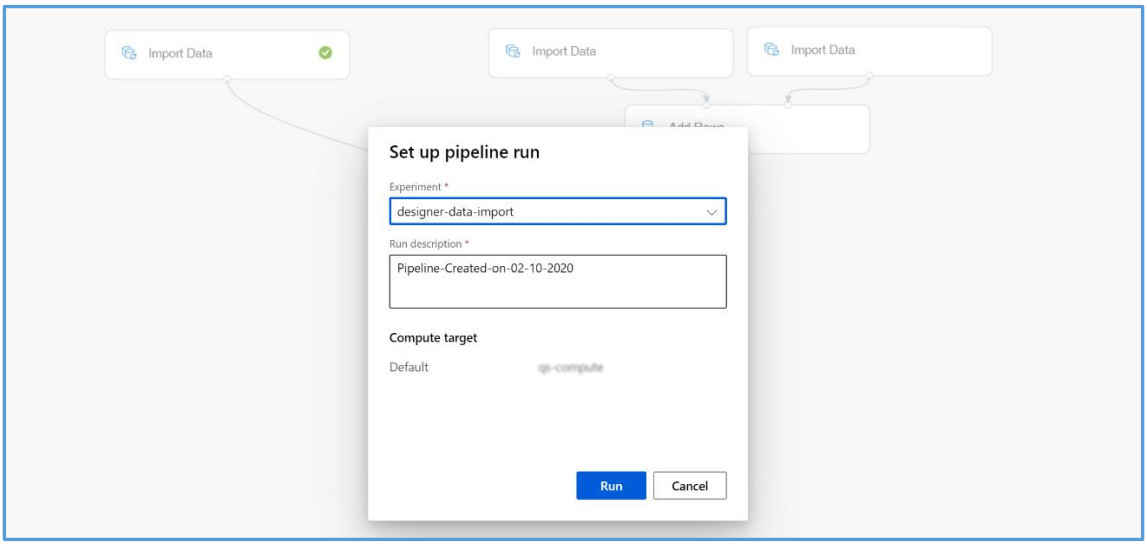
Select `Replace with mean` in the **Cleaning mode** field.



Task 3: Submit Pipeline

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

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



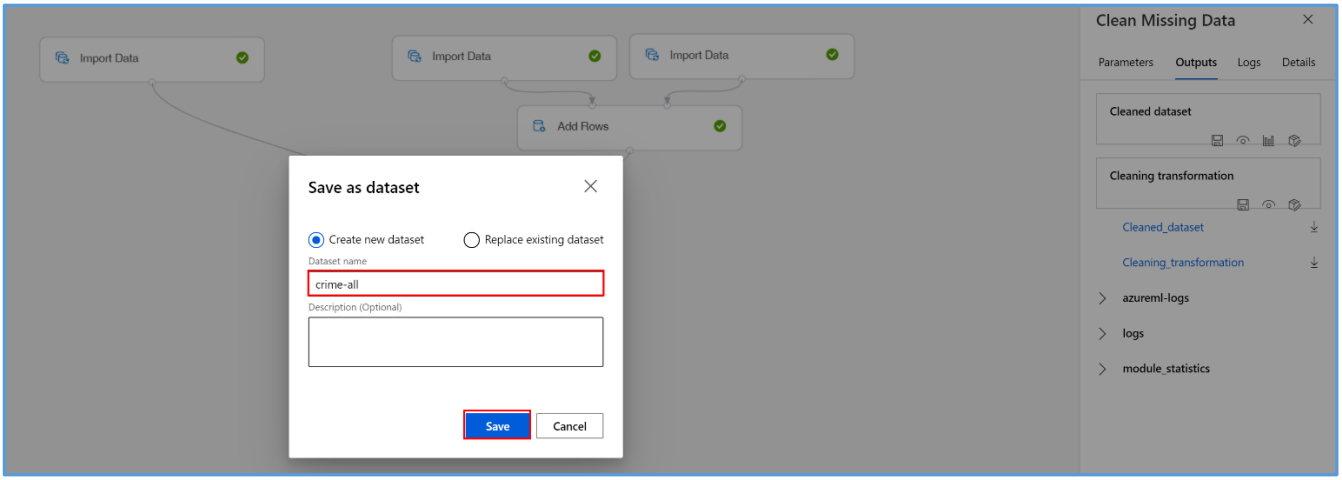
Please note that the button name in the UI is changed from **Run** to **Submit**.

Wait for pipeline run to complete. It will take around **6 minutes** to complete the run.

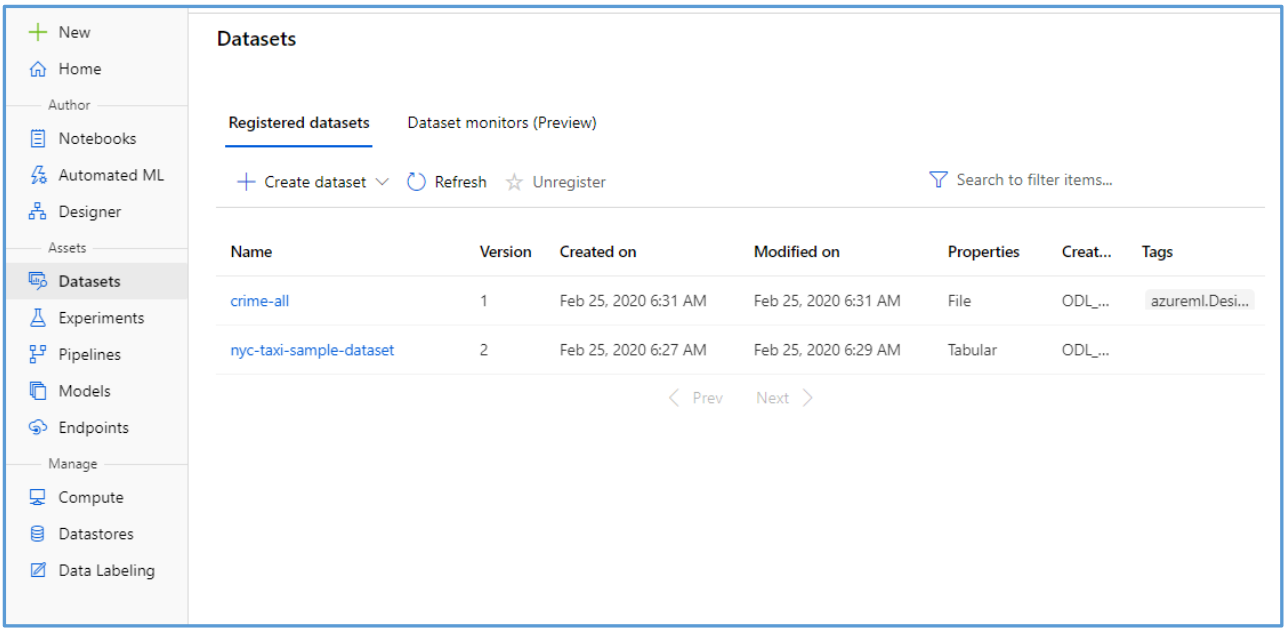
Task 4: Save the clean dataset

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

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



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.

Chapter 5: Walkthrough: Import, Transform, and Export Data

- New pipeline created via Designer
- Set default compute target for the pipeline
- Import Data >> Parameters >> Data Source >> URL via HTTP
- Run the experiment
- Check output of the Import data
- Use multiple datasets & restructure them using various data transformation tasks
- Use two data import tasks using the same data source URL via HTTP.
- Add both the Files [Combine both datasets using Add Row Task]
- Use another Add Row task for joining the Intermediate file [Output obtained by Adding two data import tasks] with the Original Import data.
- Clean missing data is the next task. Mention the names of the columns that needs to be, cleaned. [X coordinate , Y coordinate]
- Minimum missing value ratio set at 0.1 and Maximum missing value ratio has been set at 0.5. [**This condition must be met by each, and every column in order for the specified operation to apply. For example, assume you selected three columns and then set the minimum ratio of missing values to .2 (20%), but only one column actually has 20% missing values. In this case, the clean-up operation would apply only to the column with over 20% missing values. Therefore, the other columns would be unchanged**]
- Check the output of the Cleaned missing data & save it as a dataset. This is the registered dataset, which is the result of the particular run.

How to use Add Rows

To concatenate rows from two datasets, the rows must have exactly the same schema. This means, the same number of columns, and the same type of data in the columns.

1. Drag the **Add Rows** module into your experiment, you can find it under **Data Transformation**, in the **Manipulate** category.
2. Connect the datasets to the two input ports. The dataset that you want to append should be, connected to the second (right) port.
3. Run the experiment. The number of rows in the output dataset should equal the sum of the rows of both input datasets.

   If you add the same dataset to both inputs of the **Add Rows** module, the dataset is, duplicated.

Chapter 6: Managing Data

As we just discussed, Azure Machine Learning has two data management tools that we need to consider: **Datastores** and **datasets**. At first, the distinction between the two may not be entirely clear, so let us have a closer look at what each one does and how they are related.

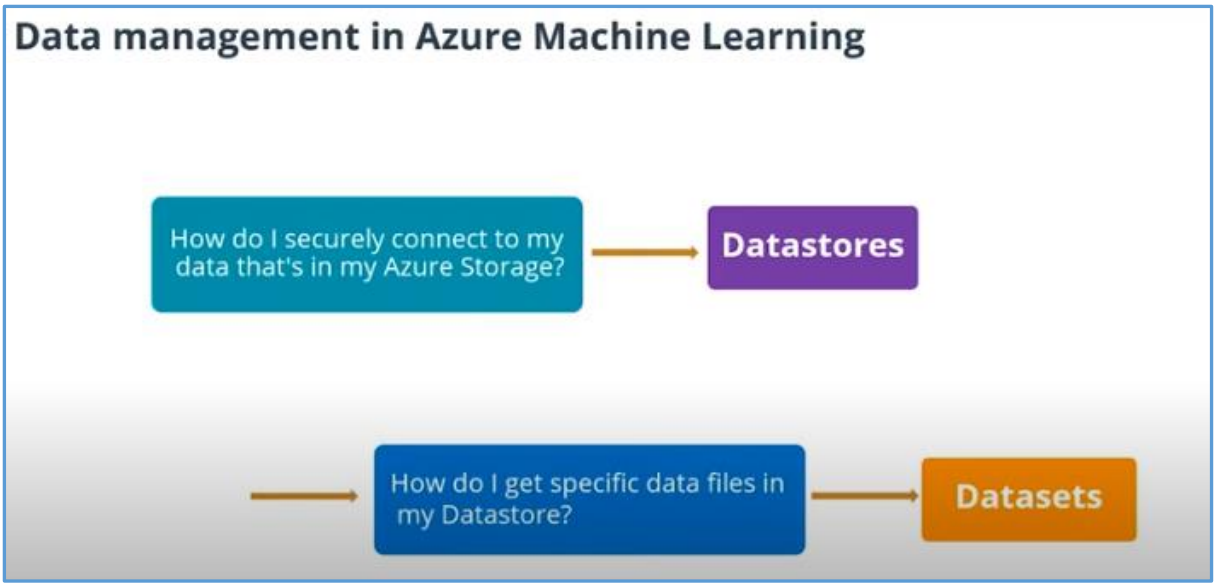
Datastores –They can be, used to access our storage in a secure manner without having to hard code connection information in our scripts. They are the Abstraction layer in Azure ML.

They offer a layer of abstraction over the supported Azure storage services. They store all the information needed to connect to a particular storage service. Datastores provide an access mechanism that is independent of the computer resource that is, used to drive a machine learning process.

Datasets - Tabular datasets, the following datasources are, supported in the designer: Delimited files, JSON files, Parquet files or SQL queries.[**How do I get specific data files, specific data into Datastores?**]. Datasets point to specific set of files that contain Training/Validation or Test data that we use in Machine Learning process.

They are resources for exploring, transforming, and managing data in Azure ML. A dataset is essentially a reference that points to the data in storage. It is, used to get specific data files in the datastores.

Data management in Azure Machine Learning

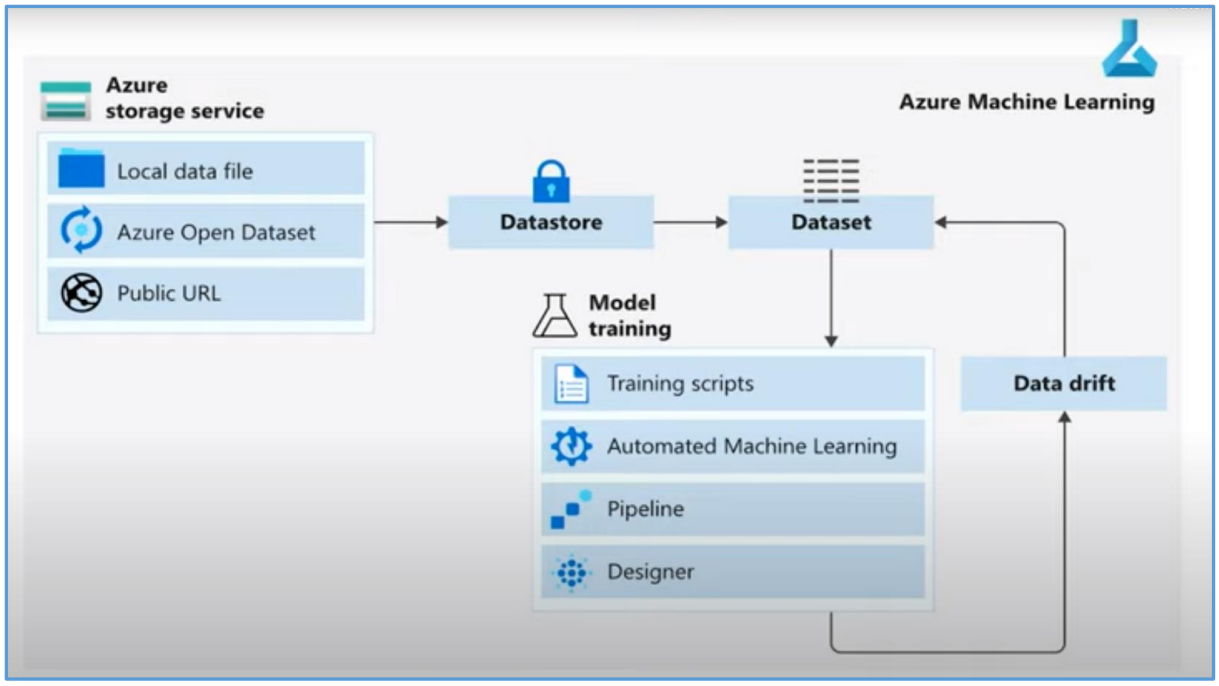


QUESTION 1 OF 2

For each of the descriptions below, mark whether it refers to **datastores** or **datasets**.

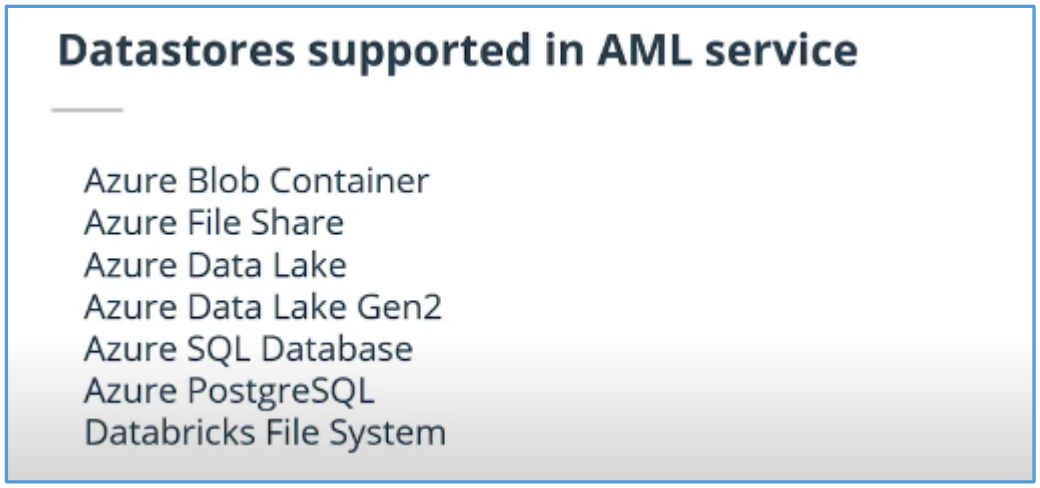
Submit to check your answer choices!

DESCRIPTION	DATASTORE OR DATASET?
Answers the question, "how do I get access to specific data files?"	Dataset
Answers the question, "how do I securely connect to the data in my Azure storage?"	Datastore
Keeps connection information internal, so it is not exposed in scripts.	Datastore
Points to specific files in your underlying storage that you want to use in your ML experiments.	Dataset

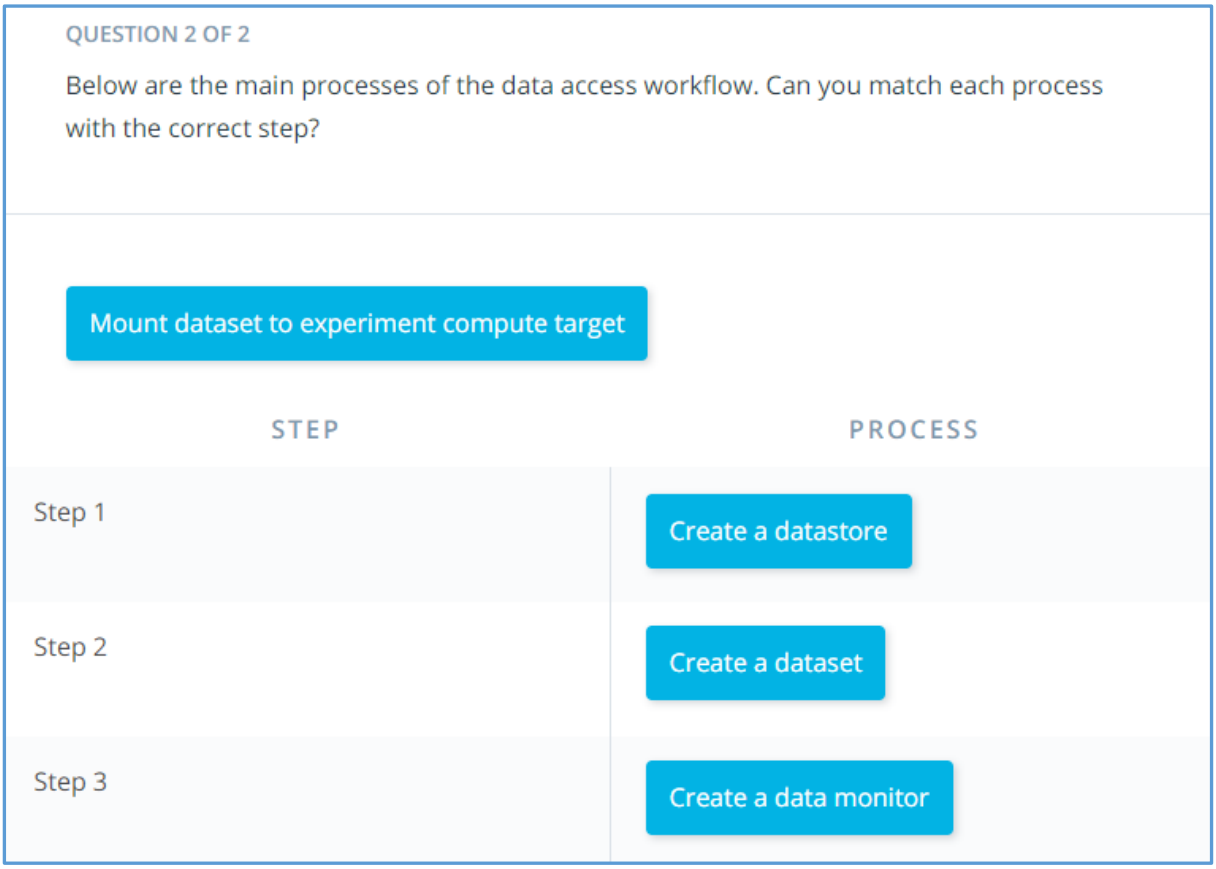


The steps of the data access workflow are:

1. **Create a datastore** so that you can access storage services in Azure.
2. **Create a dataset**, which you will subsequently use for model training in your machine learning experiment.
3. **Create a dataset monitor** to detect issues in the data, such as data drift.
   In the video, we mentioned the concept of *data drift*. Over time, the input data that you are feeding into your model is likely to change—and this is what we mean by **data drift**. Data drift can be problematic for model accuracy. Since you trained the model on a certain set of data, it can become increasingly inaccurate and the data changes more and more over time. For example, if you train a model to detect spam in email, it may become less accurate as new types of spam arise that are different from the spam on which the model was, trained.
   As we noted in the video, you can set up *dataset monitors* to detect data drift and other issues in your data. When data drift is, detected, you can have the system automatically update the input dataset so that you can retrain the model and maintain its accuracy.



Every ML Service comes with a default, preconfigured datastore, often called default datastore.
When we deploy Azure ML workspace Azure Blob container and Blob datastore is, automatically configured.



Chapter 7: Prelaunch Lab

Prelaunch your lab environment.

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

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

Chapter 8: More about Datasets

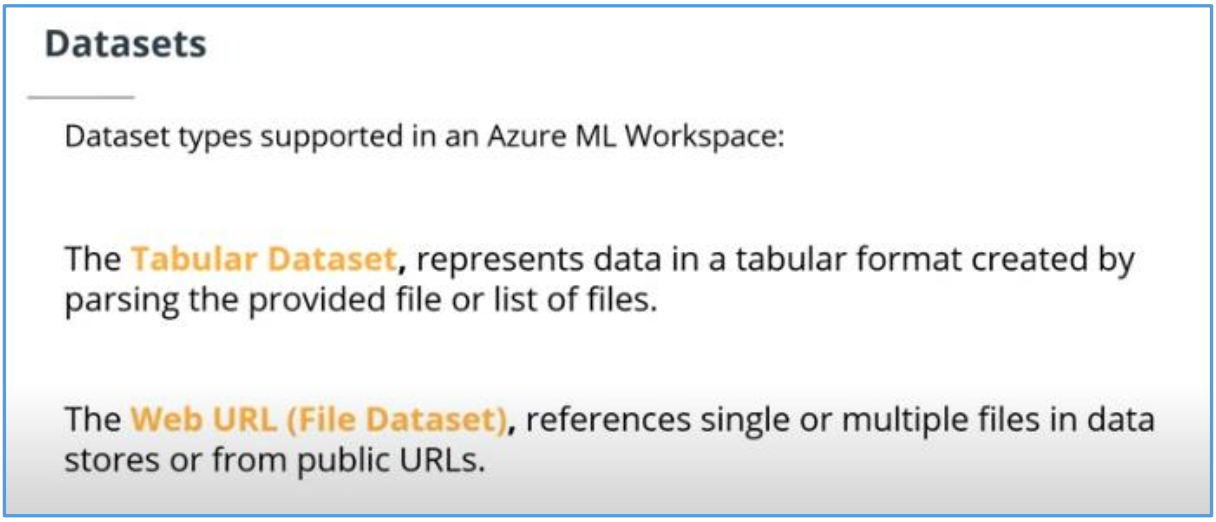
On the last page, we discussed the main features of *datastores* and *datasets*. Now, let us look a little more closely at how datasets work in Azure Machine Learning.

- Datasets interact with data in datastores; also, they package it into consumable objects for other ML tasks.

- Can be created from local files, public URLs, Azure open datasets or from specific files that have been uploaded into datastores.
- In Memory Scenarios – Pandas dataframes in python. Write data into local file upload into datastore and create a dataset out of it.
- Datasets are not copies of the data they are the references to the original data that is, kept in various storage services. Whenever a new dataset is, created in Azure ML there is no duplication & thus there is no cost for extra storage. Once dataset is, registered in the Azure ML workspace, it can be, shared or reused in various other experiments.
- Datasets can be created directly from Local files, Azure open datasets & Public URLs. This will connect the dataset with the default datastore, which was originally, created.

In summary, here are some of the main things that datasets allow you to do:

- **Have a single copy of some data in your storage, but reference it multiple times—so that you do not need to create multiple copies each time you need that data available.**
- Access data during model training without specifying connection strings or data paths.
- More easily share data and collaborate with other users. Like, multiple members of the Team looking at the same dataset.
- Bookmark the state of your data by using dataset versioning



You would do versioning most typically when:

- New data is available for retraining.
- When you are applying different approaches to data preparation or feature engineering.

Keep in mind that there are two dataset types supported in Azure ML Workspace:

- The **Tabular Dataset**, which represents data in a tabular format created by parsing the provided file or list of files.
- The **Web URL (File Dataset)**, which references single or multiple files in datastores or from public URLs.

Dataset & hence a dataset version is just a reference to our original data that is stored in storage service. Thus, we need to ensure that the original data is not, changed because this would invalidate the dataset or dataset version.

QUIZ QUESTION

Which of the following are true statements about Azure Machine Learning datasets?

(Select all that apply.)

- [ ] Datasets are copies of your data files that are copied by Azure onto your machine as needed.
- [x] Datasets are references that point to the data in your storage service; they are not actually copies, so no extra storage cost is incurred.
- [x] If you don't have an Azure storage service, you can create a dataset directly from local files, public urls, or an Azure Open Dataset.

Chapter 9: Lab - Create & version a dataset

Lab 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 are not 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 exercise, we use a modified version of the original CSV file, which includes collected records for five months (January - May).

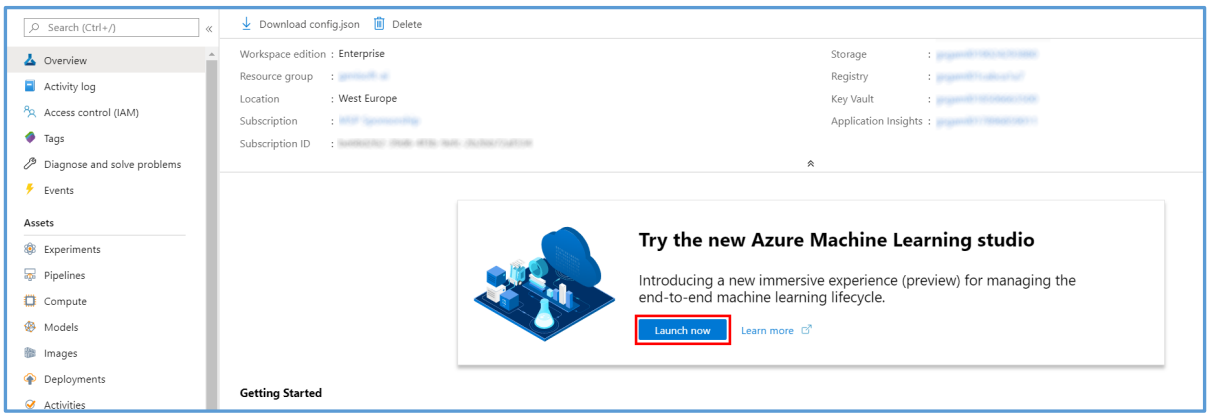
The second exercise demonstrates how we can create a new version of the initial dataset when new data is collected (in this case, we included records collected in June in the CSV file).

Exercise 1: Register Dataset with Azure Machine Learning studio

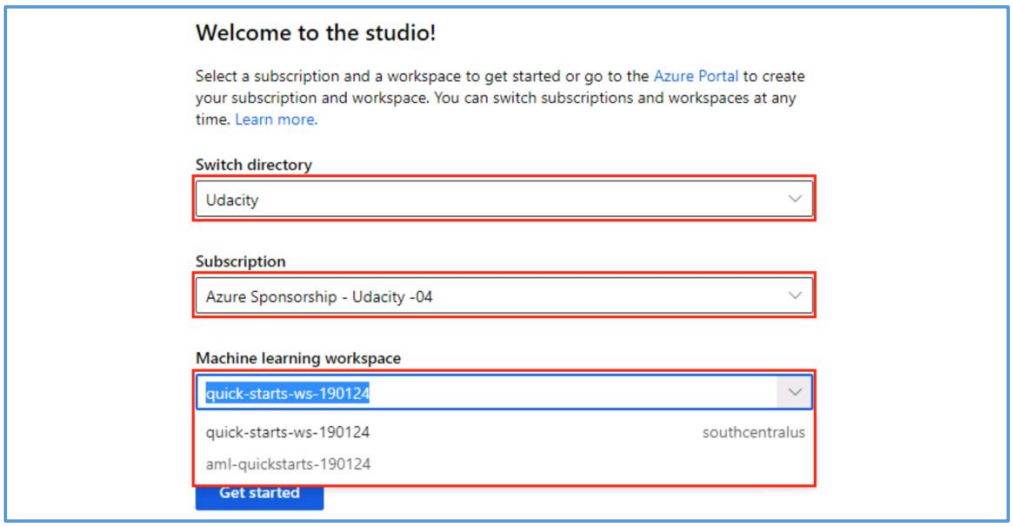
Task 1: Upload Dataset from web file

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

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

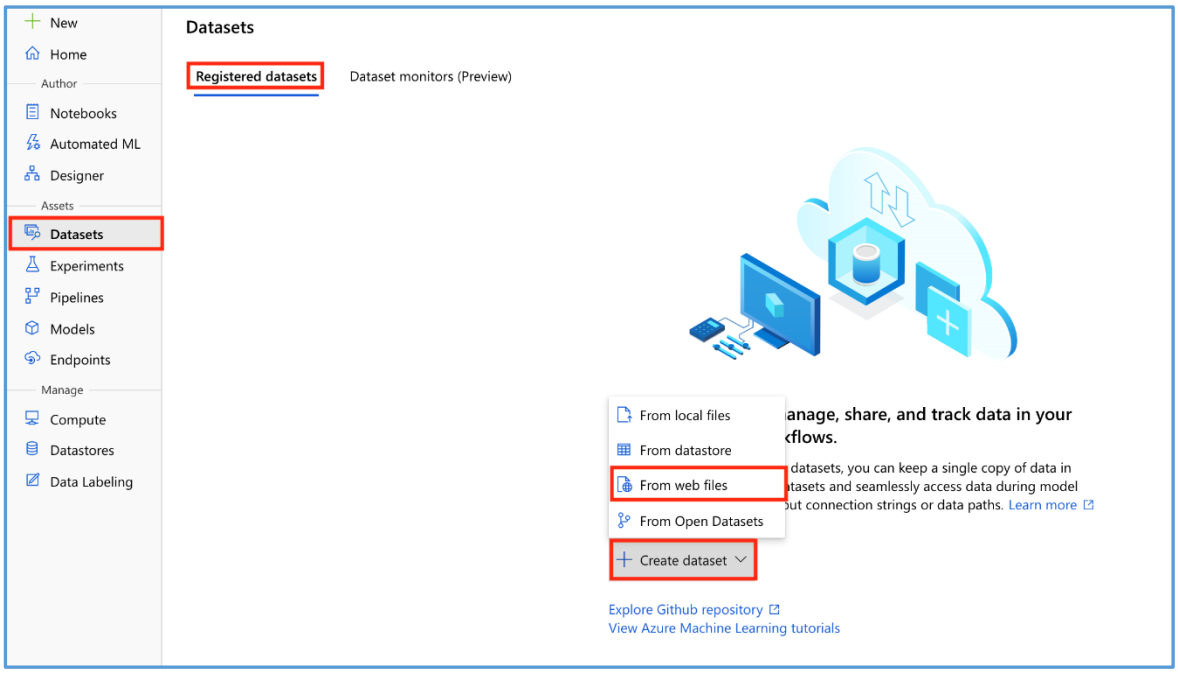


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



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 does not matter which) and then click **Get started**.

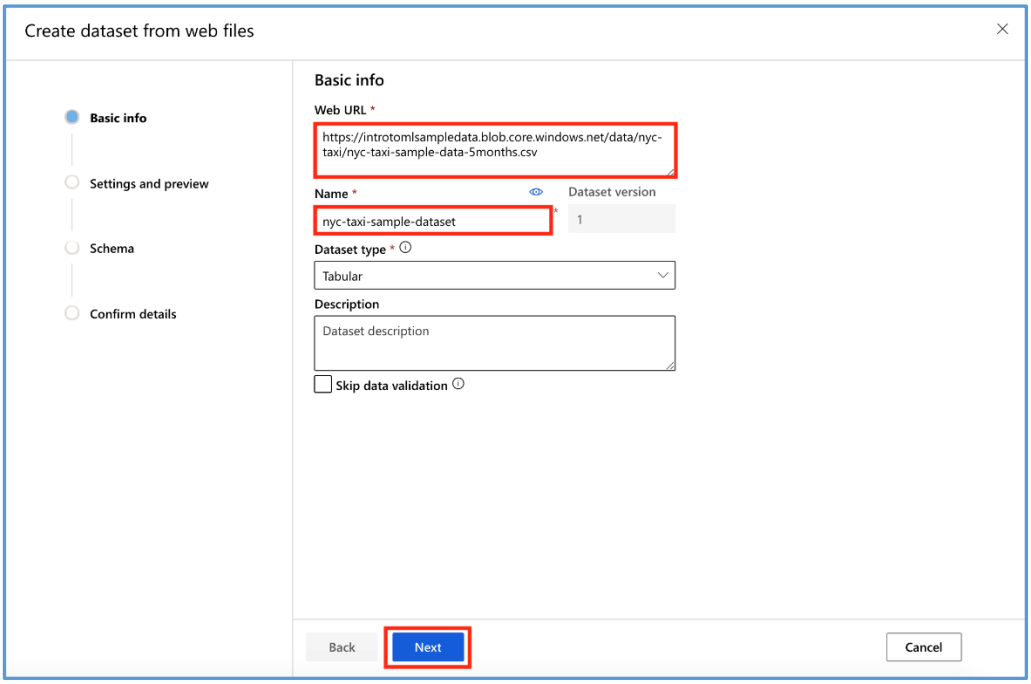
4. From the studio, select **Datasets, + Create dataset, from web files**. This will open the `Create dataset from web files` dialog on the right.



5. 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`

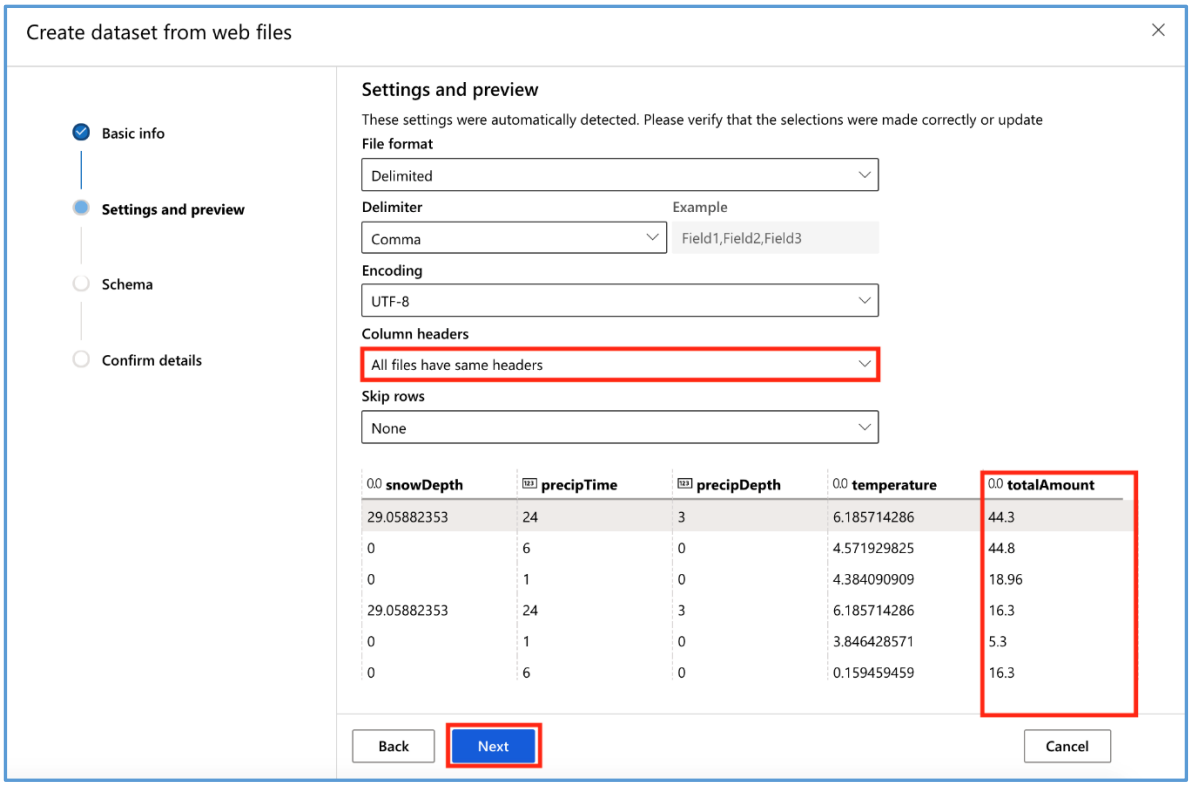
    2. Name: `nyc-taxi-sample-dataset`



Task 2: Preview Dataset

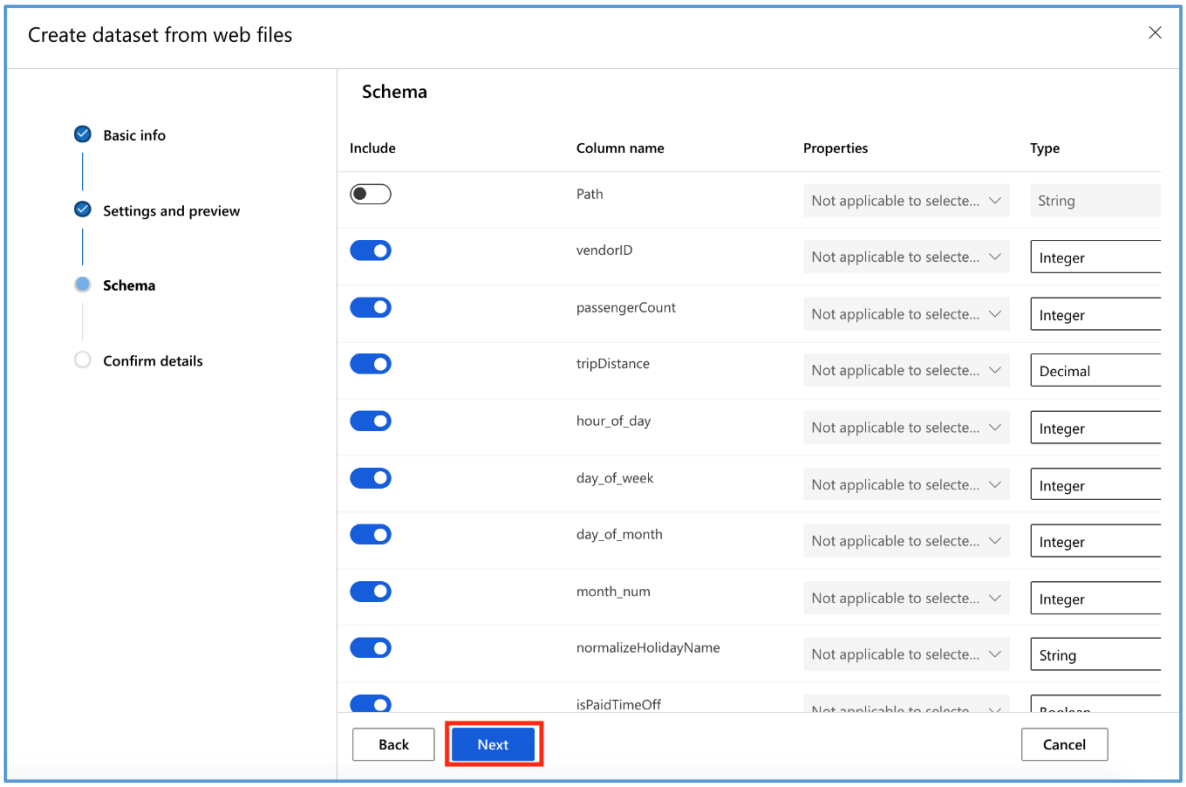
1. On the Settings and preview panel, set the **Column headers** drop down to `All files have same headers`.

2. Scroll the data preview to right to observe the target column: `totalAmount`. After you are done reviewing the data, select **Next**



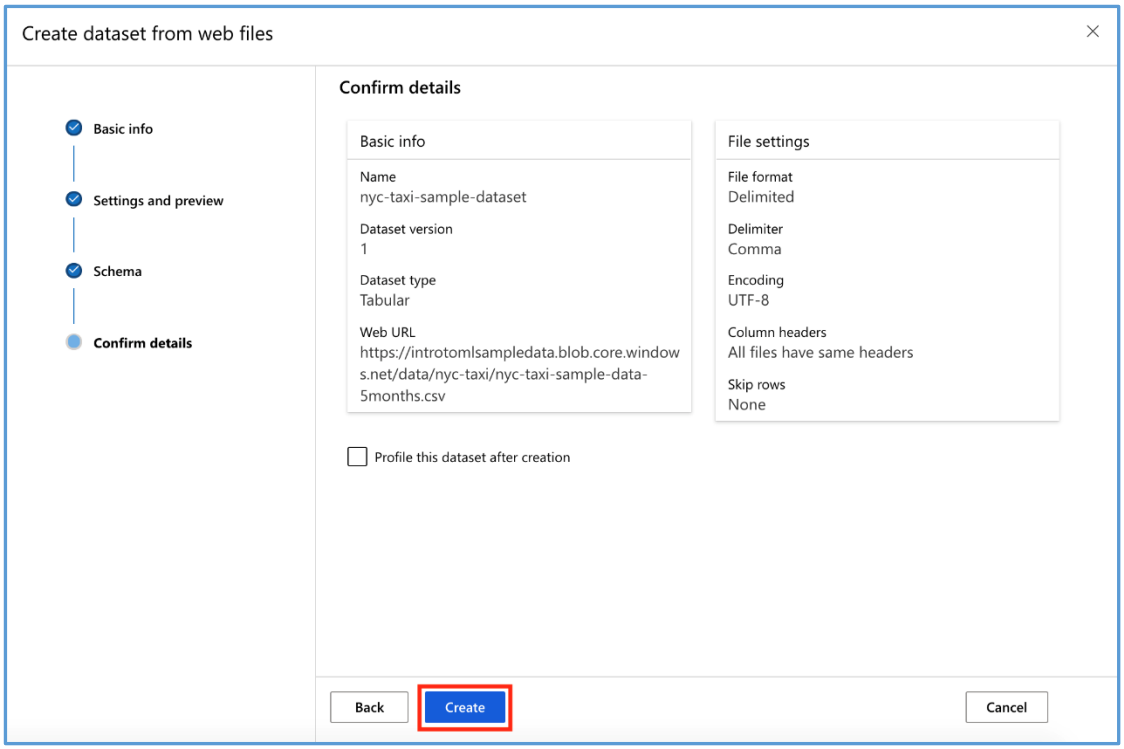
Task 3: Select Columns

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**

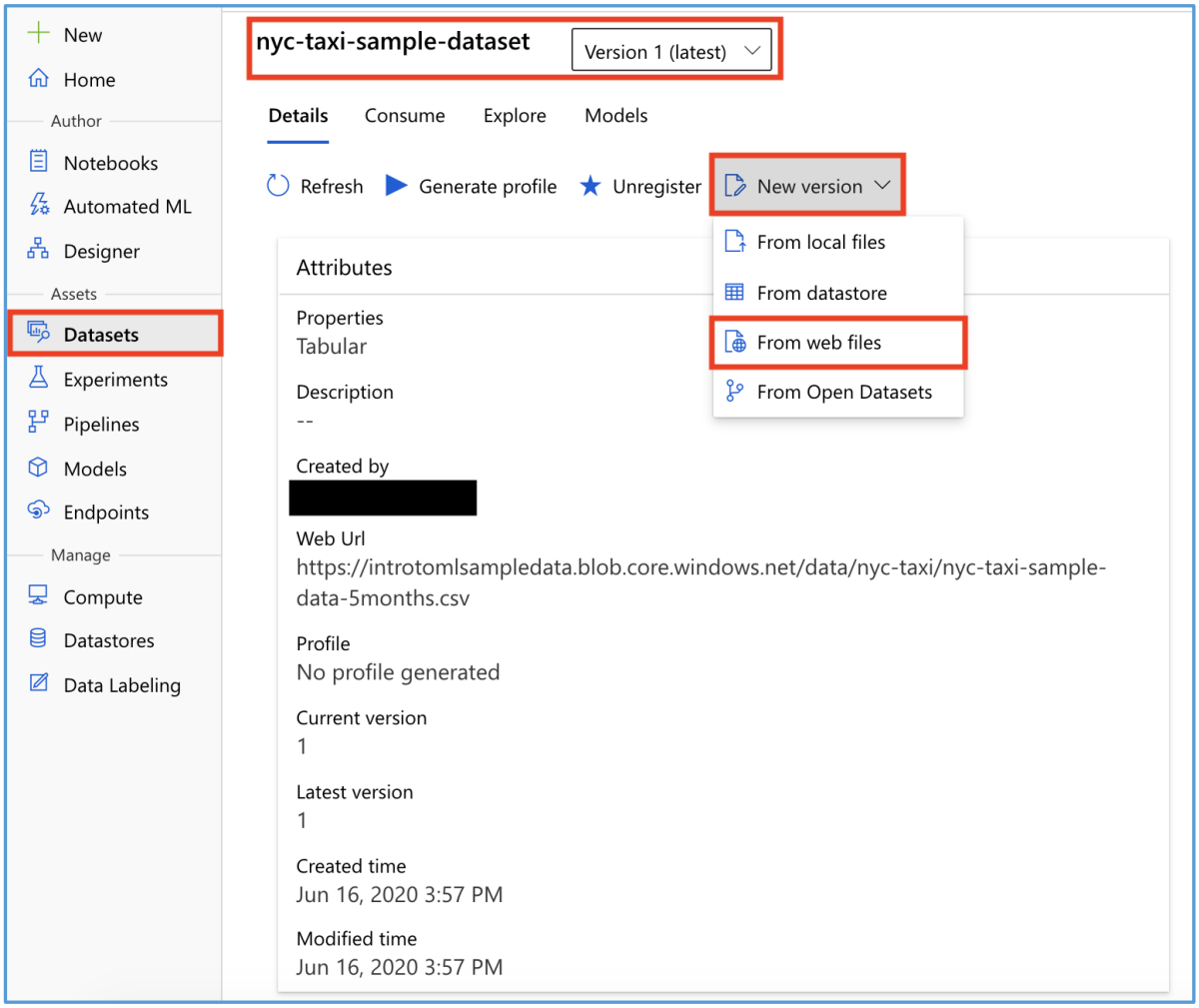


Exercise 2: Create a version of the existing Dataset

Task 1: Register new dataset version

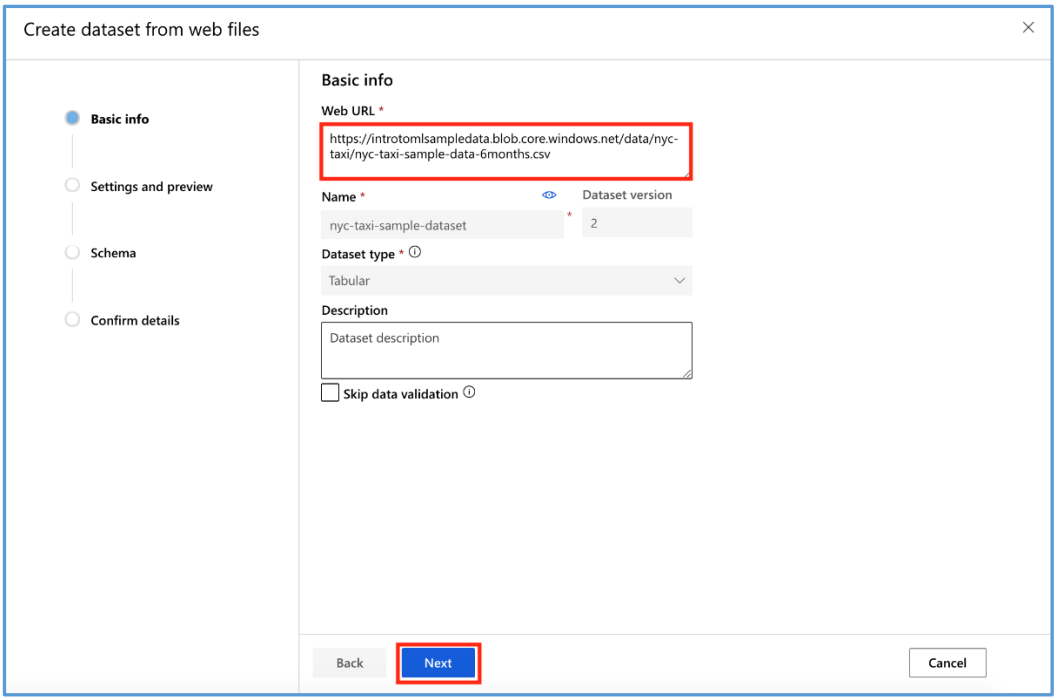
1. From the Azure Machine Learning studio, select **Datasets** and select the `nyc-taxi-sample-dataset` dataset created in the first exercise. This will open the `Dataset details` page.

2. Select **New version, From web files** to open the same `Create dataset from web files` dialog you already entered in the first exercise.



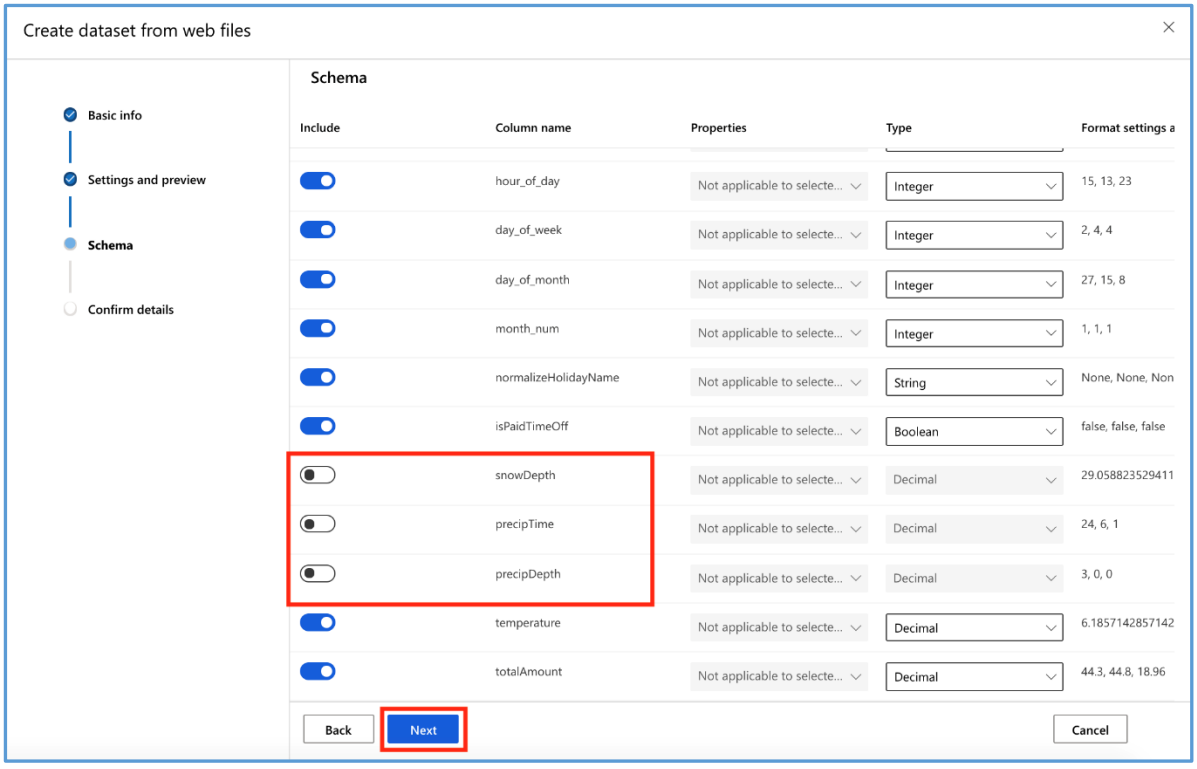
3. This time, the **Name** and **Dataset version** fields are already, filled in for you. Provide the following information and select **Next** to move on to the next step:

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

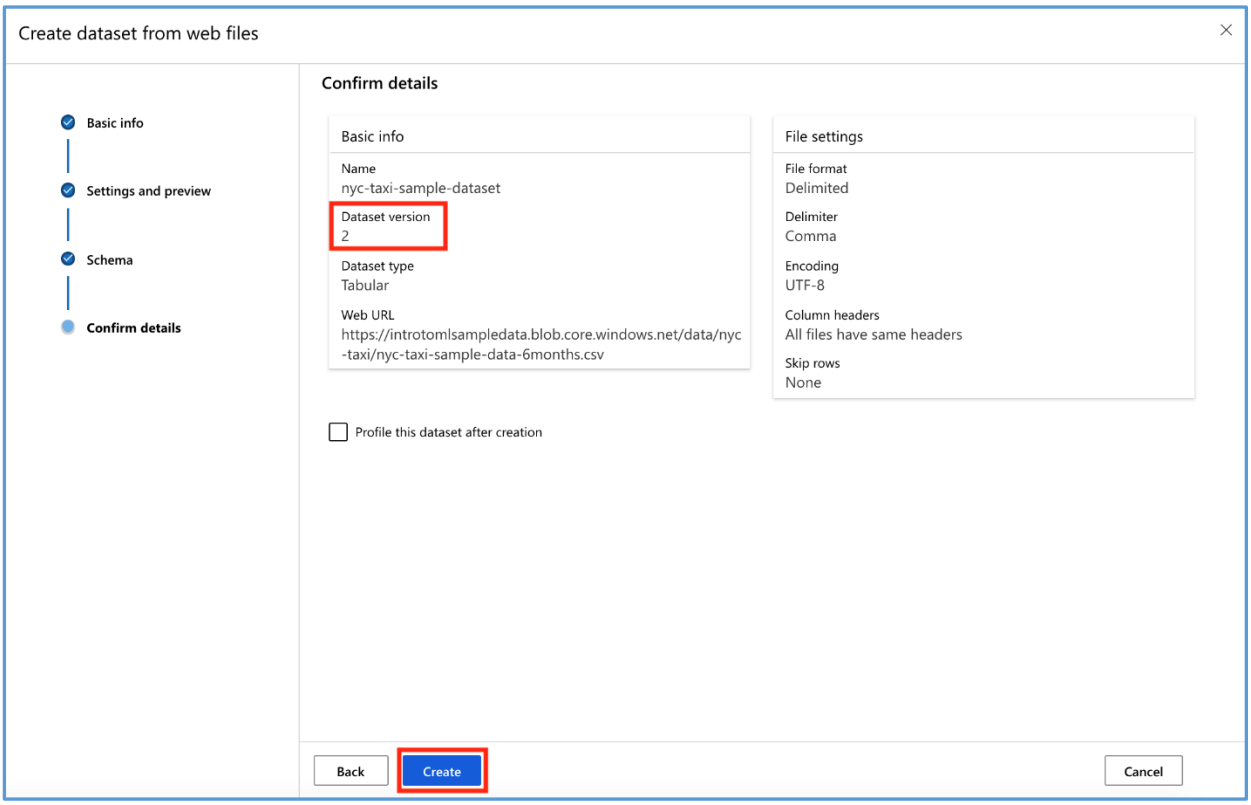


4. Select All files have the same headers in the **Column headers** drop-down and move on to the schema selection step.

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.

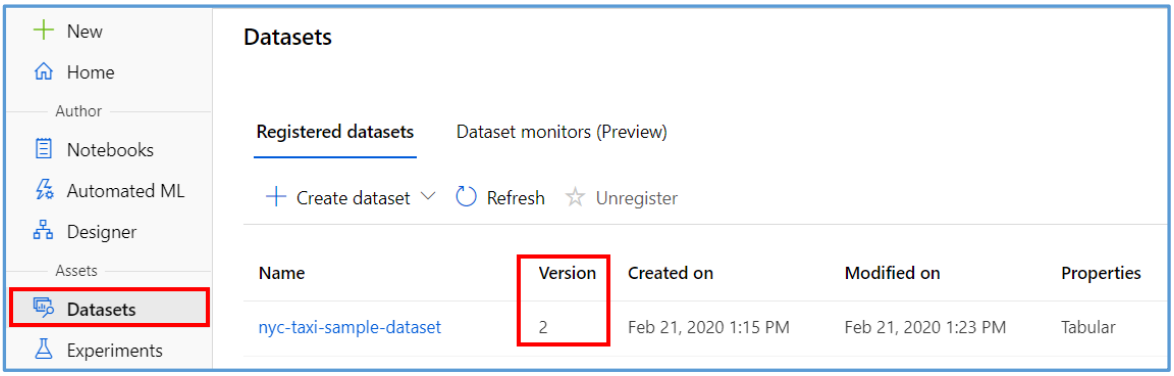


6. Notice the Dataset version value in the basic info section. Select **Create** to close the new version confirmation page.



Task 2: Review both versions of the dataset

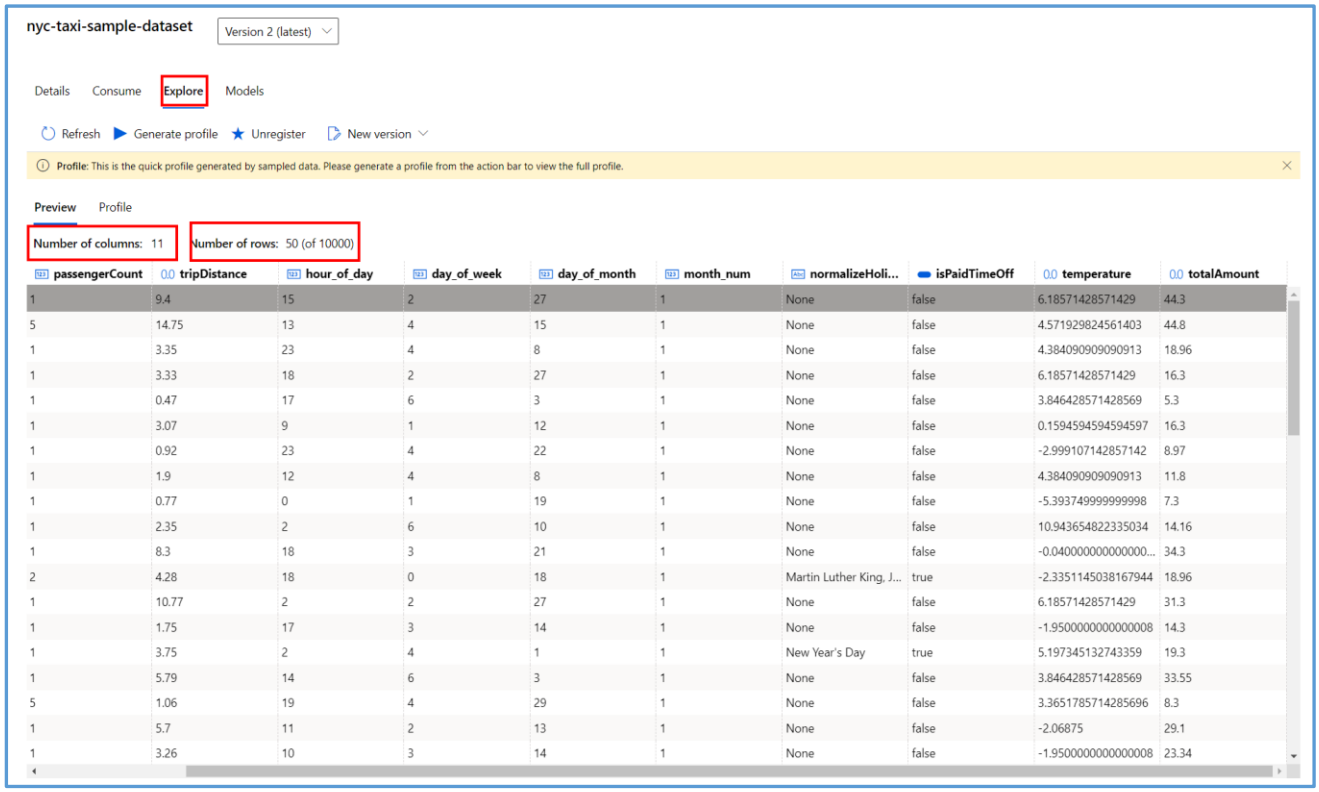
1. Back to the **Datasets** page, in the **Registered datasets** list, notice the version value for the nyc-taxi-sample-dataset dataset.



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:

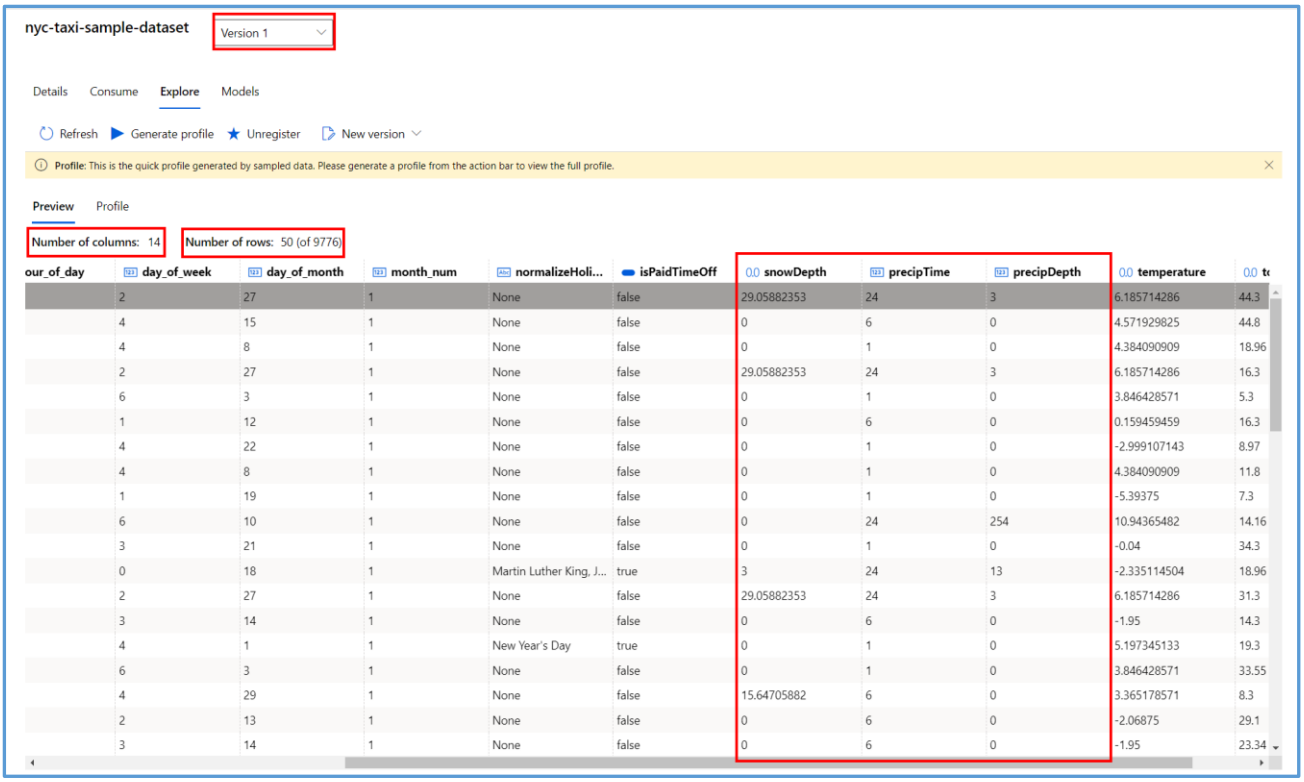
    - **Number of columns**: 11

- **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	6.18571428571429	16.3
1	0.47	17	6	3	1	None	false	3.846428571428569	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	3.846428571428569	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	tc
	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.

Chapter 10: Walkthrough - Create & version a dataset

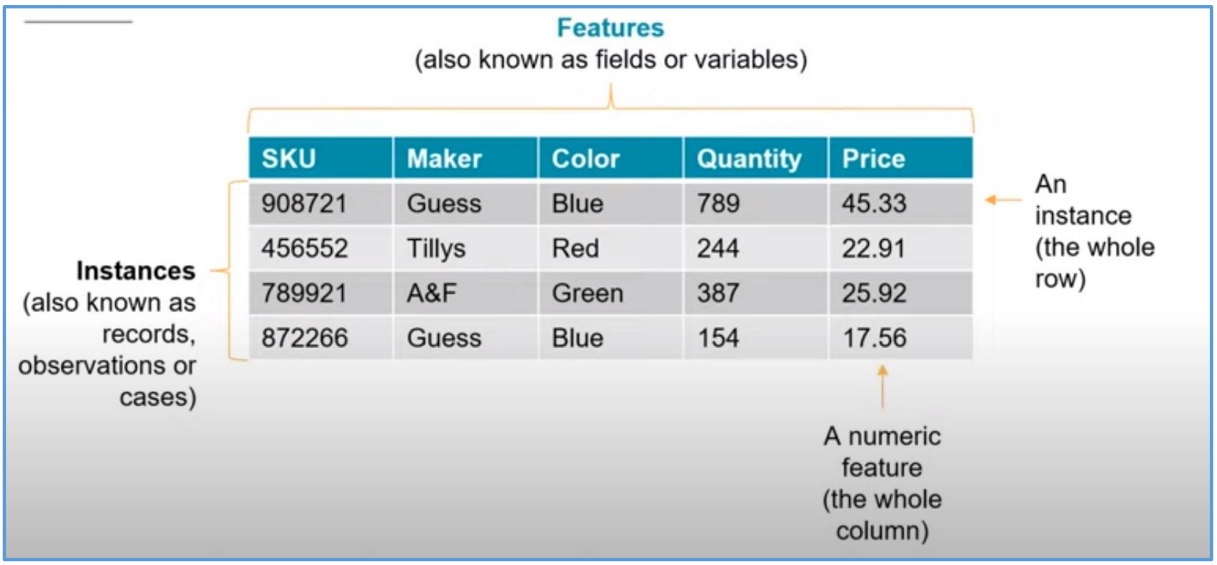
- Versioning of datasets is important for traceability of our ML model training tasks.
- Register dataset with Azure ML studio.
- Upload a dataset from a local file to create Version 1 of the dataset.[Column #14 & Row # 9776]
- Register a new version for the same dataset and upload a modified version of the csv file [3 Columns are removed, now Column # 11]. We have the same dataset conceptually, the logical structure of it. The current version of the dataset is Version 2. Thus, we have created 2 different versions.

Chapter 11: Introducing Features

In the previous lesson, we took, a look at some examples of *tabular data*:

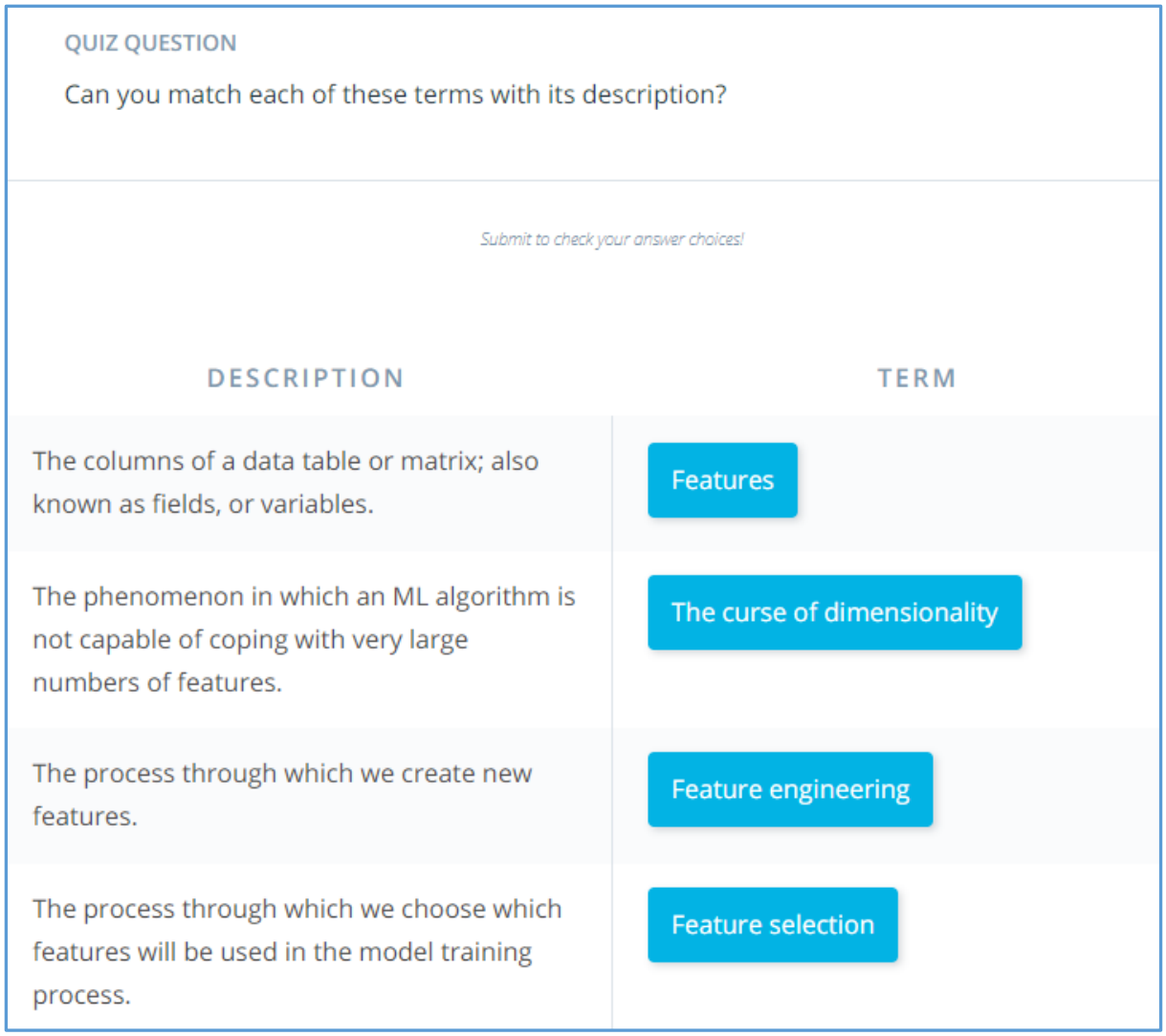
SKU	Make	Color	Quantity	Price
908721	Guess	Blue	789	45.33
456552	Tillys	Red	244	22.91
789921	A&F	Green	387	25.92
872266	Guess	Blue	154	17.56

Note: *We have been referring to this as a data table, but you will also see data in this format called a* ***matrix****. The term matrix is commonly, used in mathematics, and it refers to a rectangular array—which, just like a table, contains data arranged in rows and columns.*



Recall that the columns in a table can be, referred to as **features**. In the above example, color and quantity are *features* of the products. In the last lesson, we mentioned briefly that **feature engineering** is an important part of data preparation. In this lesson, we will look at this topic in more detail.

In many cases, the set of initial features in the data is not enough to produce high quality trained machine learning models. You can use **feature engineering** to derive new features based on the values of existing features. This process can be as simple as applying a mathematical function to a feature (such as adding 1 to all values in an existing feature) or it can be as complex as training a separate machine-learning model to create values for new features.
Once you have the features, another important task is selecting the features that are most important or most relevant. This process is, called **feature selection**.
Many machine-learning algorithms cannot accommodate a large number of features, so it is often necessary to do **dimensionality reduction** to decrease the number of features.



Chapter 12: Feature Engineering

> Feature engineering is one of the core techniques that can be used to increase the chances of success in solving machine learning problems.

- FE **[Feature Engineering]** increases the power of ML algorithms.
- FE uses the value of existing features to derive new features, new values that is helpful for the model during training process.
- We have a dataset about Customer behaviour. What is more relevant to predict the future behaviour of customer?
  - ✓ Number of purchases last month
  - ✓ Average Number of purchases in last 6 months
  - ✓ Number of purchases in last 3 months
  - ✓ Number of purchases in last month compared with the Number of purchases in the month before

There are multiple variations, all being, tied to the original information of Number of purchases made by the customer. FE helps us to derive these features.

- FE is one of the most important tasks as it lays the foundation for good models with good capabilities in terms of prediction and stability.
- FE is always not necessary.

There are various places where Feature Engineering can be implemented

- FE can be done right at the Data Source
- FE can be, performed in a dedicated environment.
- FE can also be, performed during the process of training the model.

Classical ML is much more reliant on feature engineering than Deep Learning

- In case of Classical ML, we need to create the features explicitly.
- In case of Deep learning most of these, Feature engineering occurs naturally within neural network itself.

Deep Learning can be used for the implementation of certain Feature Engineering processes (like embedding)

QUESTION 1 OF 3

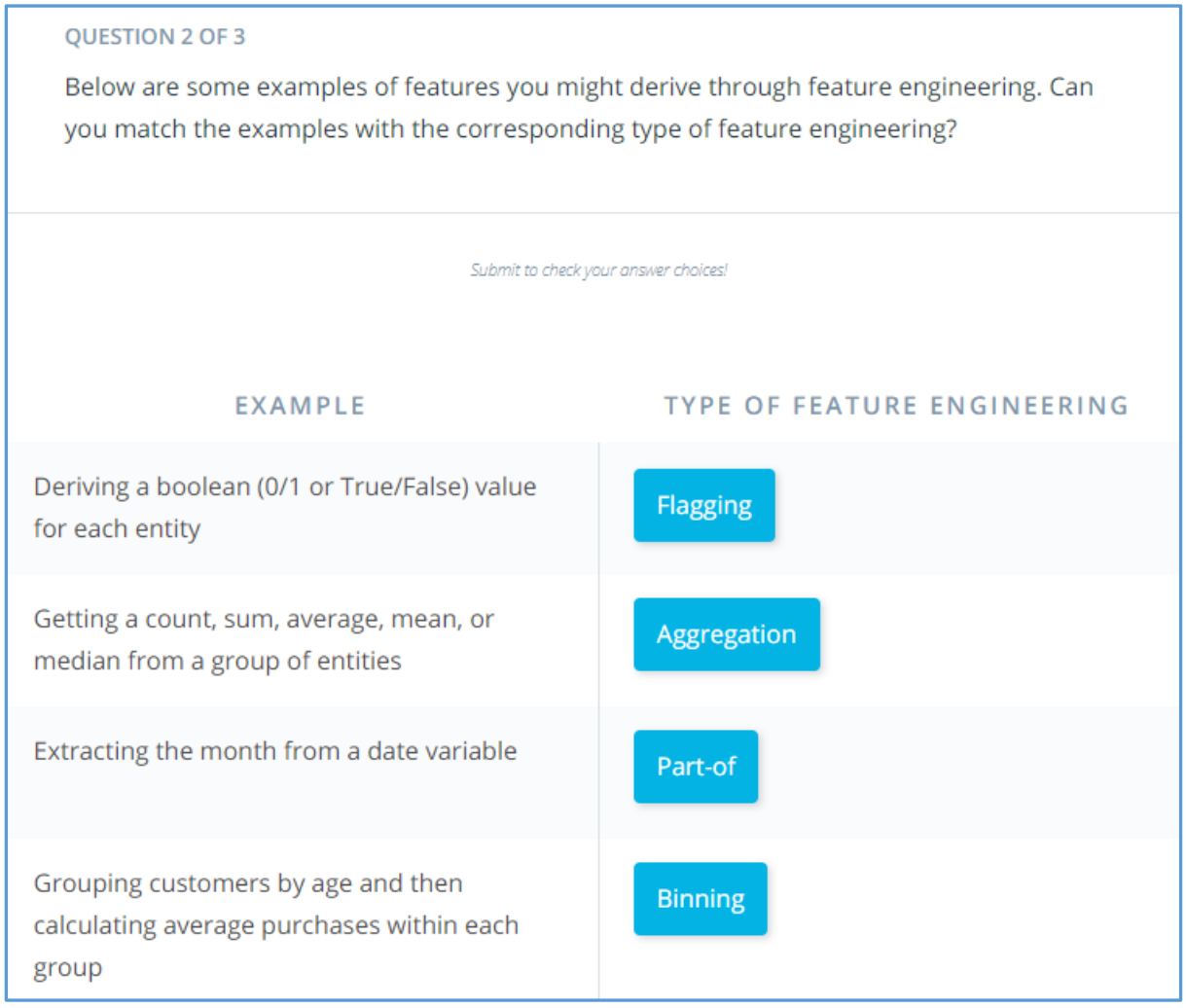
Which of the following are true statements about feature engineering?

(Select all that apply.)

- [x] Feature engineering manipulates existing features in order to create new features, with the goal of improving model training.
- [x] Feature engineering can be implemented in multiple places, such as at the data source or during model training.
- [ ] Deep learning depends on feature engineering much more than classical machine learning.
- [x] Classical machine learning depends on feature engineering much more than deep learning.

Feature Engineering Tasks

- **Aggregation –** Simple mathematical functions like Count/Sum/Distinct Count/Average/Mean/Median etc.
- **Part of –** Another type of FE task where we extract a part of a certain data structure. Like – Part of a date [Take the Date/Day/Month or Year]
- **Binning –** Group entities into bins and then apply aggregations over the bins
- **Flagging –** Deriving Boolean conditions that are, expressed through Boolean values. (True/False). Example- Did customer made, a purchase in last 6 months?
- **Frequency based –** Calculate the various frequencies for occurrences of certain amount of data
- **Embedding often referred as featured learning**
- **Deriving by example –** Aim to learn values of new features using examples of existing features.



Summary of Feature Engineering Approaches by Data Type

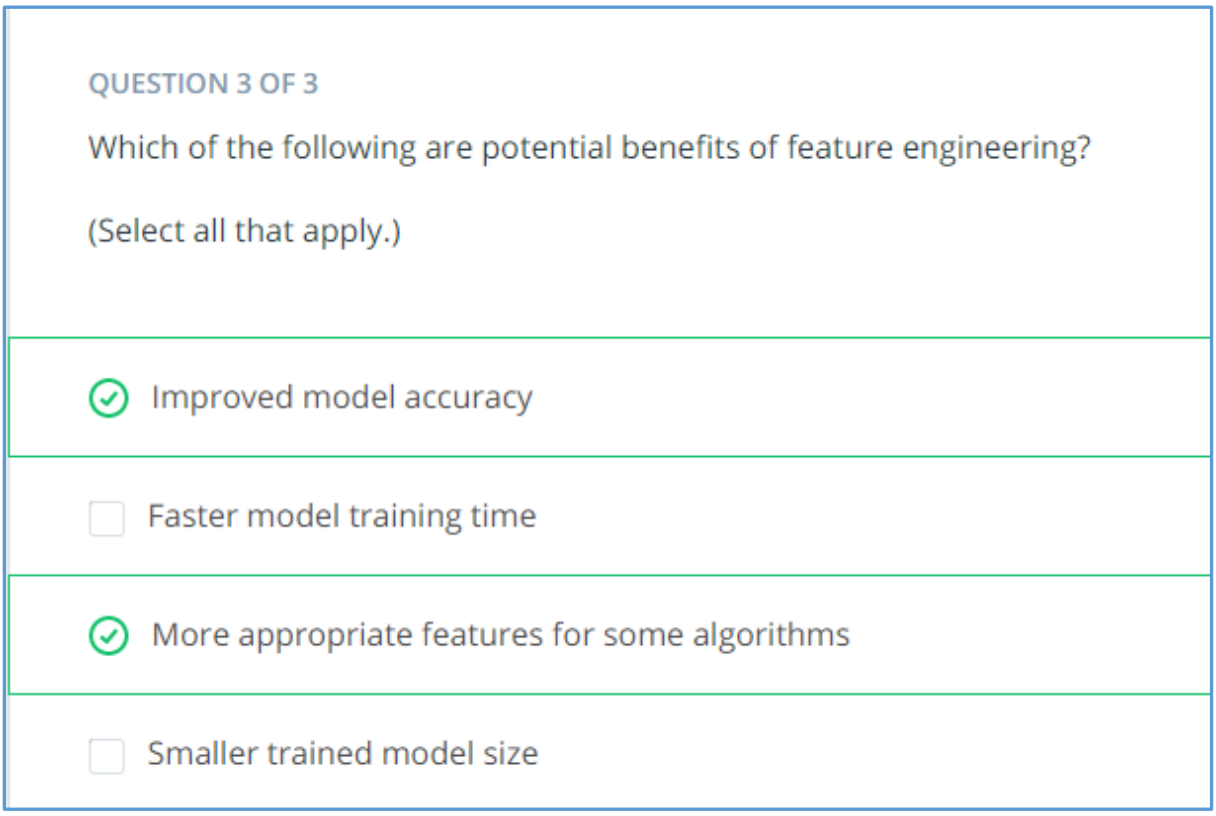
Some of the most widely used types of data used in Machine Learning are numbers, text, and image

- FE process is, applied differently depending upon the type of data incarnation.
- Numerical Data >> Tabular Format >> Classical feature engineering tasks

- Text is not suitable for numerical processing. Thus, we need to translate text into numerical format, which can be, accepted by ML algorithms. Text embedding [TF-IDF/Word Embedding]takes sequence of words which Is transformed into numerical format, which appears as numerical vectors
- Image has the same problem. RGB needs to be, translated. Image considered as Matrix of tuples of three values.[Red, Green and Blue channel].
  - ✓ 500 by 400 pixels image, where each pixel in the image has 3 values = 20,000 times 3
  - ✓ 60,000 numerical values – This is the numerical representation of the image

There is usually no dependency between the number of features and the **time needed to train a model.**

There is usually no dependency between the number of features and the **size of the resulting trained model.**



Chapter 13: Prelaunch Lab

Prelaunch your lab environment.

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

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

Chapter 14: Feature Selection

- Through FE with a given dataset, we create a number of features. Thus, number of features resulting from feature engineering could be huge.

> Which are the features that are most useful for a given model?

Feature Selection helps in providing answer to the question above. Process of filtering the features is very important as it helps in building highly accurate ML models.

> Reasons for using Feature Selection:
>
> Elimination of irrelevant, redundant, or (highly) correlated features
>
> Reduction of dimensionality to increase performance

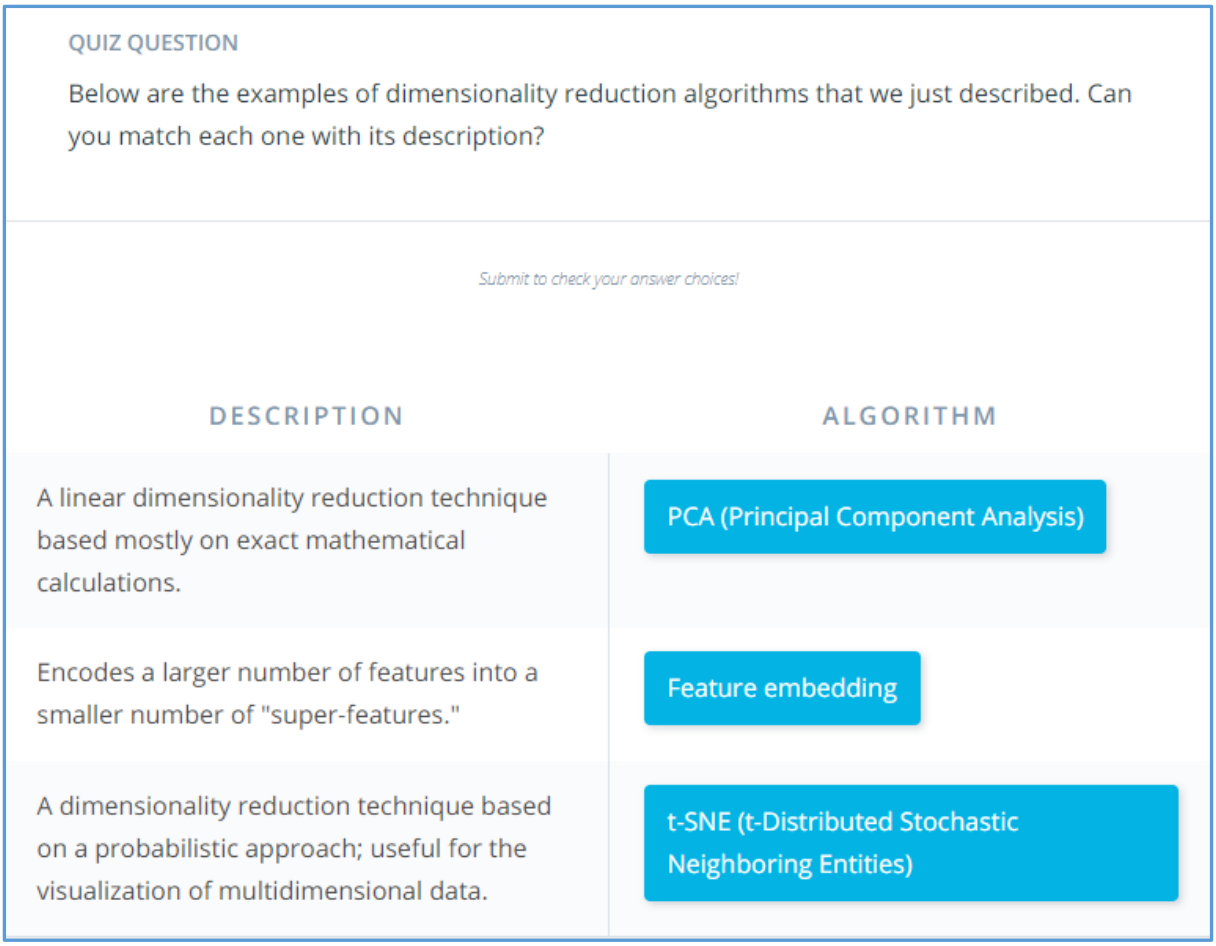
There are mainly two reasons for feature selection. Some features might be highly irrelevant or redundant. Therefore, it is better to remove these features to simplify the situation and improve performance. Additionally, it may seem like engineering more features is always a good thing, but as we mentioned earlier, many machine learning algorithms suffer from the ***curse of dimensionality***—that is, they do not perform well when given a large number of variables or features.

We can improve the situation of having too many features through **dimensionality reduction**.
Commonly used techniques are:

- **PCA (Principal Component Analysis**) – Linear technique that we use, which is, based on exact mathematical calculations. It is a statistical approach rather than an ML based approach.
- **t-SNE (t-Distributed Stochastic Neighbouring Entities)** – Probabilistic approach for reducing dimensionality. It has the remarkable feature of keeping close points from the multi-dimensional space close in the two dimensional space which the human brain can visualise.
- **Feature embedding-** Training a separate ML model to encode a large number of features into smaller number of features

Azure ML prebuilt modules:

- Filter-based feature selection: identify columns in the input dataset that have the greatest predictive power
- Permutation feature importance: determine the best features to use by computing the feature importance scores



Chapter 15: Lab – Engineer and Select features

Lab Overview

This lab demonstrates the feature engineering process for building a regression model using bike rental, demand prediction as an example. In machine learning predictions, effective feature engineering will lead to a more accurate model. We will use the Bike Rental UCI dataset as the 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

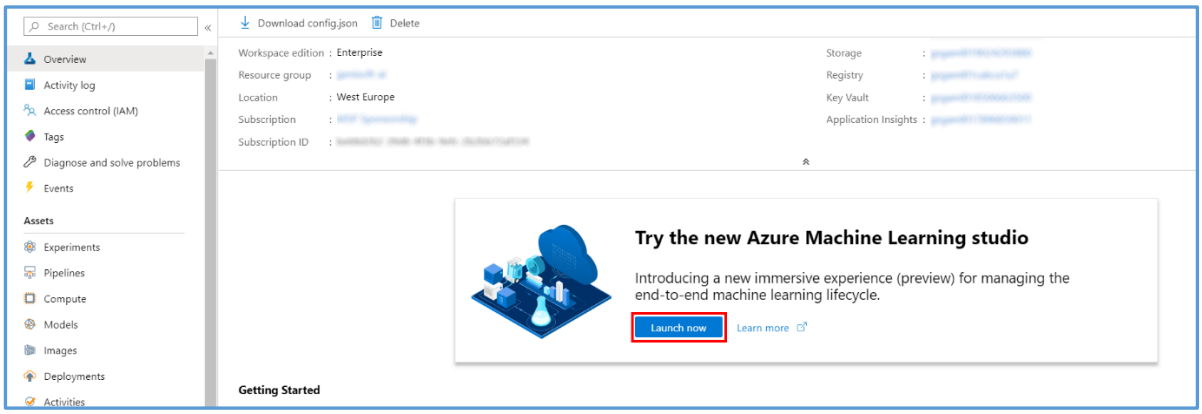
For the model, we are using regression because the number of rentals (the label column) contains continuous real numbers. As the algorithm for the experiment, we will be using the Boosted Decision Tree Regression.

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

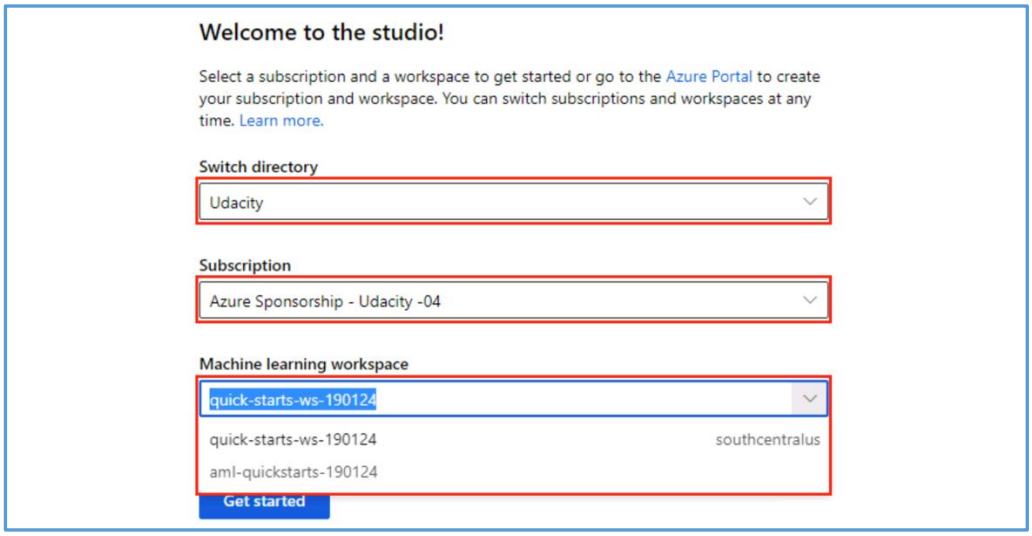
Task 1: Upload Dataset

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

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

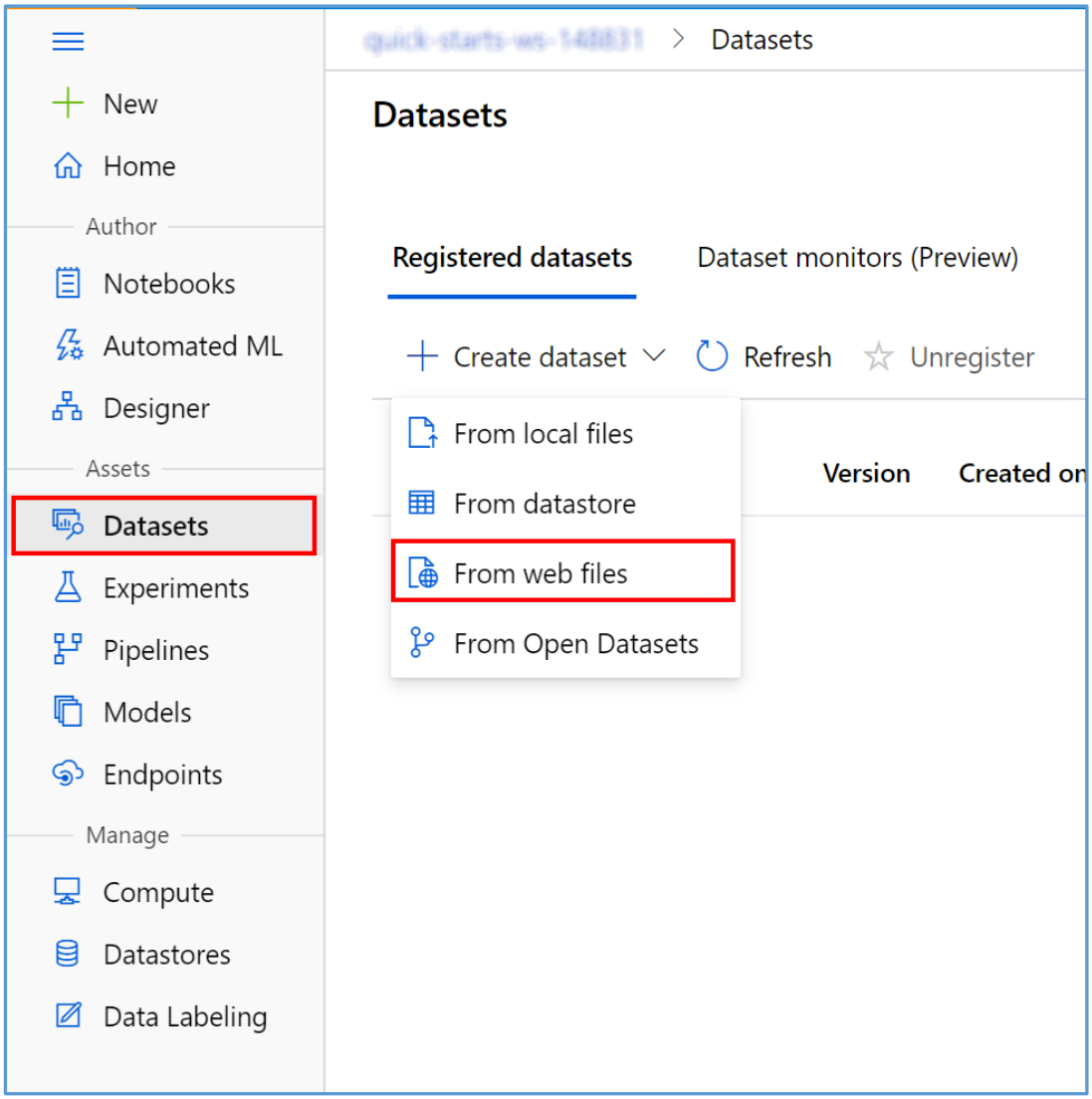


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



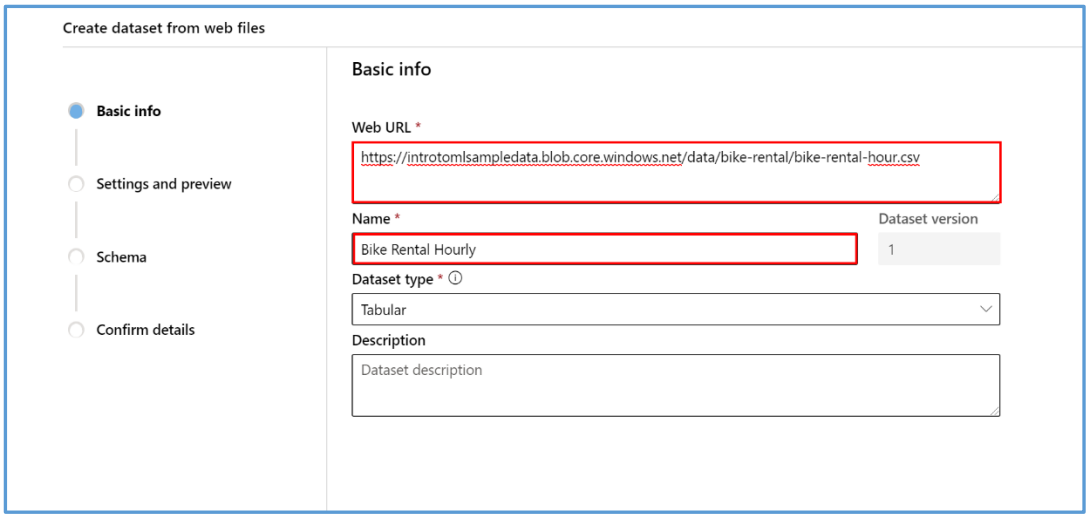
For the directory, select **Udacity** and for the subscription, select **Azure Sponsorship**. For the machine learning workspace, you may see multiple options listed. **Select any of these** (it doesn't matter which) and then click **Get started**.

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

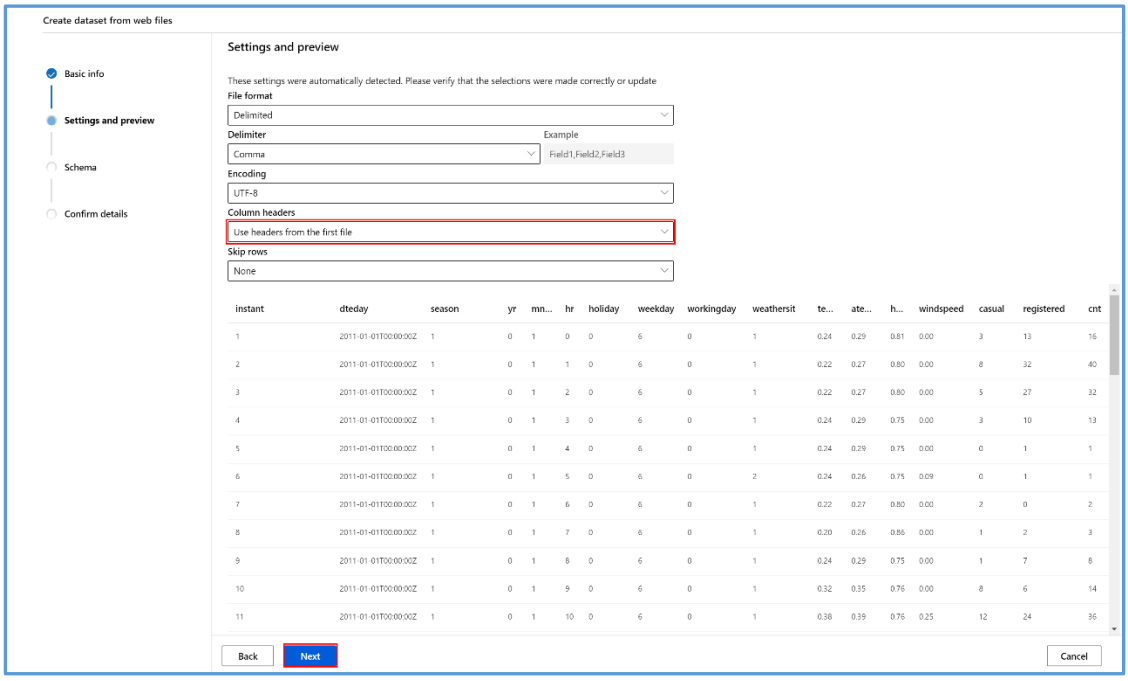


5. In the Web URL field provide the following URL for the training data file: https://introtomlsampledata.blob.core.windows.net/data/bike-rental/bike-rental-hour.csv

6. Provide Bike Rental Hourly as the Name, leave the remaining values at their defaults and select **Next**.

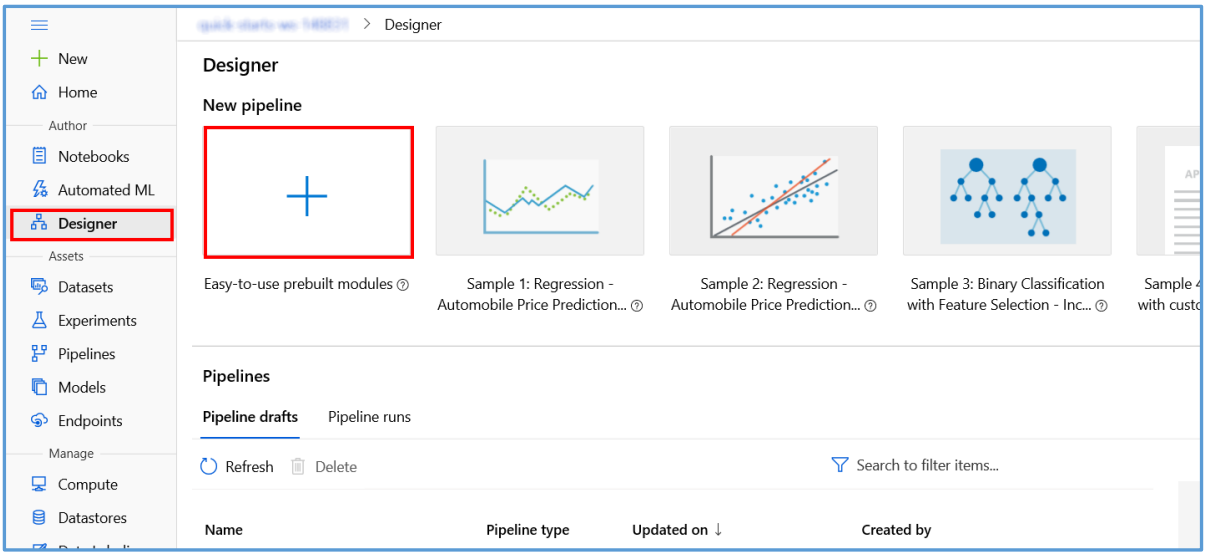


7. Select the option to Use headers from the first file in the **Settings and preview** dialog and then select **Next**, **Next** and **Create** to confirm all details in registering the dataset.



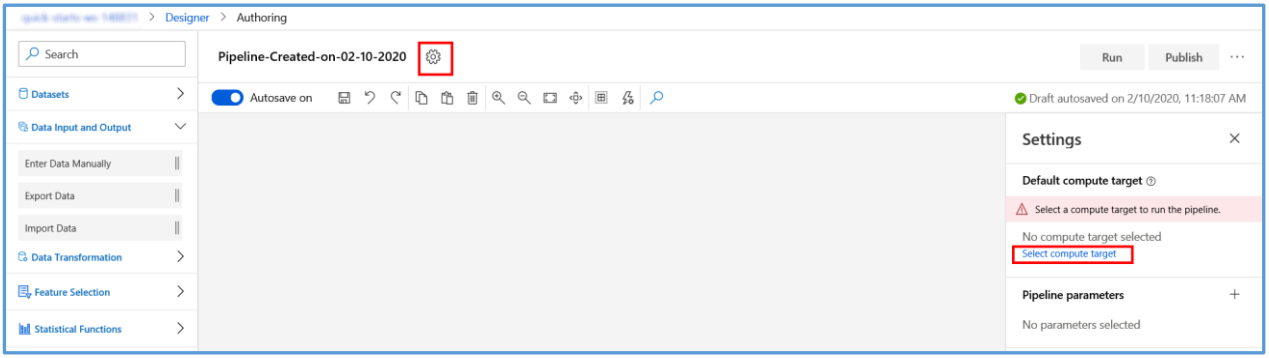
Task 2: Open Pipeline Authoring Editor

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**.

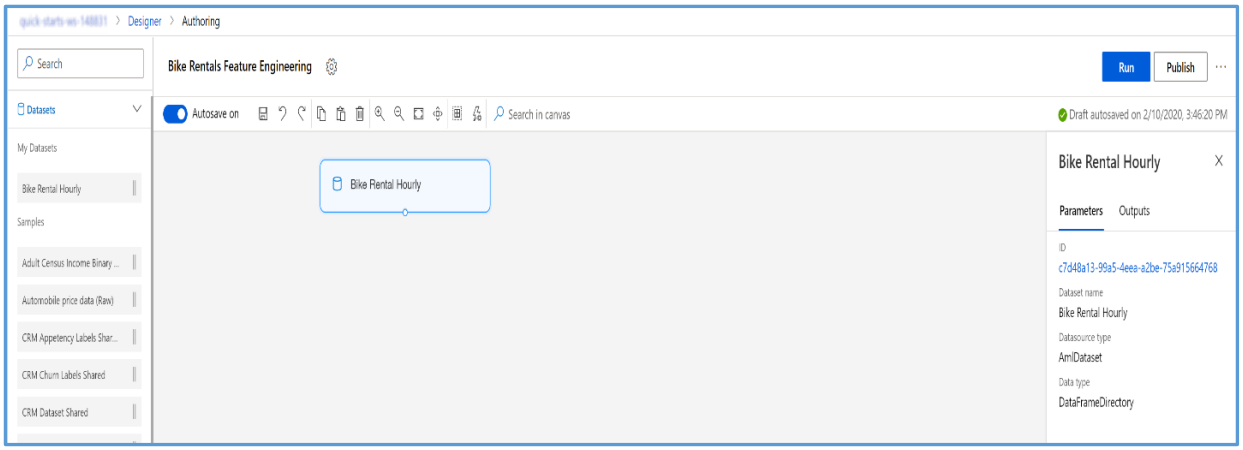


2. In the Set up compute target editor, select the existing compute target, choose a name for the pipeline draft: Bike Rental Feature Engineering and then select **Save**.

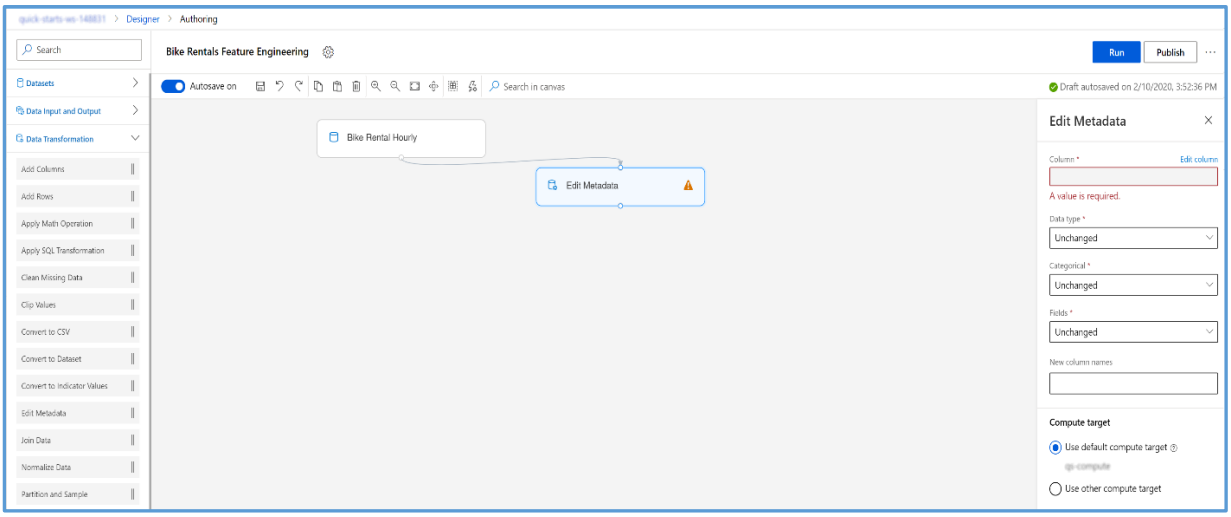
Note: If you are facing difficulties in accessing pop-up windows or buttons in the user interface, please refer to the Help section in the lab environment.

Task 4: Select columns in the dataset

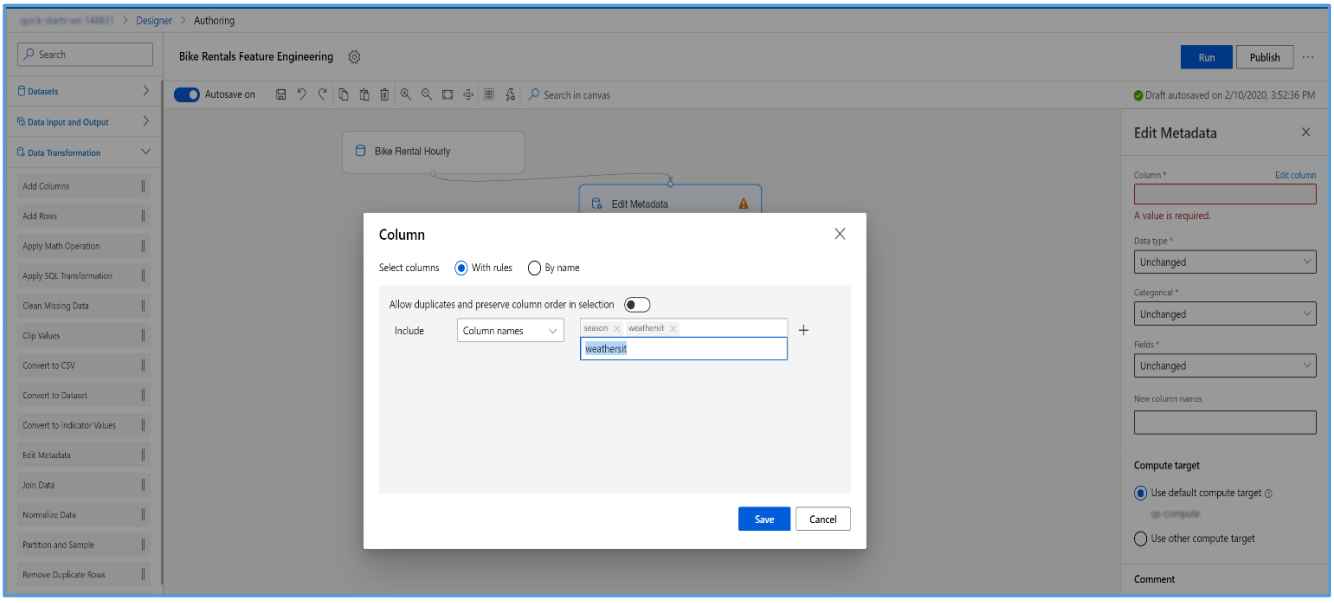
1. Drag and drop on the canvas, the available Bike Rental Hourly dataset under the **Datasets** category on the left navigation.



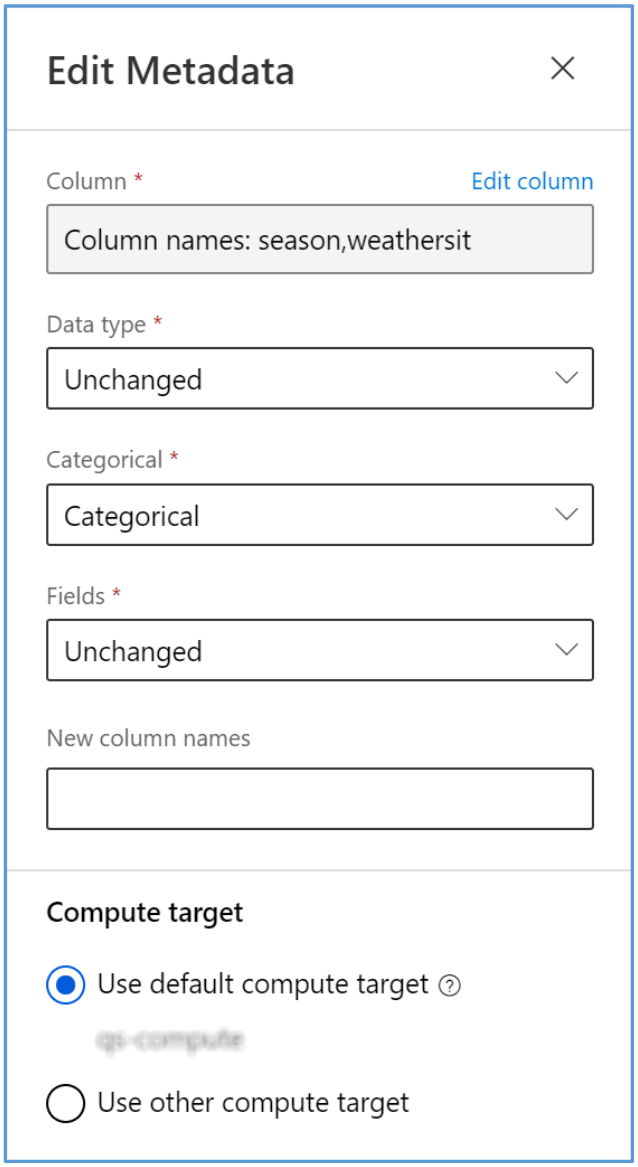
2. Under the **Data transformation** category drag and drop the Edit metadata module.



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

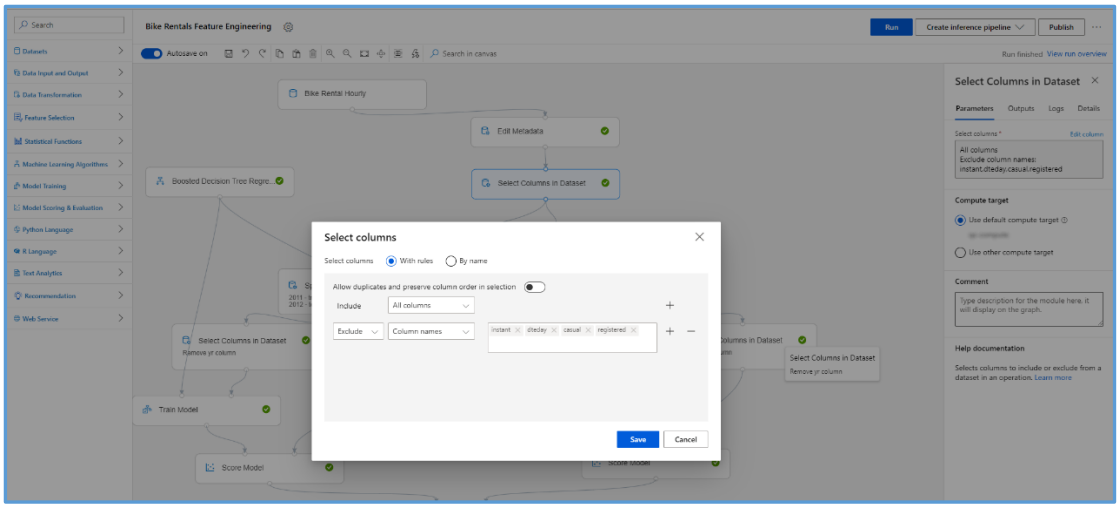


4. Configure the Edit metadata module by selecting the `Categorical` attribute for the two columns.



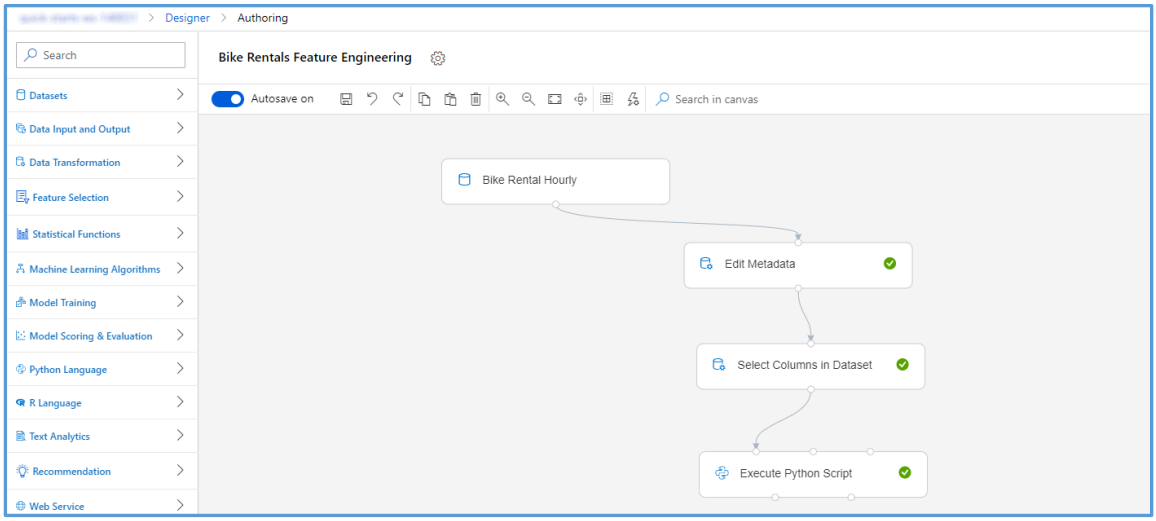
5. Next, use the **Select columns in Dataset** module under the **Data transformation** category and configure it as follows:

- 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.
```

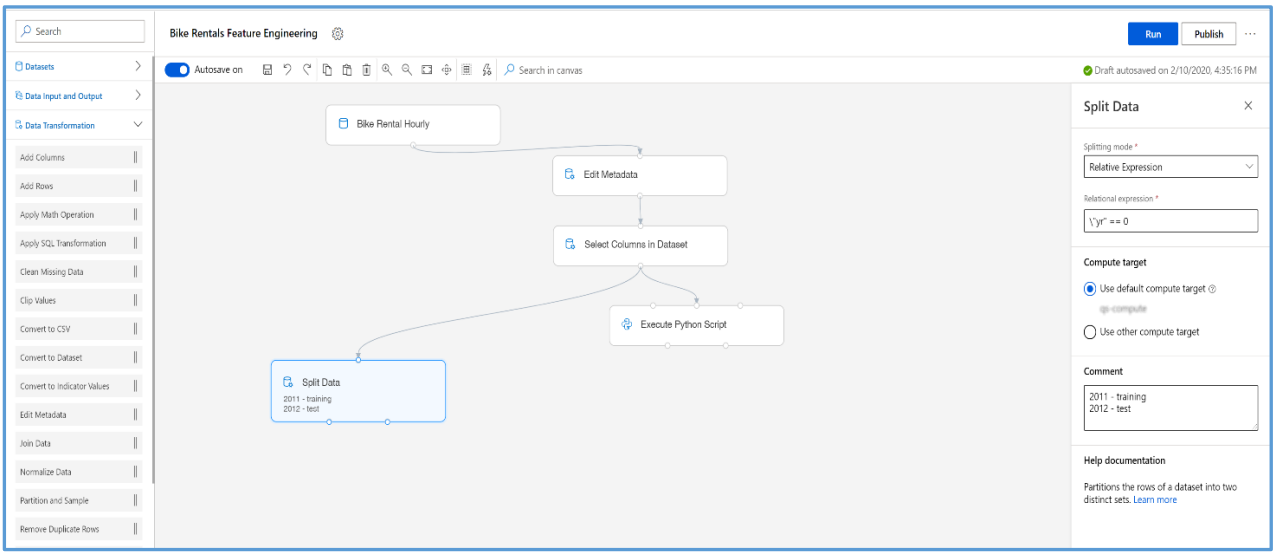
```
# imports up here can be used to
import pandas as pd
import numpy as np

# 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.

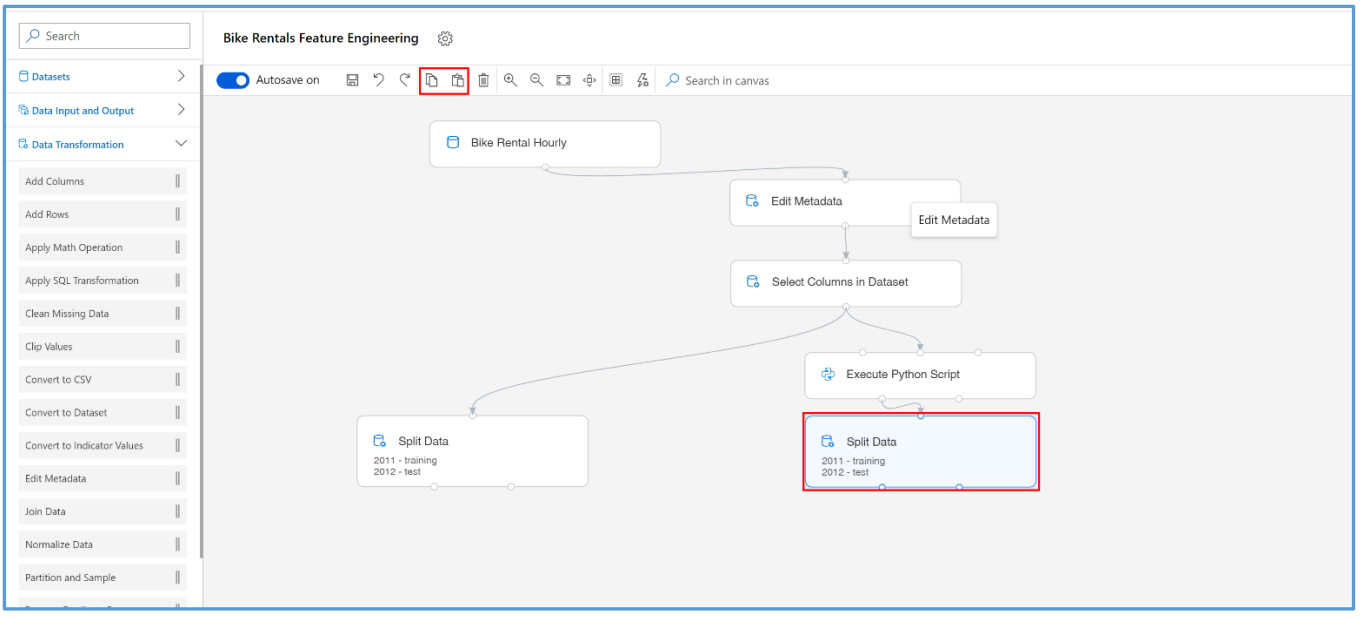
Task 5: Split data into train and test datasets

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`

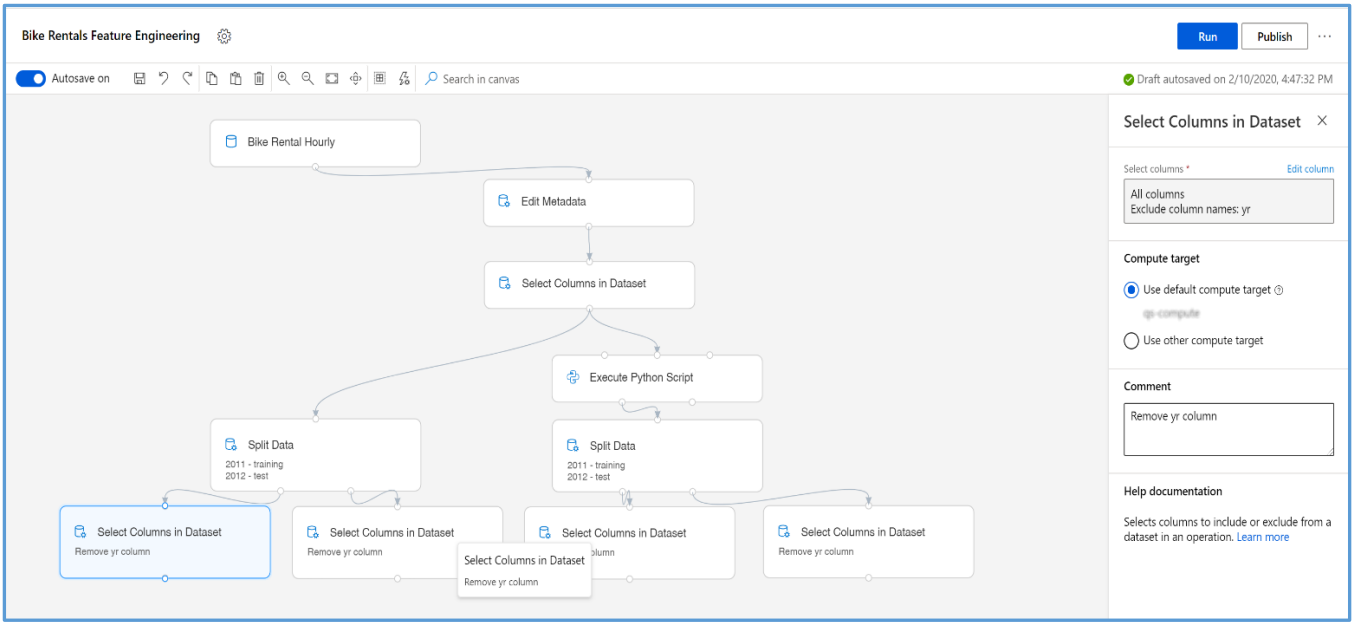


2. Select the **Split Data** module block, and use the menu buttons to Copy and Paste it on the canvas. Connect the second one to the output of the Python Script execution step, which is the featured B set.

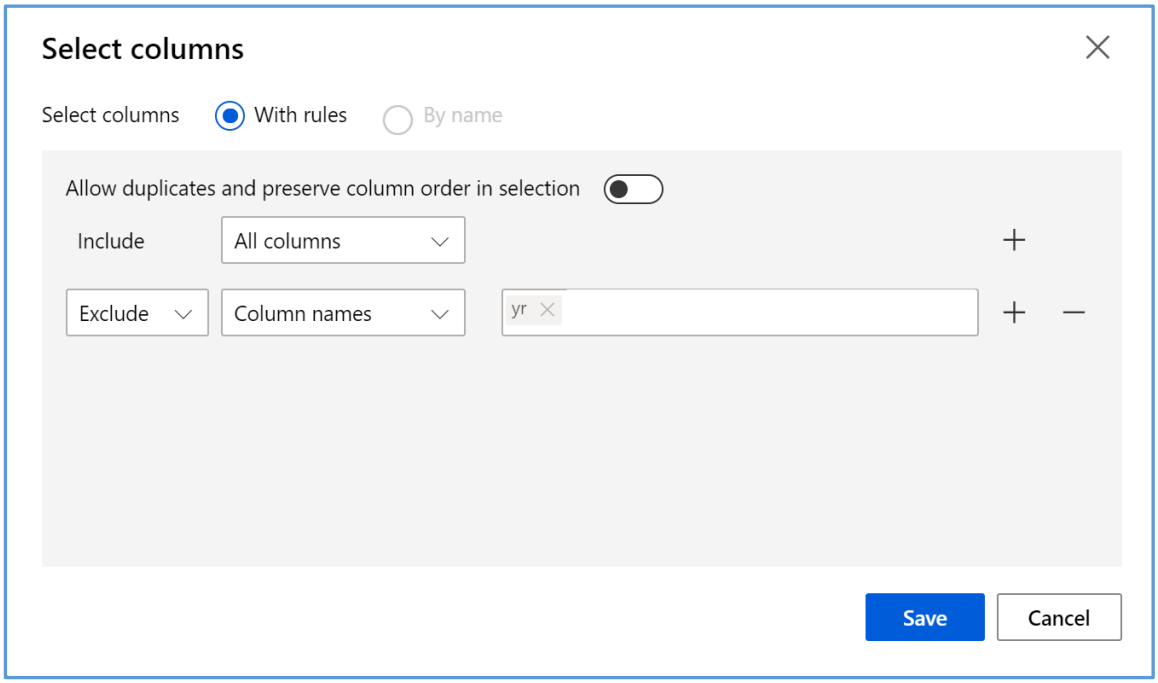


Task 6: Select columns from the test and training resulted sets

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.

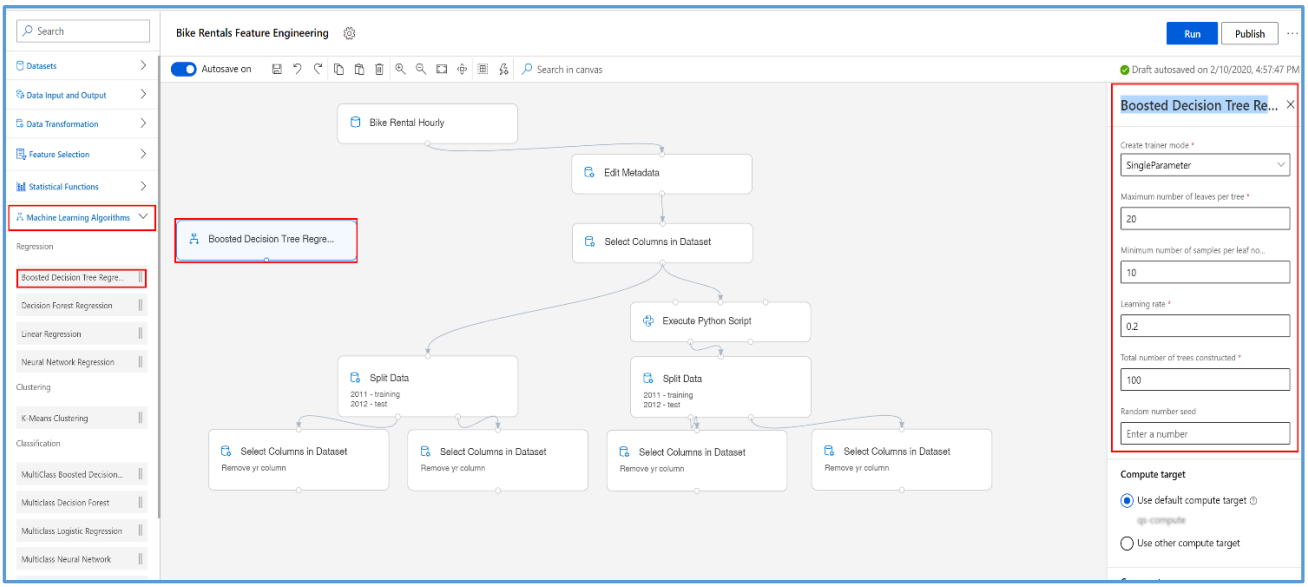


2. Use the following structure for the columns field in each module:



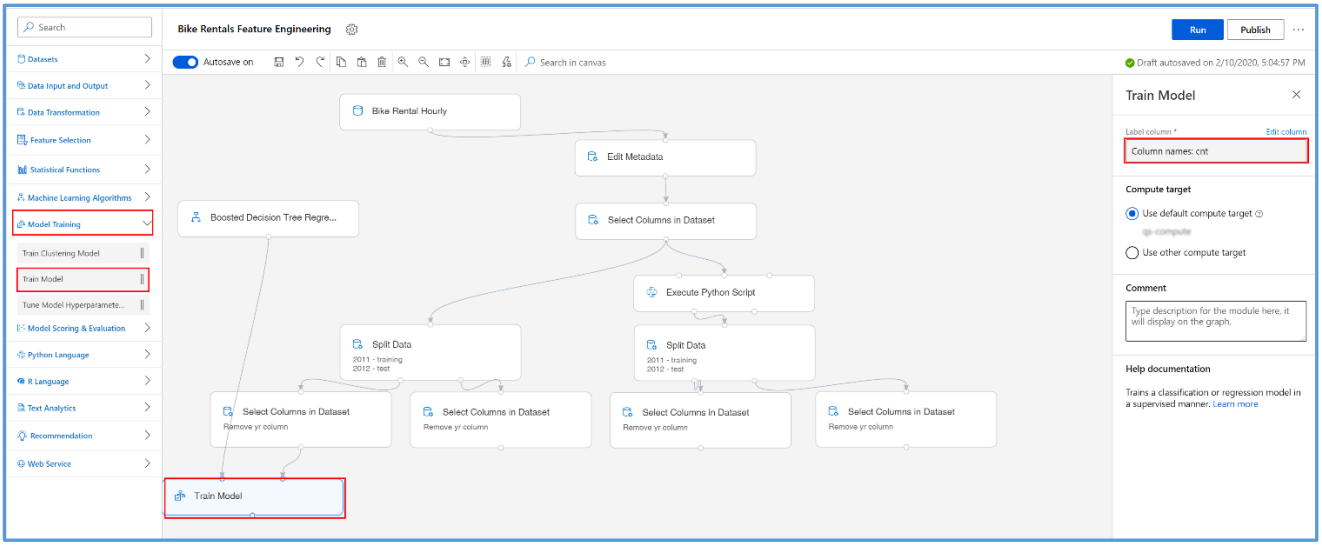
Task 7: Create the regression model

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.

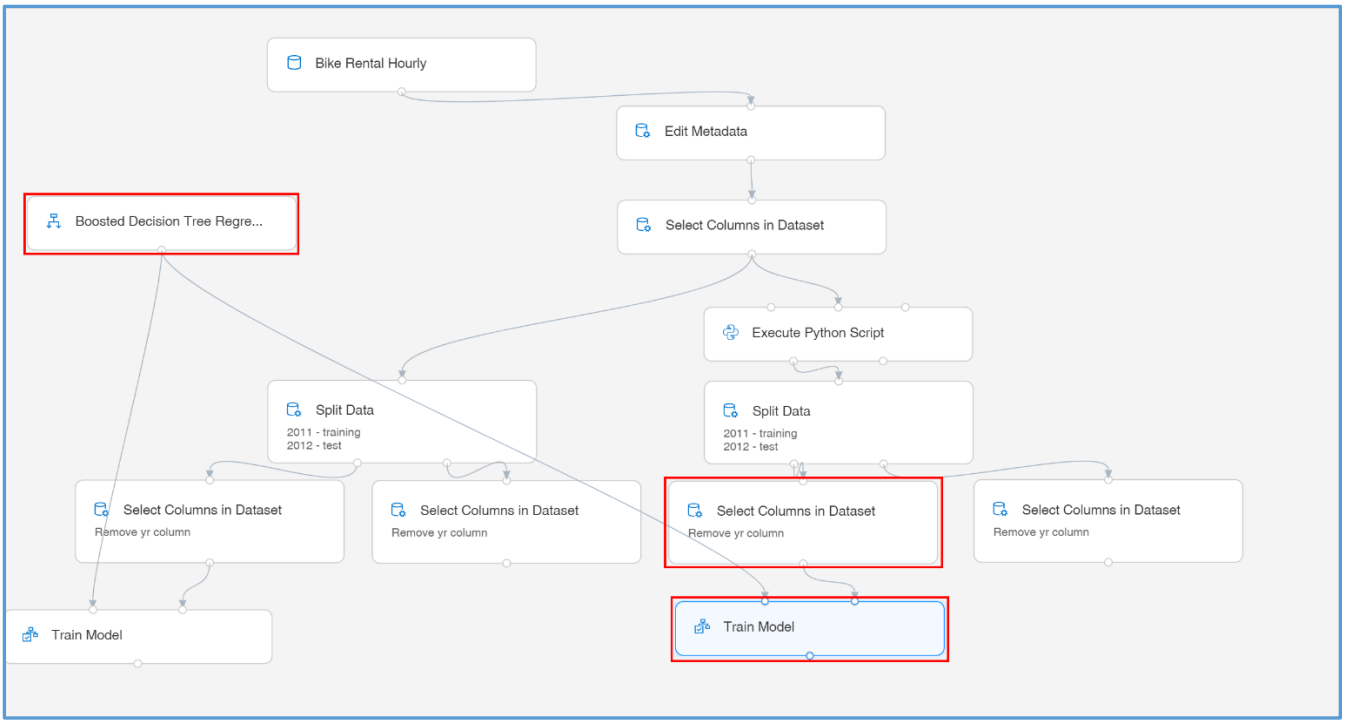


2. Next, use the **Train model** module under the **Model-training** category and enter the `cnt` column in the **Label column** field.

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.



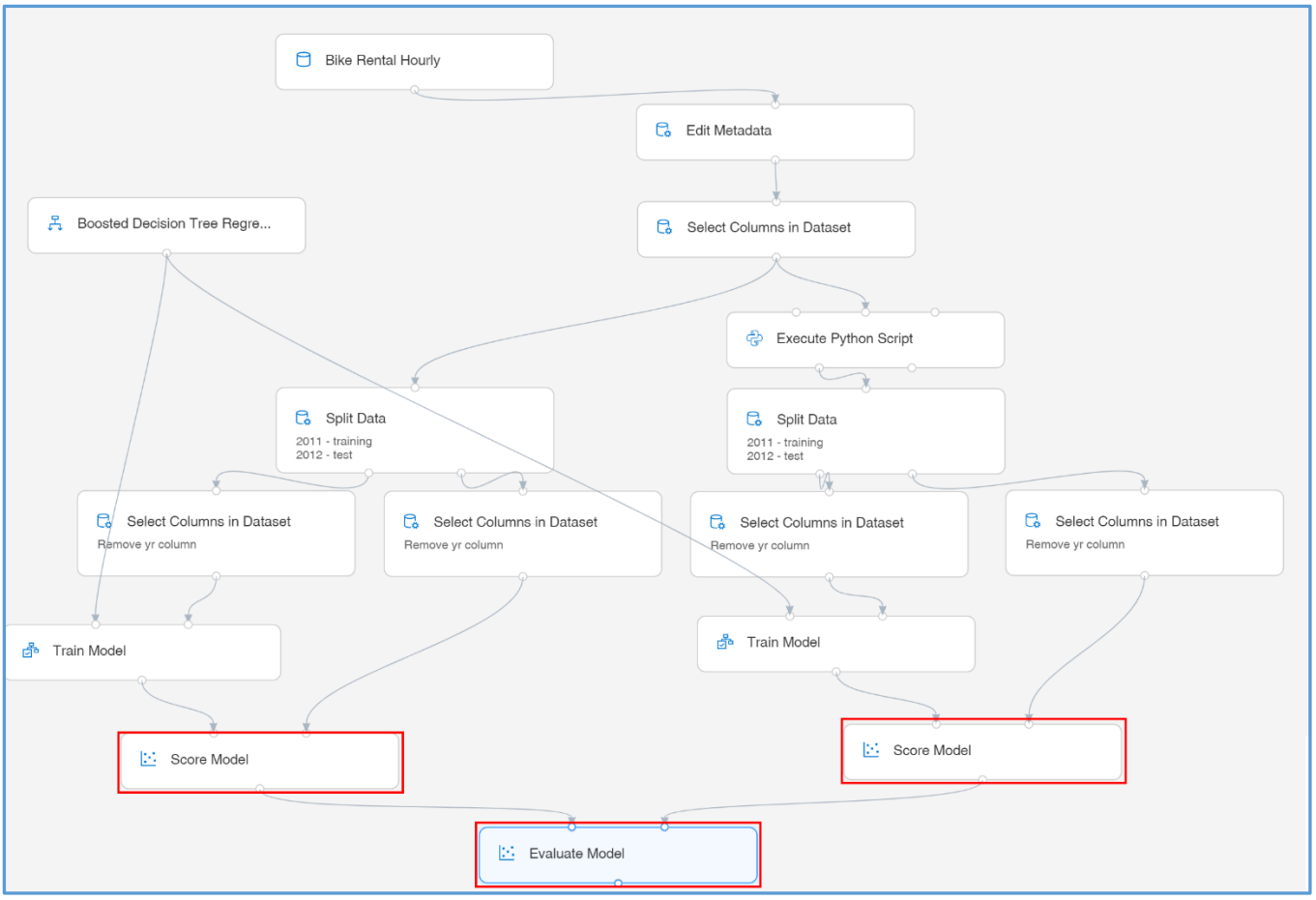
4. Use the exact same configuration on the right branch that uses the output from the Python Script.



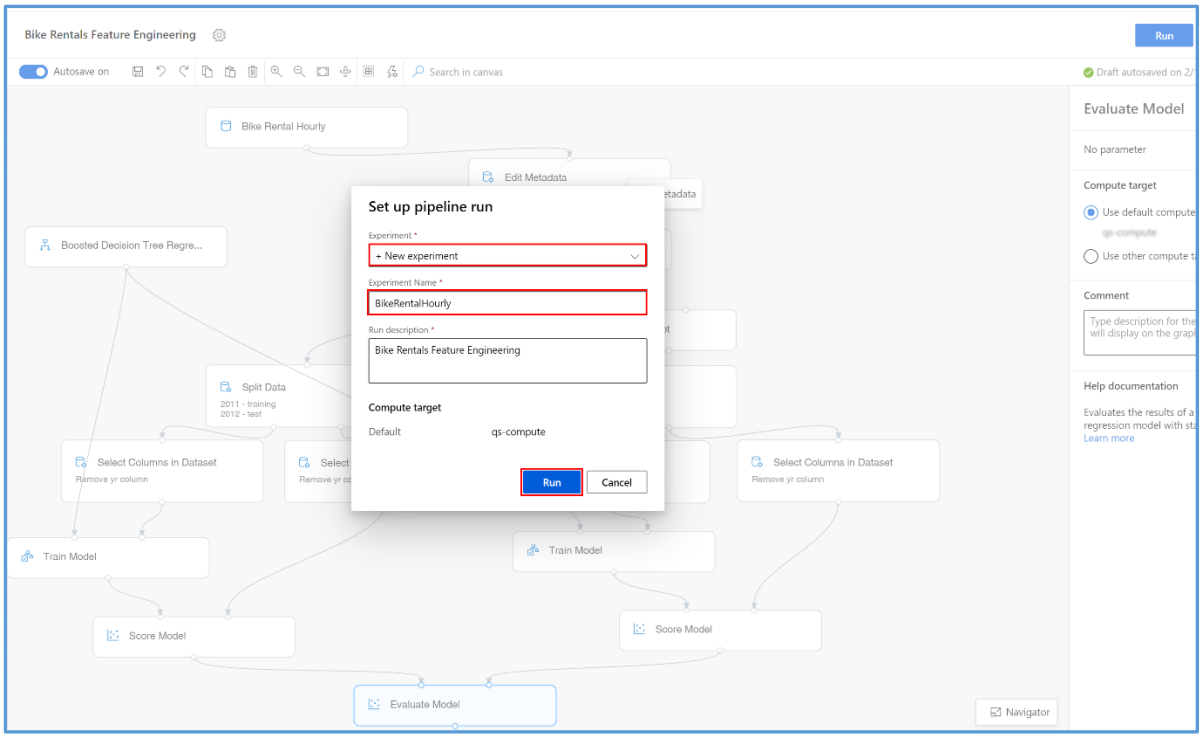
Task 8: Evaluate and score models

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.



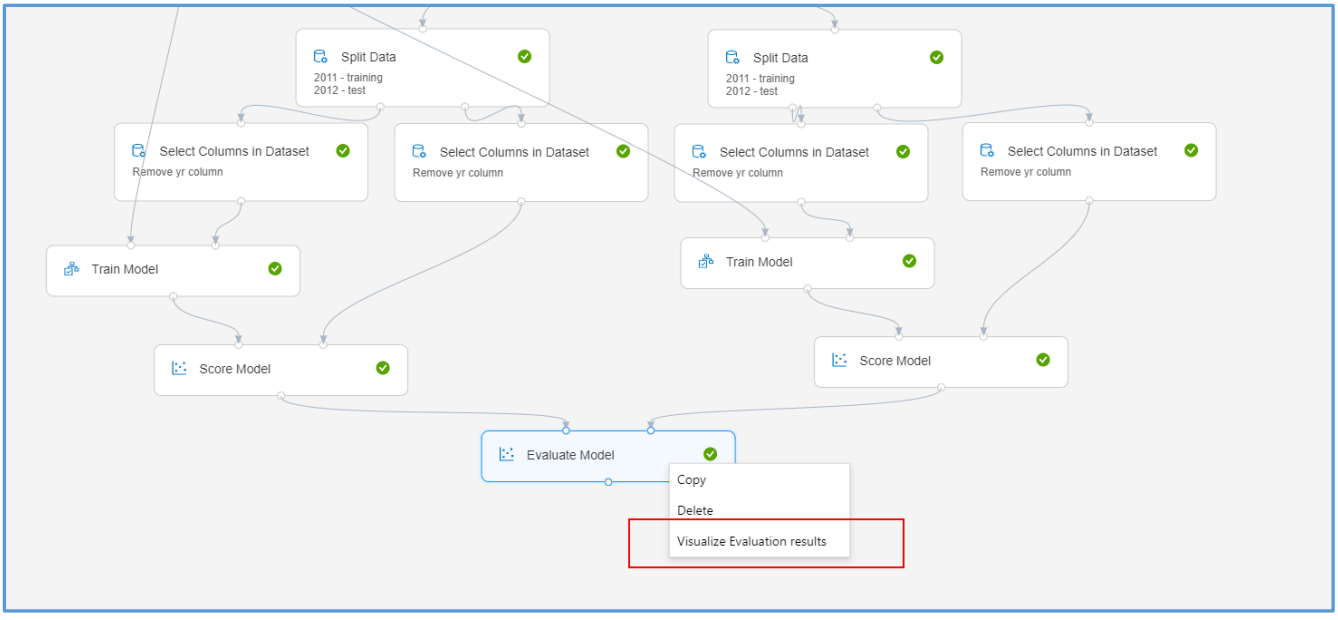
3. Select **Submit** to open the Setup pipeline run editor. In the Setup pipeline run editor, select **Experiment, Create new** and provide New experiment name: **BikeRentalHourly**.



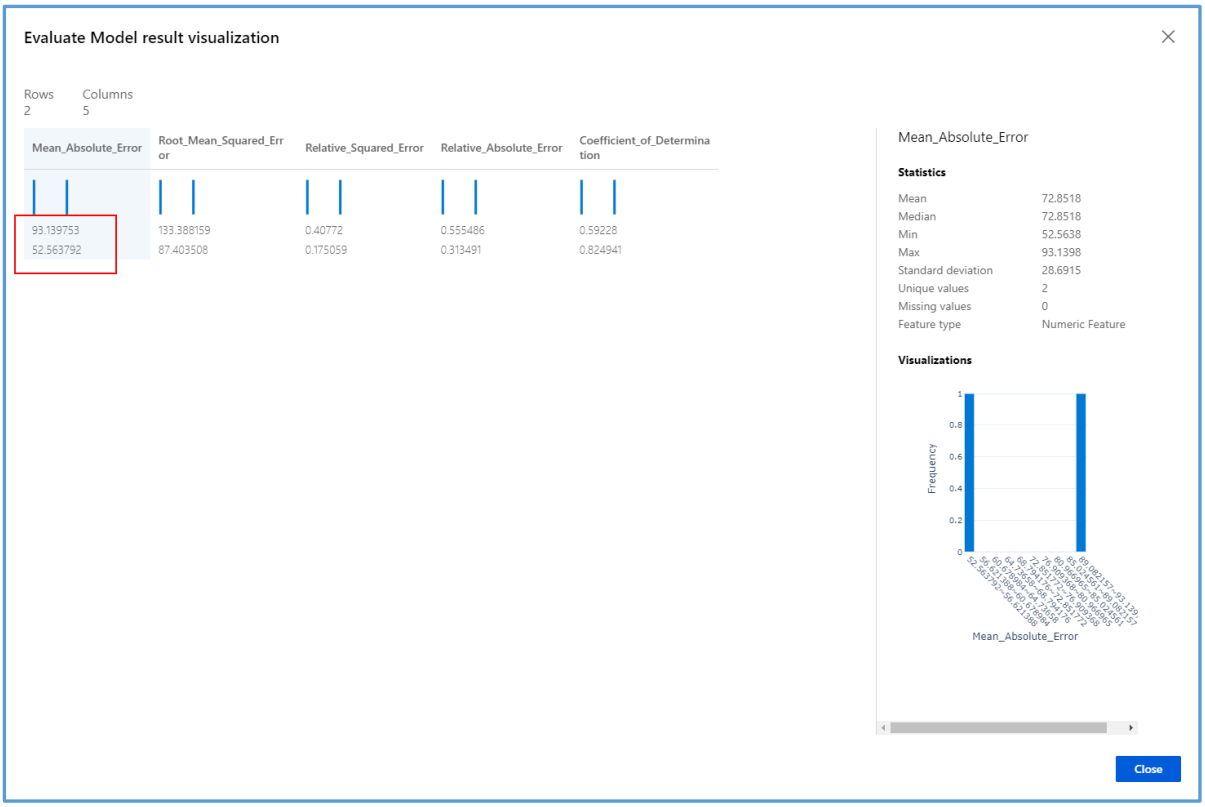
Please note that the button name in the UI is, changed from **Run** to **Submit**.

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.

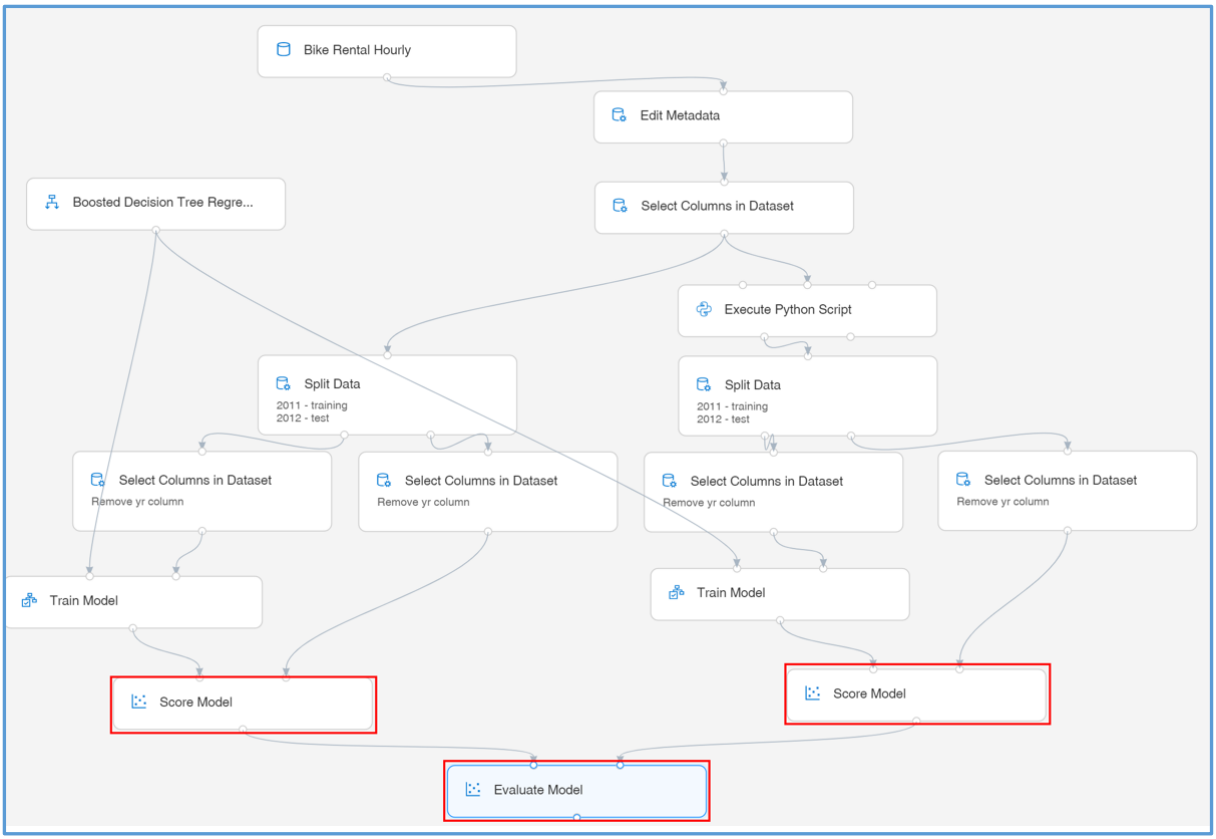
It is remarkable how, using simple feature engineering to derive new features from the existing data set, a new context was created that allowed the model to better understand the dynamics of the data and hence, produce a better prediction.

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.

Chapter 16: Walkthrough – Engineer and Select features

- Azure pipeline needs to be, created and we need to select default compute target.
- Upload a file from web and register the same as a dataset.
- Dataset added to the designer canvas.
- **Edit the metadata** associated with the dataset. “season” & “weathersit” column is edited as Categorical data.
- **Select columns in dataset** specifies the list of columns that we wish to use further down in the pipeline.[“instant”,” dteday”,” casual” and “registered” columns need to be excluded]
- Actual Feature engineering step is performed via **Execute Python script** task.[Created 12 new columns within the dataset containing the number of bikes that were rented in each of the previous 12 years]. Dataset is being, enriched with 12 columns.
- **Split data** is, used to split the data into Training and Test dataset using the rule \”yr” == 0 [“yr” == 0 indicates Year-2011 & “yr” == 1 indicates Year–2012]. Thus 2011 would be Training and 2012 would be Test data.
- One **Split data** task has been, applied to the output derived from the Python script, which has additional data points. **[Feature engineered dataset]**
- Second **Split data** task has been applied to the output derived from Select Column in Dataset**[Original dataset]**
- Output from each of the **Split data** task is being divided into 4 [2 from each split data task] datasets [Training and Test data from each Split data]. This goes into each of the **“Select Columns in Dataset”** task with the “yr” column being, excluded.
- Training data from Original dataset and Feature engineered data set goes into the “**Train Model**” Task using the “**Boosted decision tree regression**” algorithm.
- Output from the **Train Model** and Test Data [“**Select columns in dataset**”] for the two datasets are being, passed to the “Score Model” task.
- Output from each of the “Score Model” task is being, fed into the “Evaluate Model” task.



Chapter 17: Data Drift

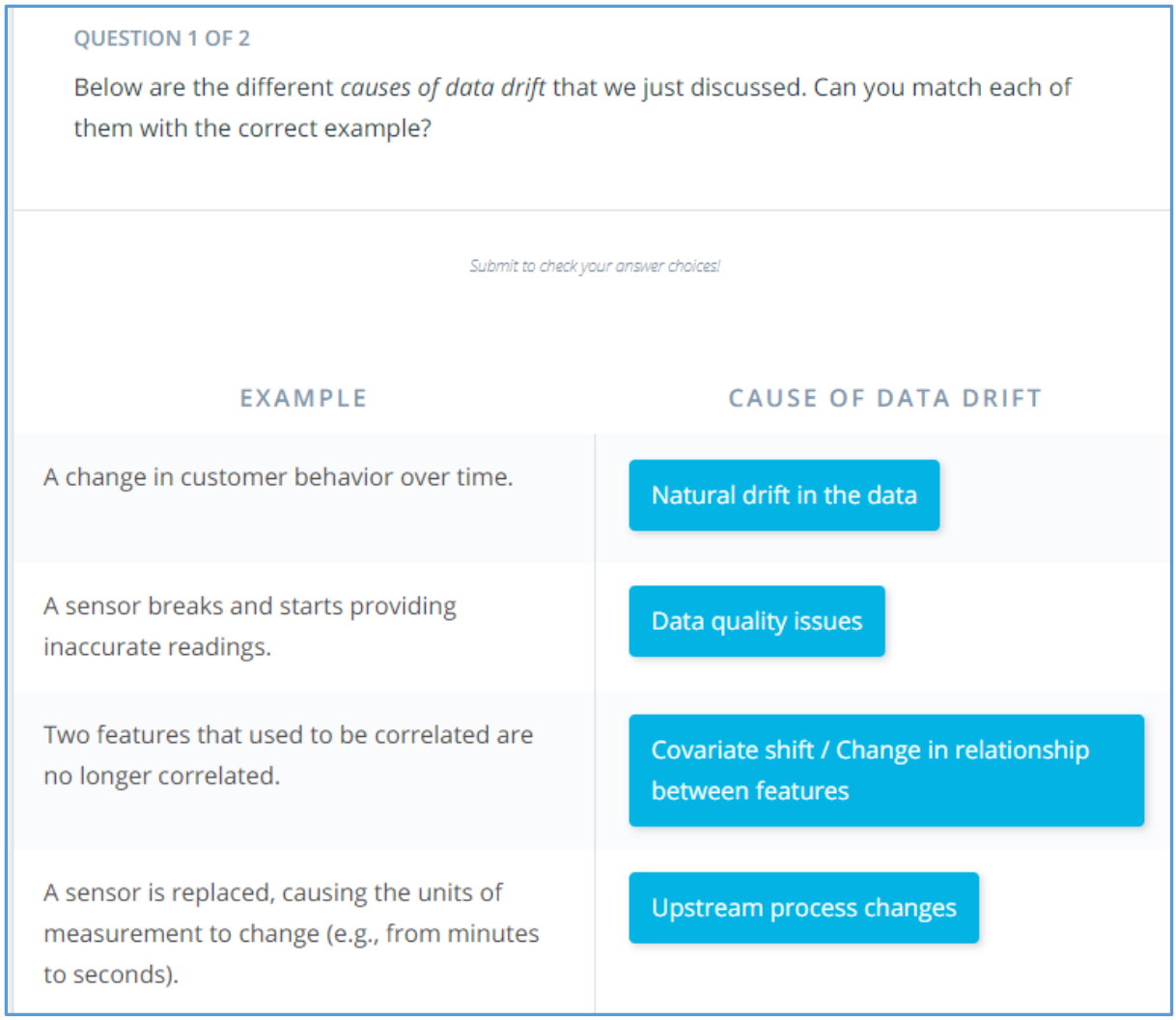
As we mentioned earlier, **data drift** is change in the input data for a model. Over time, data drift causes degradation in the model's performance, as the input data drifts farther and farther from the data on which the model was, trained.

Due to data drift, model accuracy degrades over time. Monitoring, identifying, detecting data drifts helps, on one hand, detect the associated model performance issues and on the other hand triggers the retraining process to avoid them.

There are multiple causes of data drift:

- Change in the Upstream process
- Data quality issues
- Natural data drift in the data

Dataset monitors provide alerts that help in detecting the data drifts over our datasets in time.



Remember, the process of monitoring for data drift involves:

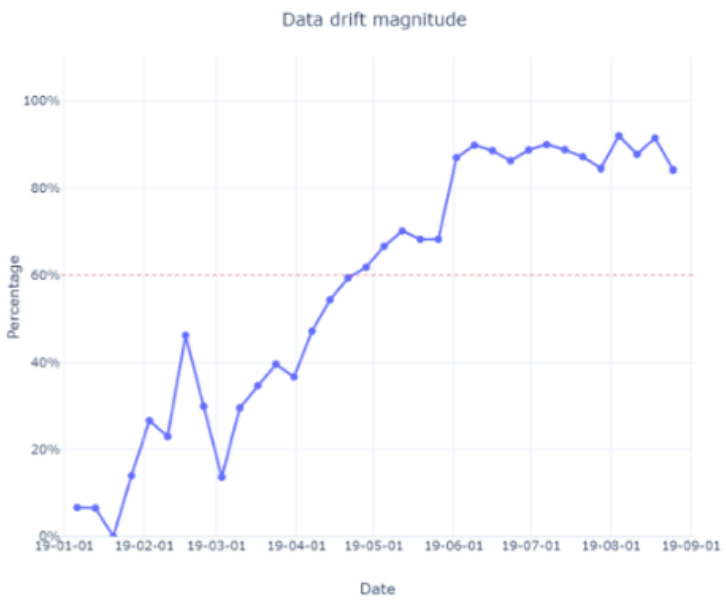
- Specifying a **baseline dataset** – usually the training dataset
- Specifying a **target dataset** – usually the input data for the model
- Comparing these two datasets over time, to monitor for differences
  Here are a couple different types of comparisons you might want to make when monitoring for data drift:

- **Comparing input data vs. training data.** This is a proxy for model accuracy; that is, an increased difference between the input vs. training data is likely to result in a decrease in model accuracy.
- **Comparing different samples of time series data.** In this case, you are checking for a difference between one time, period and another. For example, a model trained on data collected during one season may perform differently when given data from another time of year. Detecting this seasonal drift in the data will alert you to potential issues with your model's accuracy.
- **Perform analysis on past data.**

When looking at a data drift magnitude chart, 0% indicates that the datasets are identical, while 100% indicates that Azure Machine Learning can completely tell the two datasets apart.

QUESTION 2 OF 2

Here's the graph of data-drift magnitude that we looked at in the video:



In this example, a baseline January dataset was compared with a dataset containing all 2019 data. Which of these statements about the graph are true?

(Select all that apply.)

- ☐ The baseline vs. target datasets show a lot of difference in January
- ☑ The baseline vs. target datasets show very little difference in January
- ☑ The baseline vs. target datasets show a lot of difference in August
- ☐ The baseline vs. target datasets show very little difference in August

Chapter 18: Model Training basics

In this section, we will get into more detail on the steps involved in training a model, and then we will get some hands-on practice with the process in the upcoming lab.

Remember that our ultimate goal is to produce a model we can use to make predictions. Put another way, we want to be able to give the model a set of input features, X, and have it predict the value of some output feature, y.

Parameters and Hyperparameters

When we train a model, a large part of the process involves learning the values of the *parameters* of the model. For example, earlier we looked at the general form for linear regression:

$$y = B_0 + B_1 * x_1 + B_2 * x_2 + B_3 * x_3 ... + B_n * x_n$$

The coefficients in this equation, $B_0 ... B_n$, determine the intercept and slope of the regression line. When training a linear regression model, we use the training data to figure out what the value of these *parameters* should be. **Thus, we can say that *a major goal of model training is to learn the values of the model parameters.***
In contrast, some model parameters are, *not* learned from the data. These are, called **hyperparameters** and their values are set before training. Here are some examples of hyperparameters:

- The number of layers in a deep neural network
- The number of clusters (such as in a k-means clustering algorithm)
- The learning rate of the model

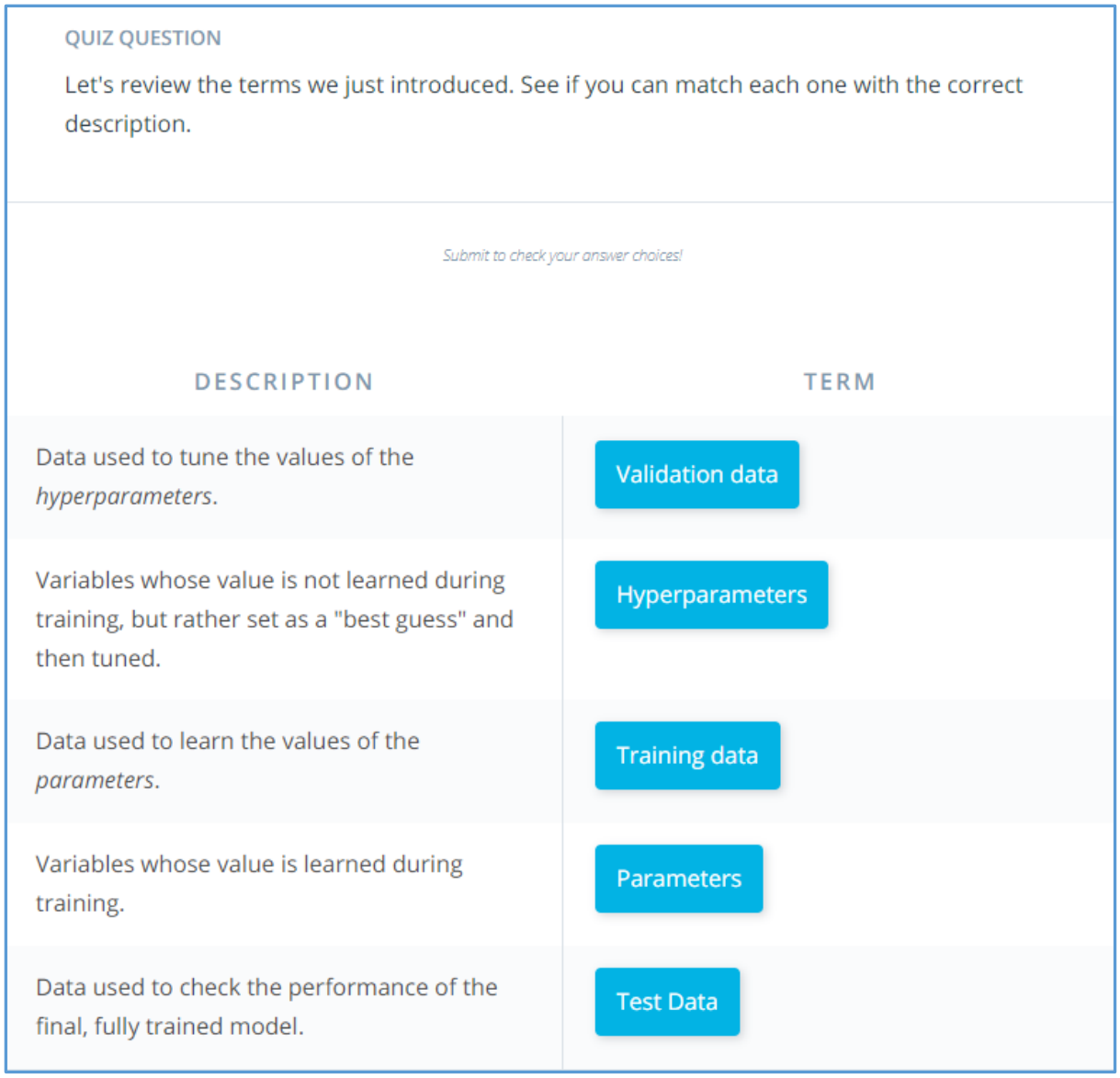
We must choose some values for these hyperparameters, but we do not necessarily know what the best values will be prior to training. Because of this, a common approach is to take a best guess, train the model and then tune adjust or *tune* the hyperparameters based on the model's performance.

Splitting the Data

As mentioned in the video, we typically want to split our data into three parts:

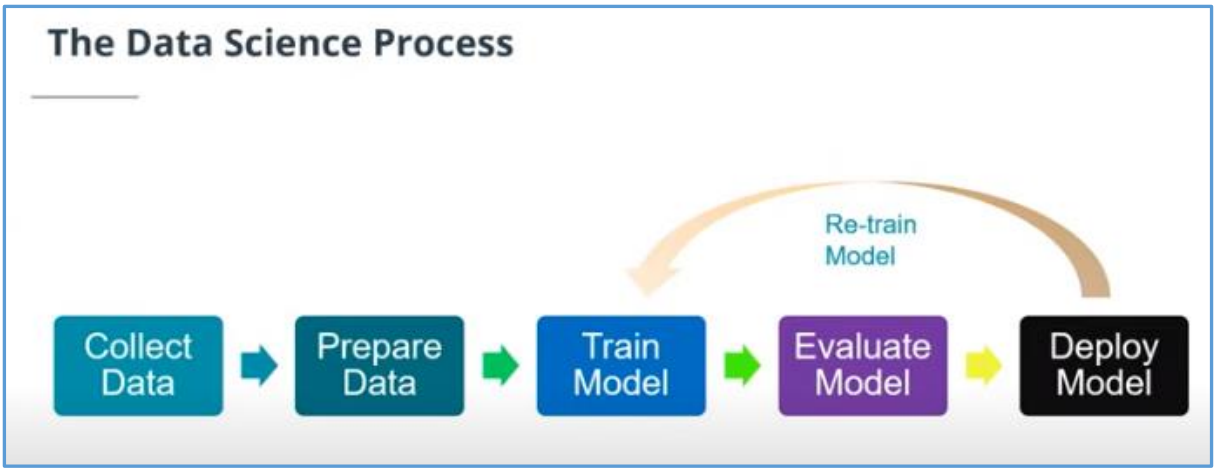
- **Training data**
- **Validation data**
- **Test data**

We use the **training data** to learn the values for the *parameters*. Then, we check the trained model's performance on the **validation data** and *tune* the hyperparameters until the model performs well with the validation data. For instance, perhaps we need to have more or fewer layers in our neural network. We can adjust this hyperparameter and then test the model on the validation data once again to see if its performance has improved.
Finally, once we believe we have our finished model (with both parameters and hyperparameters optimized), we will want to do a final check of its performance—and we need to do this on some fresh **test data** that we did not use during the training process.



Chapter 19: Model Training in Azure Machine Learning

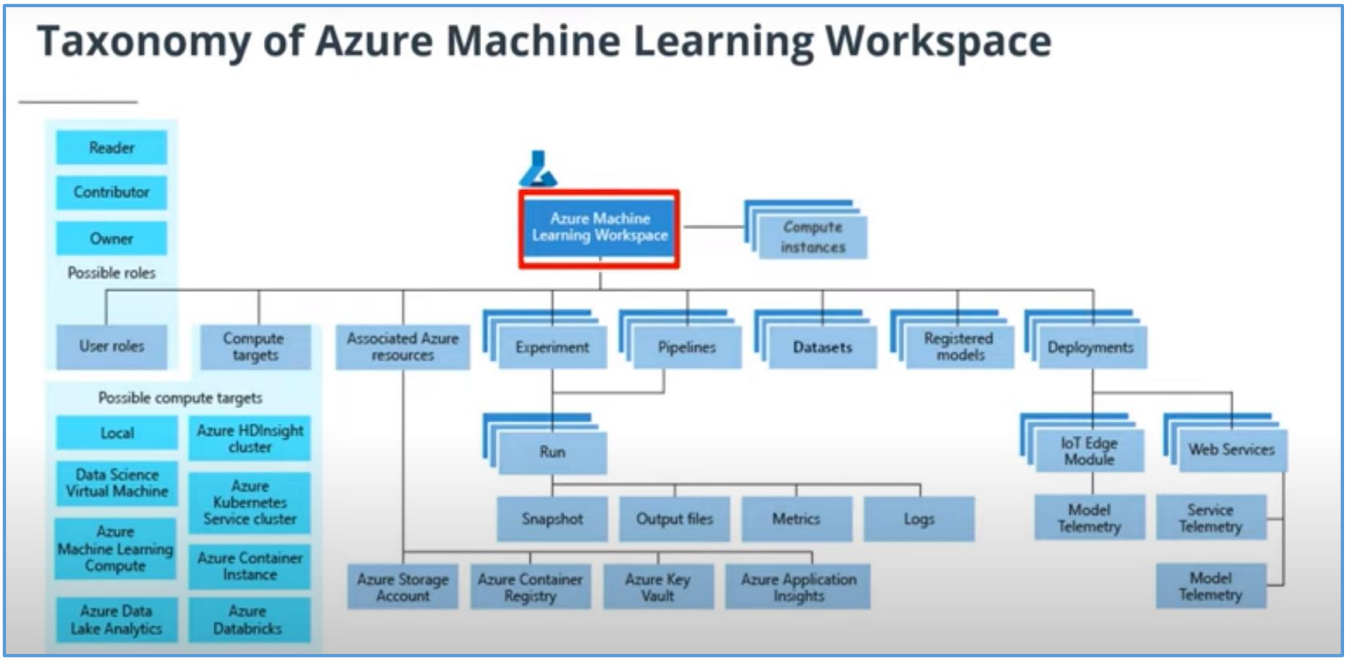
How Azure ML does provides support to the entire data science process?



Azure Machine Learning service

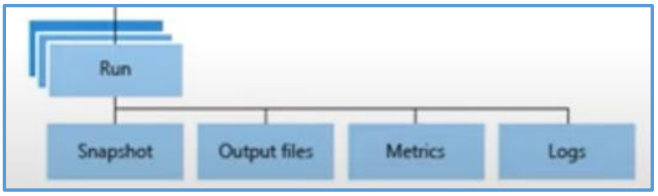
- Azure Machine Learning service provides a comprehensive environment to implement model training processes, giving you a centralized place to work with all the artifacts involved in the process.

Taxonomy of Azure Machine Learning

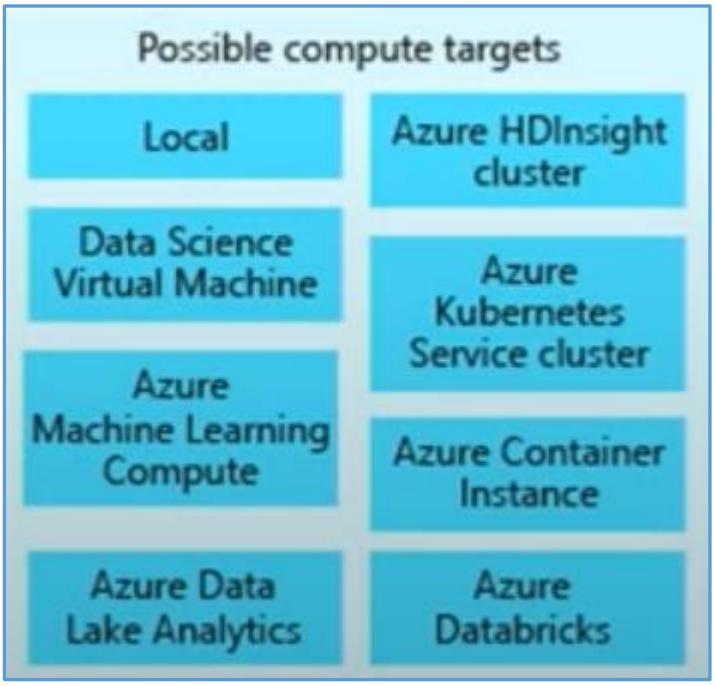


- **Azure ML workspace is the first thing that we create when we start working with Azure ML.**
- **Create a bunch of Compute instances.**
- **Datasets are key component in the data preparation and data transformation process, which makes data available for ML training processing.**
- **Experiment is set of task that we run within Azure ML. It is a container, which helps us, to group various artifacts, which are, related to ML process.**

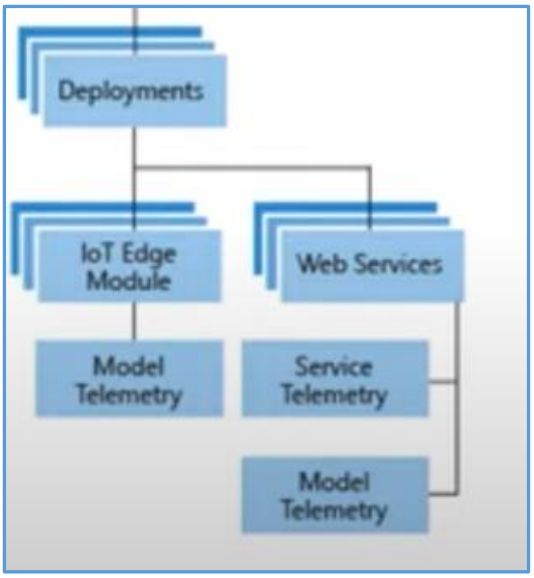
- **One of the most important artifact is, RUN. Run is a process that is executed in one of the compute resource made available by Azure ML. Examples of Run includes – Training of Models, Validation of Models, Feature engineering using extensive Python codes to create features.**
- **Every run will output a set of artifacts like – Files, Metrics, Snapshot of data and Logs.**



- **Our ML processes can run on a variety of environment like – Local environment, Remote environment – Native Azure ML Compute, Azure Data Lake analytics, Azure HDInsight, Spark clusters, Kubernetes Clusters.**



- **Once the model is, created then it needs to be, registered within Model registry.**
- **Finally, we can deploy the models. Deployments can be in the form of Web services or they can go to other types of environments like IoT Edge.**
- **Once we have models in operations, it is very important to collect Telemetry data about the model themselves or about the services that are, exposed using those models or through the training process themselves.**
  [Telemetry: The process of recording and transmitting the readings of an instrument]



QUIZ QUESTION

Below are some of the components of Azure Machine Learning. Can you match each of them with the correct description?

(We'll get practice working with all of these in the labs, but it will help if you get some general familiarity with them now.)

Model telemetry | Run

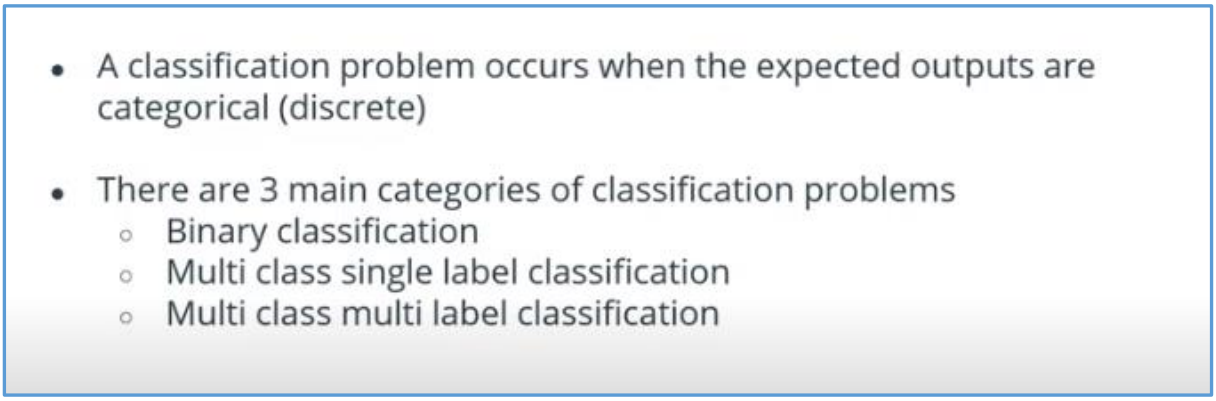
DESCRIPTION	COMPONENT
The centralized place for working with all the components of the machine learning process.	Workspace
A container that helps you organize the model training process.	Experiment
A service that provides snapshots and versioning for your trained models.	Model registry
A cloud-based workstation that gives you access to various development environments, such as Jupyter Notebooks.	Compute instance

Chapter 20: Training Classifiers

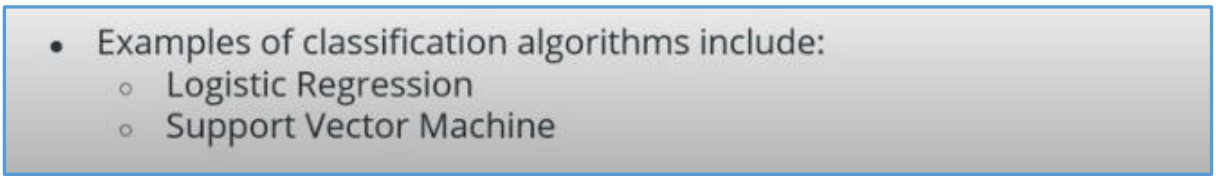
As we described in the last lesson, two of the main types of *supervised learning* are **classification** and **regression**. In this section, we will get some practice training both of these types of models. First, let us discuss the concepts in more detail—starting with **classification**.

*In a **classification** problem, the outputs are categorical or discrete.*

For example, you might want to classify emails as *spam* or *not spam*; each of these is a discrete category.



- **Binary Classification -** In case of Binary classification the **prediction output is a Binary value** [True/False] or [Zero / One]. Some of the most difficult ML tasks fall under this category. Example: Anomaly detection or Fraud detection in case of banking transactions.
- **Multi class single label classification – Output is not just binary output but your class contains multiple values.** Instead of choosing between two classes, we need to choose between three or more classes. **Example:** Recognition of written numbers where the numbers are between, 0 to 9, and we need to classify each image being one of the 10 digits that we are using for numbering. **[Multiple potential outputs but output belongs to a single class]**
- **Multi class, multi label classification –** We have multiple categories and the outputs can belong to more than one. **[Multiple potential outputs and output can belong to multiple classes]**



QUIZ QUESTION

As we described in the video, there are three main types of classification problem. Can you mark which type each of these examples belongs to?

Submit to check your answer choices!

EXAMPLE	TYPE
Classify an image as one (and only one) of five possible fruits.	Multi-class single-label classification
Classify medical test results as "positive" or "negative".	Binary classification
Classify music as belonging to multiple groups (e.g., "upbeat", "jazzy", "pop").	Multi-class multi-label classification

Chapter 21: Training Regressors

Now let us turn to *regression*. The main distinction that sets a regression problem apart from a classification problem is the form of the output:
*In a **regression** problem, the output is numerical or continuous.*
A classic example would be a problem in which you are, given data, concerning houses and then asked to predict the price; this is a regression problem because *price* is a continuous, numerical output.

- A regression problem occurs when the expected outputs are numerical (continuous)
- There are 2 main categories of regression problems
  - Regression to arbitrary values
  - Regression to values between 0 and 1

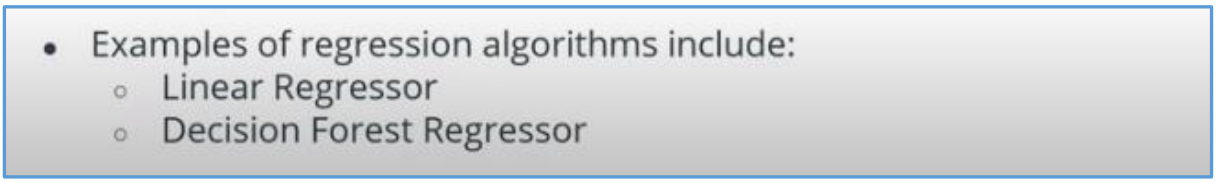
In case of fraudulent transactions, when we worked with Classification problem we categorized the transactions in category of Zero [Not a fraudulent transaction] and One [Fraudulent transaction]

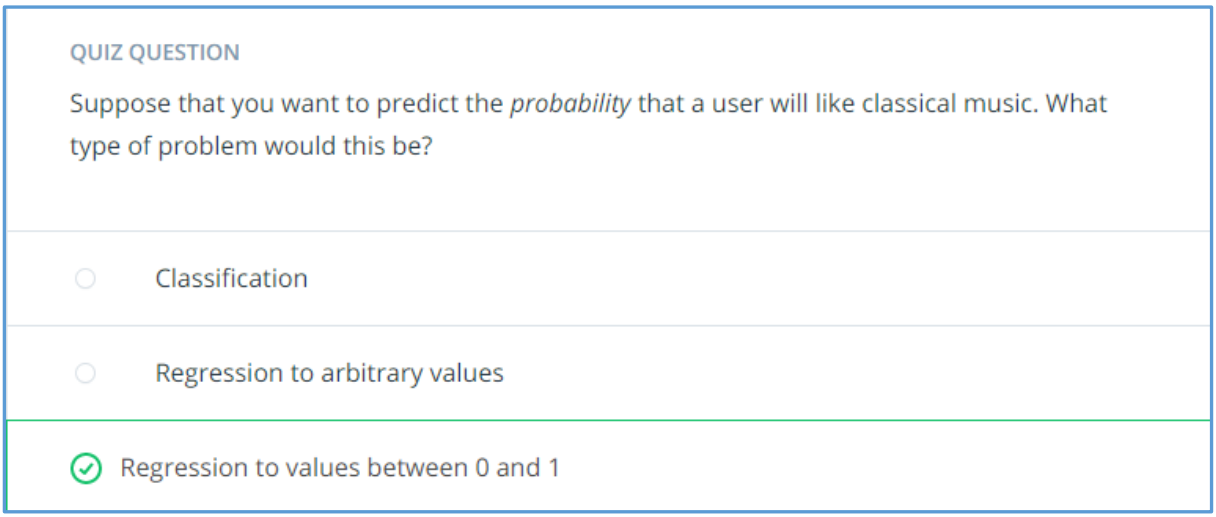
However if the same problem is considered under Regression then we consider the same in terms of probability.

Probability of a transaction to be fraudulent = p

Thus, probability of a transaction to be non-fraudulent = 1- p

Overall probability is 1.





Chapter 22: Evaluating Model Performance

It is not enough to simply train a model on some data and then assume that the model will subsequently perform well on future data. Instead, as we have mentioned previously, we need to split off a portion of our labelled data and reserve it for evaluating our model's final performance. This helps in deciding on the predictive power and the accuracy of the model. We refer to this as the *test dataset*.

*The **test dataset** is a portion of labelled data that is, split off and reserved for model evaluation.*

If a model learns, to perform well with the training data, but performs poorly with the test data, then there may be a problem, that we will need to address before putting our model out into the real world. In practice, we will also need to decide what metrics we will use to evaluate performance, and whether there are any, particular thresholds that the model needs to meet on these metrics in order for us to decide that it is "good enough."

When splitting the available data, it is important to preserve the *statistical properties* of that data. This means that the data in the training, validation, and test datasets need to have similar statistical properties as the original data to prevent bias in the trained model.
For different models, we would use different metrics for measuring the performance.

Criteria also needs to be set for Evaluation. [Primary metric for evaluation] In addition, threshold needs to be set. [Meet or exceed the threshold before it is, considered to be, good]

QUIZ QUESTION

A researcher has collected datasets from two neighboring cities in order to develop a model that predicts the sale price of a house based on various features. Which of the following approaches would be best for this researcher?

- Split the data by city, so that data for the first city is used to train the model, and data for the second city is used to evaluate the model.
- Use all of the data for training the model, as this will result in greater power and the best possible model optimization.
- ✓ Split the data randomly, so that some houses from each city are included in both the training dataset and the test dataset.

Splitting the data up randomly will help ensure that the two datasets are statistically similar. For instance, if there are many highly expensive houses in one city and many inexpensive houses in the other, a random split will ensure that all housing types are included both, during training and testing.

Chapter 23: Confusion Matrices

Suppose that we have trained a simple binary classification model: Given an image, this model will indicate whether it is a picture of a cat or a picture of a dog. How can we evaluate our model's performance? What is a good metric for doing so?

Let us first consider what it means for the model to perform well. If the model tells us an image has a dog in it *and that image actually has a dog*, we would say it performs well. Similarly, if it says that the image has a cat *and it actually has a cat* that would also be good. To help us think about the problem, we can construct a table that shows all of the possibilities:

		Actual class	
		Cat	**Dog**
Predicted class	**Cat**	**Correct Cat**	Incorrect Cat
	Dog	Incorrect Dog	**Correct Dog**

As you can see, the columns here represent the *actual class*—that is, whether an image *actually* has a dog or a cat. The rows represent the *predicted class*—that is, whether the model concludes that an image has a dog or a cat. When the predicted class matches the actual class (e.g., the model says the image has a cat and the image does indeed have a cat), this is a *correct* classification.

The key is to look at the diagonals. **If the upper left and lower right cells are high relative to the others, then the model is making more correct classifications than incorrect classifications:**

		Actual class	
		Cat	**Dog**
Predicted class	**Cat**	**4**	1
	Dog	2	**3**

Whereas if the upper right and lower left cells are comparatively higher, the model is making more, incorrect classifications:

		Actual class	
		Cat	**Dog**
Predicted class	**Cat**	2	**3**
	Dog	**4**	1

This type of table is, called a **confusion matrix**. A confusion matrix gets its name from the fact that it is easy to see whether the model is getting *confused* and misclassifying the data. You will often see the confusion matrix represented in a more general, abstract form that uses the terms *positive* and *negative*:

		Actual class	
		Positive	**Negative**
Predicted class	**Positive**	**True Positives (TP)**	False Positives (FP)
	Negative	False Negatives (FN)	**True Negatives (TN)**

- **True positives** are the *positive* cases that are *correctly* predicted as *positive* by the model
- **False positives** are the *negative* cases that are *incorrectly* predicted as *positive* by the model

- **True negatives** are the *negative* cases that are *correctly* predicted as *negative* by the model
- **False negatives** are the *positive* cases that are *incorrectly* predicted as *negative* by the model

QUESTION 1 OF 3

We can use a table like this to organize our model's output. Suppose that we give the classifier test data containing 10 images, and the results are as follows:

		Actual class	
		Cat	**Dog**
Predicted class	**Cat**	**5**	1
	Dog	1	**3**

How well has the model performed?

- ○ The model performed perfectly—there were no misclassified images at all
- ✓ The model performed well, though not perfectly—there were a few misclassified images
- ○ The model performed poorly—most of the images were misclassified
- ○ The model performed very badly—it misclassified *all* of the images

The model correctly classified 8 images (5 cat images and 3 dog images), while it incorrectly classified 2 images (1 dog image and 1 cat image).

QUESTION 2 OF 3

OK, here is a different set of results:

		Actual class	
		Cat	**Dog**
Predicted class	**Cat**	**0**	4
	Dog	6	**0**

How well has the model performed this time?

- The model performed perfectly—there were no misclassified images at all
- The model performed quite well, though not perfectly—there were a few misclassified images
- The model performed poorly—most of the images were misclassified
- ✓ The model performed very badly—it misclassified *all* of the images

Model classified none of the images correctly.

QUESTION 3 OF 3

Which one of the following is **incorrect** about confusion matrices?

- The sum of TN and FN tells us the number of cases predicted as negative by the model
- ✓ The sum FP and TP tells us the number of actual positive cases in the dataset
- The sum of FP and TN tells us the number of actual negative cases in the datasets
- The sum of TP and TN tells us the number of cases correctly predicted by the model

Chapter 24: Evaluation Metrics for Classification

The most important evaluation matrix for a classifier is confusion matrix. In case of a binary classification problem the output has cardinality of two.[0/1 or True/False]

- Confusion Matrix

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

- **True positives** are the *positive* cases that are *correctly* predicted as *positive* by the model
- **False positives** are the *negative* cases that are *incorrectly* predicted as *positive* by the model
- **True negatives** are the *negative* cases that are *correctly* predicted as *negative* by the model
- **False negatives** are the *positive* cases that are *incorrectly* predicted as *negative* by the model

$$\text{Accuracy: } \frac{TP+TN}{TP+FP+FN+TN}$$

Accuracy is the proportion of correct predictions.
Correct Predictions = TP + TN
Total Predictions = TP+TN+FP+FN

$$\text{Precision: } \frac{TP}{TP+FP}$$

$$\text{Precision} = \frac{\text{Number of True Positives}}{\text{Number of Predicted Positives}}$$

- Confusion Matrix

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

Precision is the proportion of positive cases that were correctly, identified. This is also, called Positive predictive value.
Denominator = Overall Positive predictions (TP+FP)
Numerator = Actual True Positives

$$\text{Recall: } \frac{TP}{TP+FN}$$

$$\text{Recall} = \frac{\text{Number of True Positives}}{\text{Number of Actual Total Positives}}$$

- Confusion Matrix

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

Recall is the proportion of actual positive cases that were correctly, identified. This is also, called Sensitivity or True positive rate.

Denominator = Overall Actual Positives (TP+FN)
Numerator = Actual True Positives

$$\text{F1 Score: } 2 * \frac{Precision * Recall}{Precision + Recall}$$

Measures the balance between Precision and Recall.

In order to get the exact view of the classification algorithm we need to use the metrics in pairs like Accuracy & Precision or Accuracy & Recall.

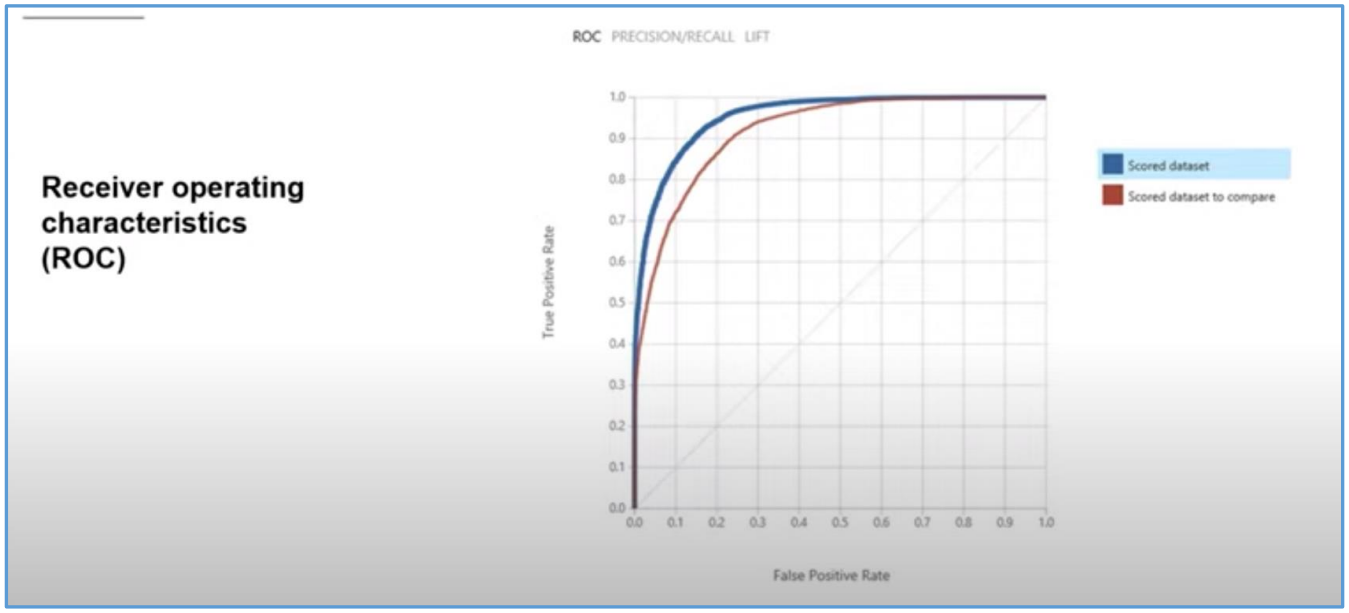
QUESTION 1 OF 2

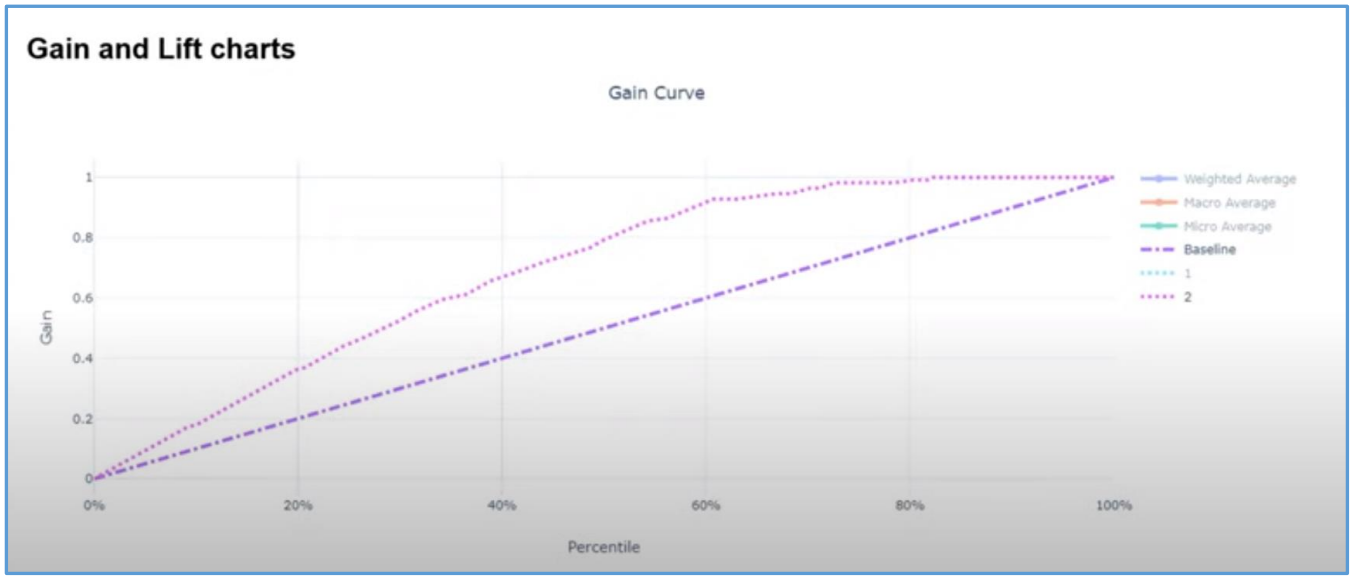
As we just saw, the confusion matrix gives us several different metrics we can use to measure the performance of our model. See if you can remember which formula is used to calculate each metric.

Submit to check your answer choices!

FORMULA	METRIC
$\frac{TP+TN}{TP+FP+FN+TN}$	Accuracy
$\frac{TP}{TP+FP}$	Precision
$\frac{TP}{TP+FN}$	Recall
$2 * \frac{Precision*Recall}{Precision+Recall}$	F1 score

Model Evaluation Charts





Gain and Lift chart deals with Rank ordering the prediction probabilities, and they actually measure how much better you are, expected to perform by using the classifier as opposed to random guessing.

AUC (Area under Curve) ROC (Receiver Operating Characteristics) Curve

The AUC ROC curve is a graph, which shows the performance of a classification model at all thresholds. ROC is a probability curve and AUC represents degree of separability. ROC plots the following parameters:

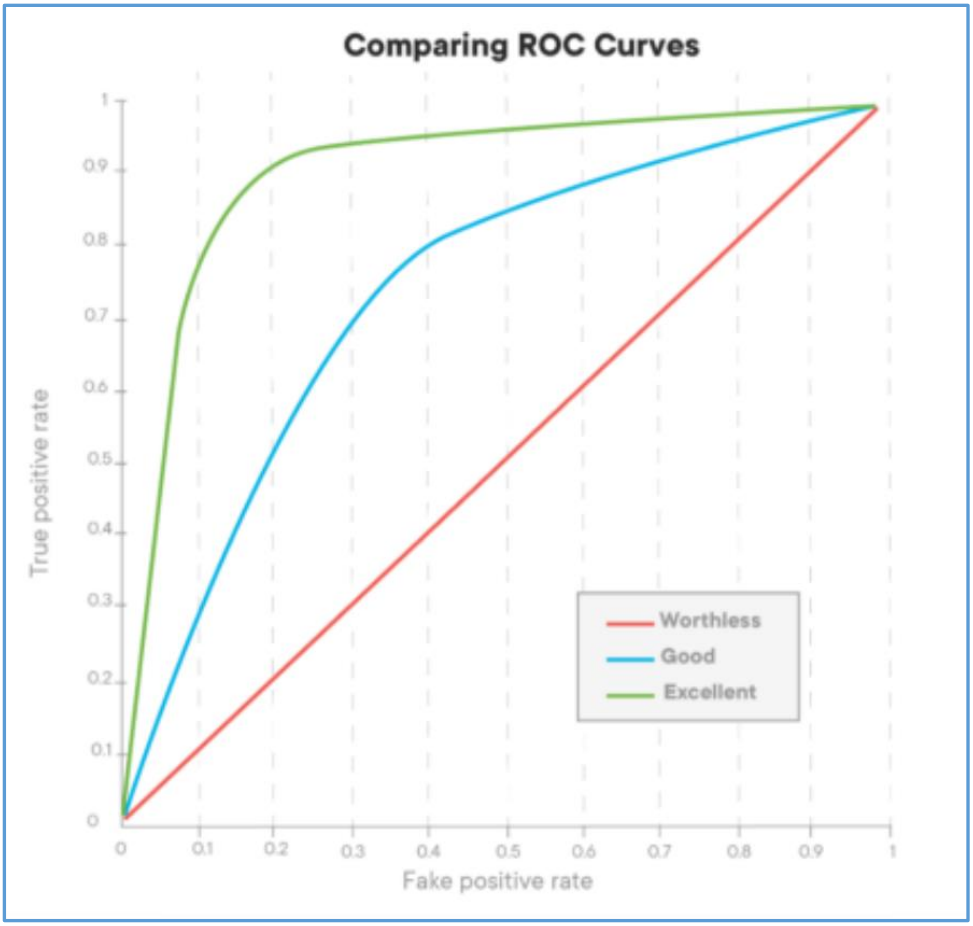
- **True Positive Rate (TPR)**, also known as recall or sensitivity, which was, explained in the previous section.

$$TPR = \frac{TP}{TP + FN}$$

- **False Positive Rate (FPR)**, also known as Fall-out, the ratio of the false positive predictions compared to all values that, are actually negative.

$$FPR = \frac{FP}{FP + TN}$$

Both the TPR and FPR are within the range [0, 1]. The curve is the FPR vs TPR at different points in the range [0, 1]. The best performing classification models will have a curve similar to the green line in the graph below. The green line has the largest Area under the Curve. The higher the AUC, the better your model is performing. A classifier with only 50–50 accuracy is realistically no better than randomly guessing, which makes the model worthless (red line).



An AUC of 0.5 indicates random guessing, while an AUC of 1.0 indicates perfect classification.

QUESTION 2 OF 2

In the **Receiver Operating Characteristics (ROC)** chart that we just looked at, the **Area Under the Curve (AUC)** for the diagonal line is 0.5. What does this indicate?

- A classifier that performs perfectly
- A classifier that performs moderately well
- ✓ A classifier that performs no better than random guessing
- A classifier that is always (or almost always) wrong

Chapter 25: Prelaunch Lab

Prelaunch your lab environment.

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

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

Chapter 26: Evaluation Metrics for Regression

If you recall, classification yields discrete outputs (e.g., cat vs dog or positive vs. negative), while regression yields continuous, numerical outputs (e.g., 3.229, 23 minutes, $17.78). Not surprisingly then, we need a different set of metrics for evaluating regression models. Let us have a look.

- **Root Mean square Error (RMSE)** – Measures the square root of the average of the squared differences between the predictions and the true values.

RMSE is the most popular evaluation metric used in regression problems. It follows an assumption that error are unbiased and follow a normal distribution. Here are the key points to consider on RMSE:

1. The power of 'square root' empowers this metric to show large number deviations.
2. The 'squared' nature of this metric helps to deliver more, robust results, which prevents cancelling the positive and negative error values. In other words, this metric aptly displays the plausible magnitude of error term.
3. It avoids the use of absolute error values, which is highly undesirable in mathematical calculations.

4. When we have more samples, reconstructing the error distribution using RMSE is, considered to be, more reliable.
5. RMSE is highly affected by outlier values. Hence, make sure you have removed outliers from your data set prior to using this metric.
6. As compared to mean absolute error, RMSE gives higher weightage and punishes large errors.

RMSE metric is, given by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Where, N is Total Number of Observations.

- **Mean absolute Error (MAE)** – Measures the average of the absolute difference between each prediction and true value

  The Mean Absolute Error measures the average of the absolute difference between each ground truth and the predictions.
  Whether the predictions is 10 or 6while the ground truth was 8, the absolute difference is 2.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n} |y_j - \hat{y}_j|$$

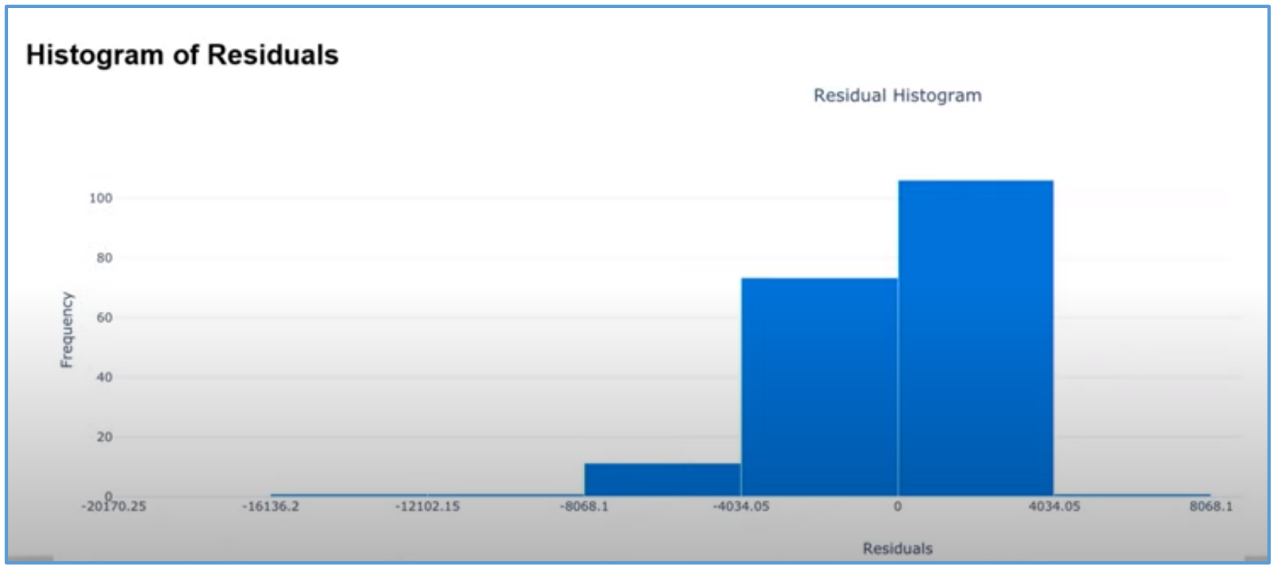
  The y^hat is the prediction but again, the order does not matter since we are calculating the absolute contrast.

- **R-Squared/Coefficient of determination** – The *coefficient of determination*, is a measure that provides information about the goodness of fit of a model. In the context of regression, it is a statistical measure of how well the regression line approximates the actual data. It is therefore important when a statistical model is used either to predict future outcomes or in the testing of hypotheses. There are a number of variants (see comment below); the one presented here is widely used

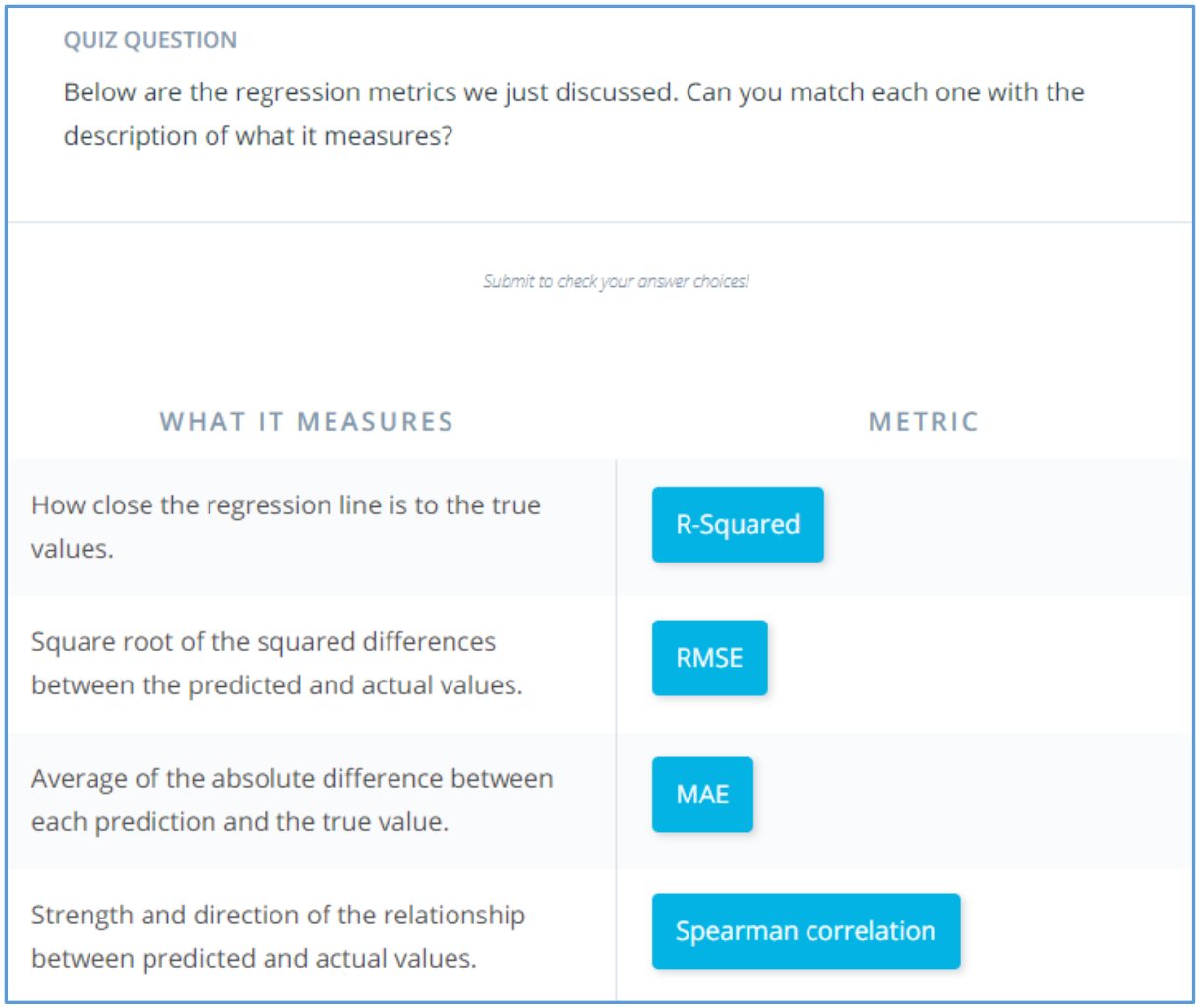
$$R^2 = 1 - \frac{\text{sum squared regression (SSR)}}{\text{total sum of squares (SST)}},$$
$$= 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}.$$

- *The* sum-squared regression *is the sum of the residuals squared, and the* total sum of squares *is the sum of the distance the data is away, from the mean all squared. As it is a percentage, it will take values between 0 and 1.*
- *R^2=1 - All the variation in the y values is accounted for by the x values*
- *R^2=0.83 - 83% of the variation in the y values is accounted for by the x values*

- **Spearman correlation** – Measures the strength and direction of correlation.



When a model has Low Bias, the histogram should approach Normal distribution.



Chapter 27: Lab – Train and Evaluate a Model

Lab Overview

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

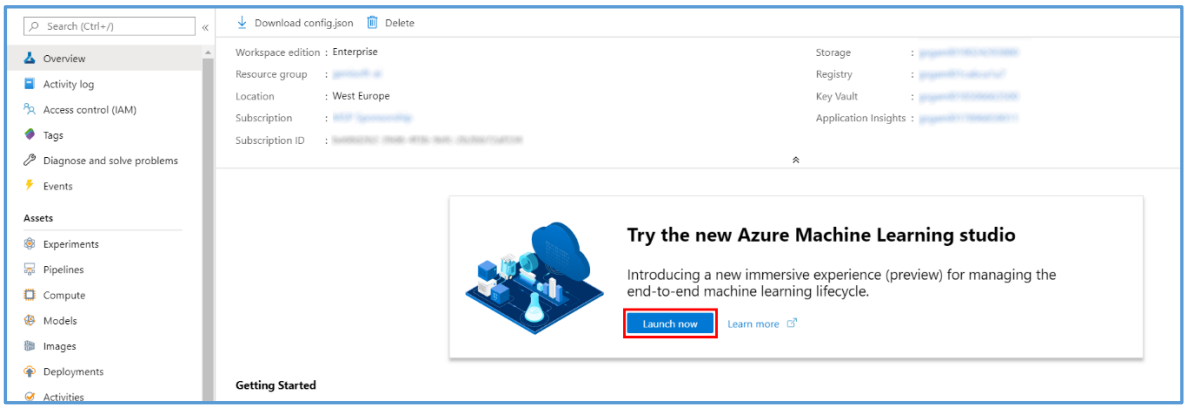
In this lab, we will be using the Flight Delays data set that is, enhanced with the weather data. Based on the enriched dataset, we will learn to use the Azure Machine Learning Graphical Interface to process data, build, train, score, and evaluate a classification model to predict if a, particular flight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

Exercise 1: Register Dataset with Azure Machine Learning studio

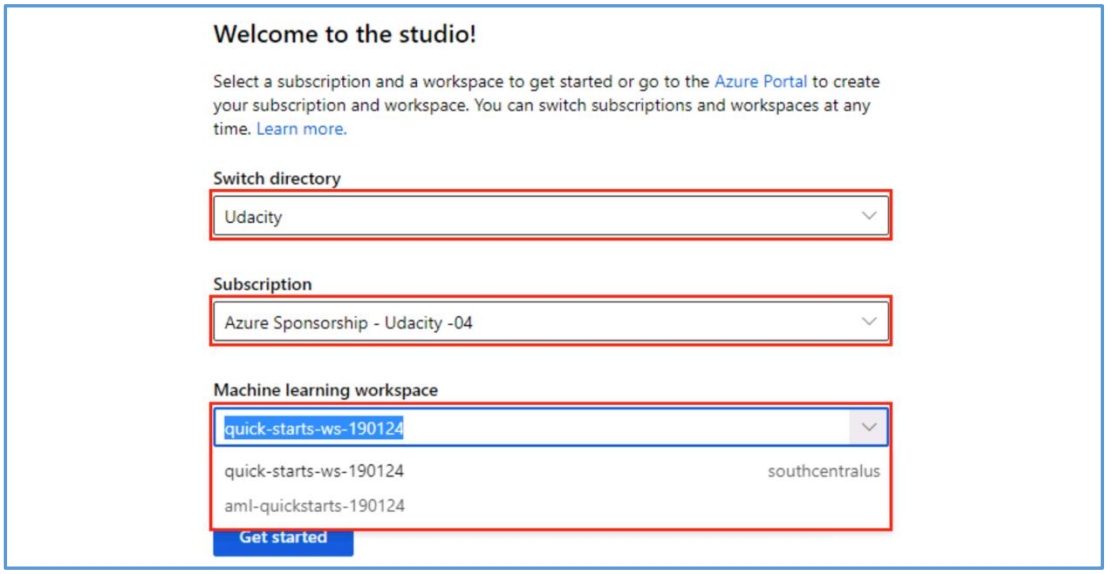
Task 1: Upload Dataset

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

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

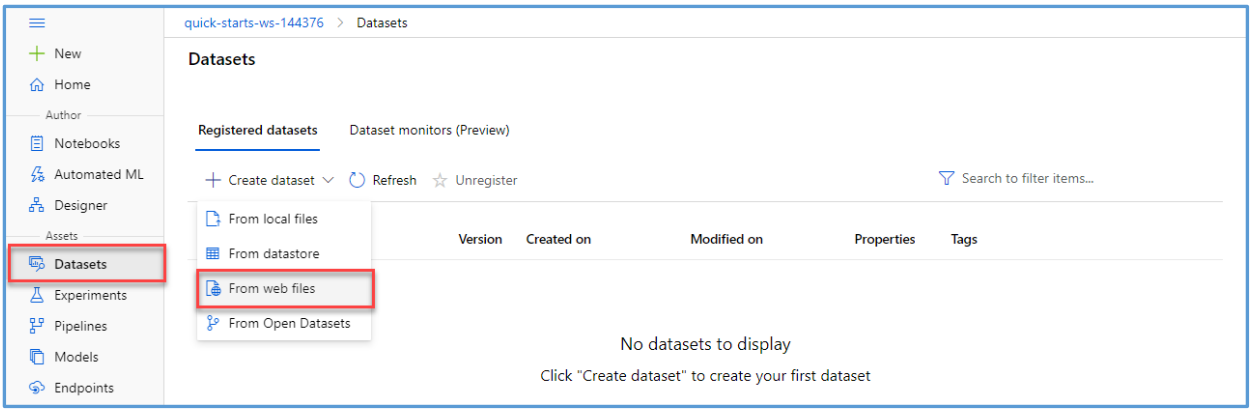


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



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

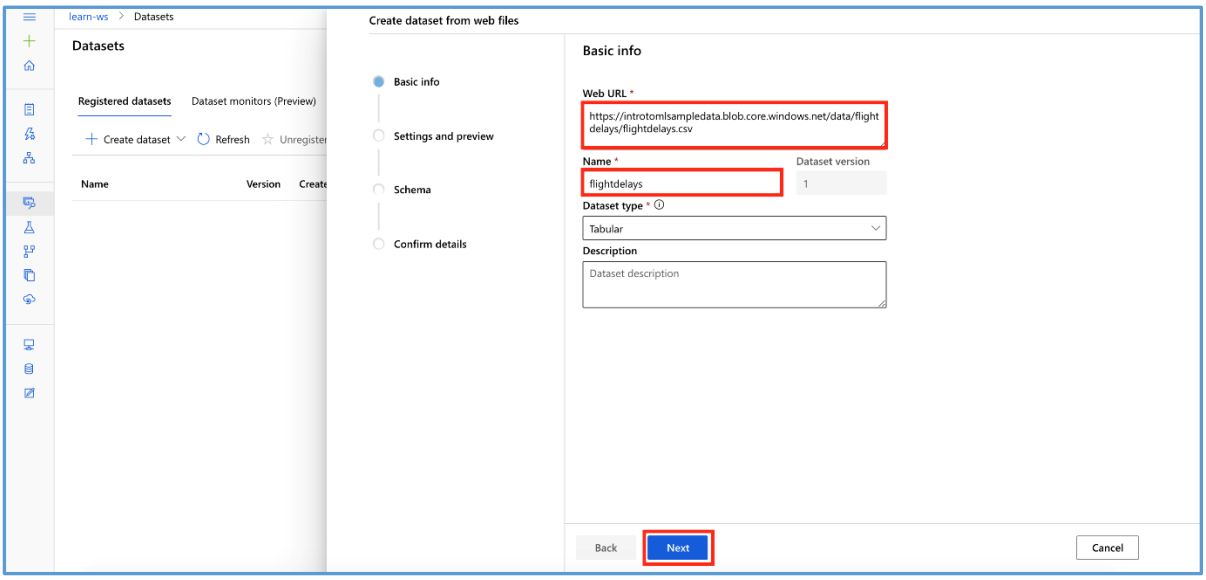
4. From the studio, select **Datasets, + Create dataset, from web files**. This will open the Create dataset from web files dialog on the right.



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

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

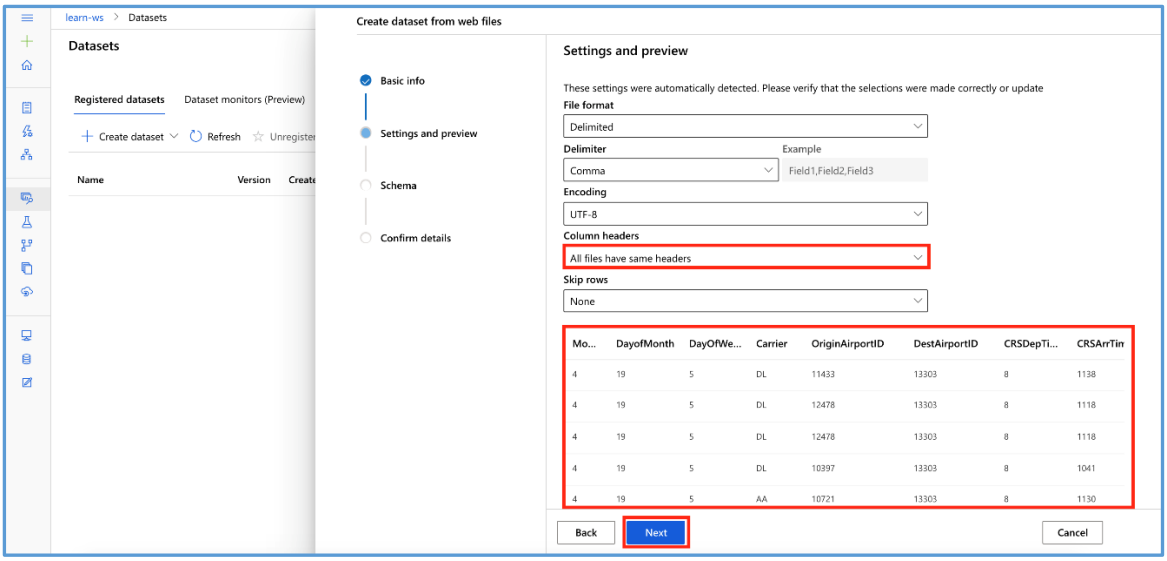
6. Provide `flightdelays` as the Name leave the remaining values at their defaults and select **Next**.



Task 2: Preview Dataset

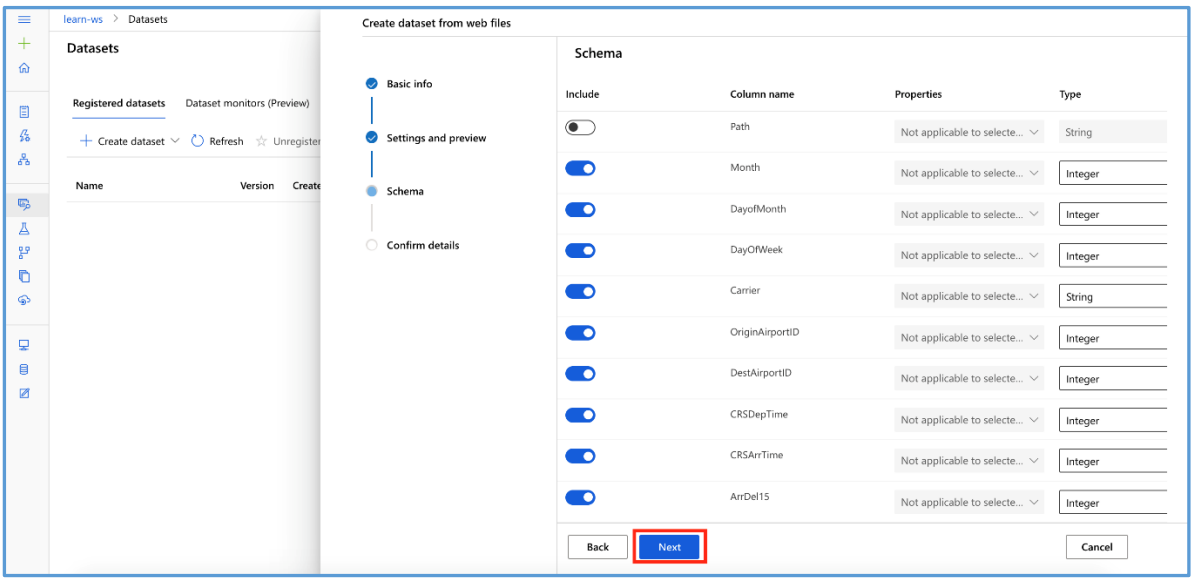
1. On the Settings and preview panel, set the column headers drop down to `All files have same headers`.

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



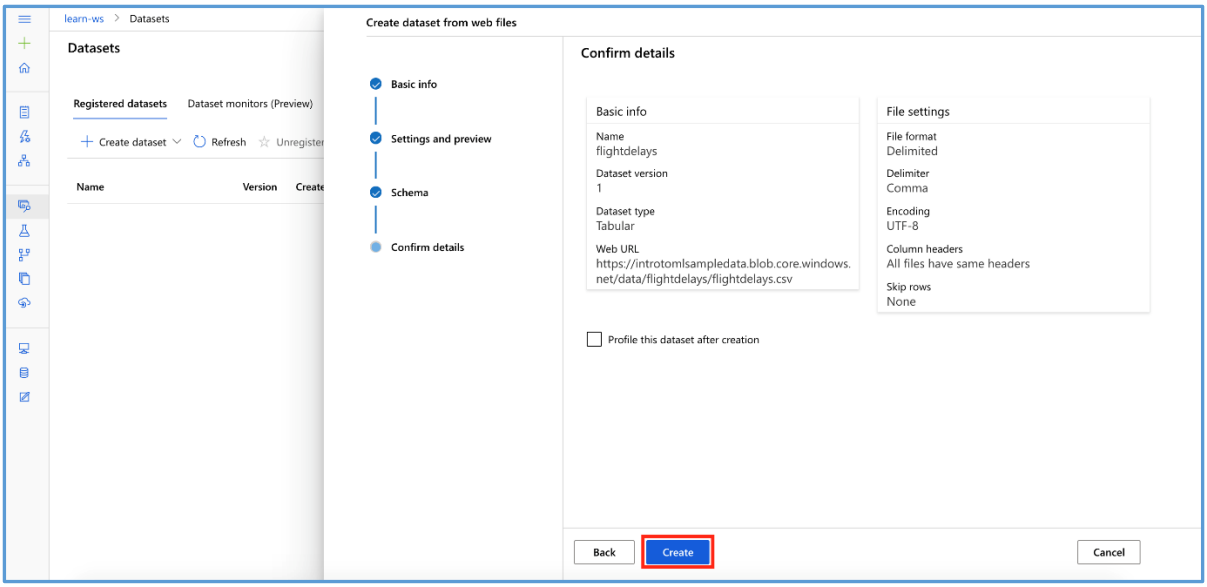
Task 3: Select Columns

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

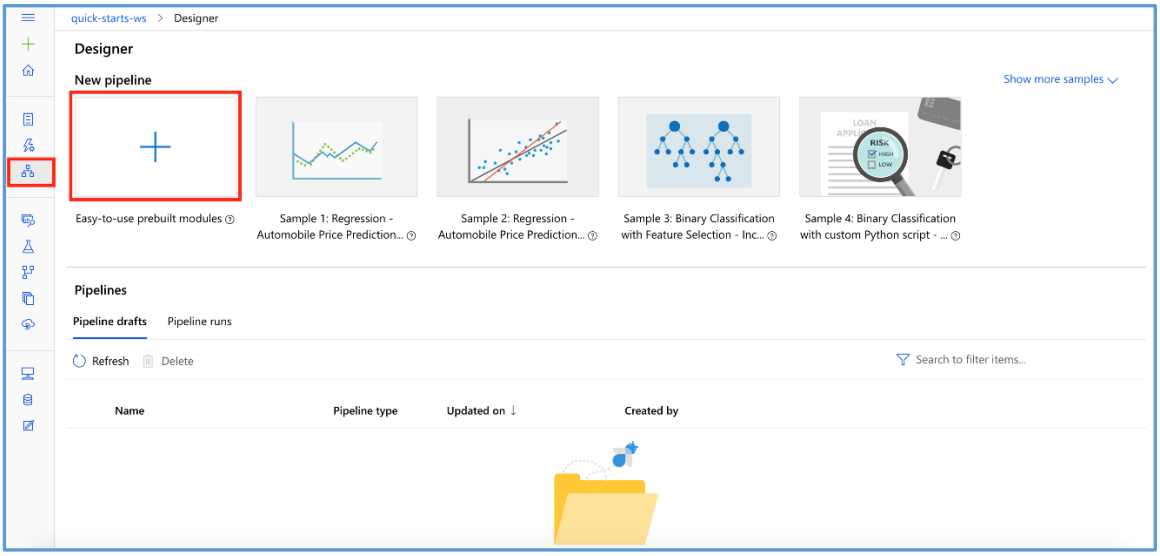
1. Confirm the dataset details and select **Create**



Exercise 2: Create New Training Pipeline

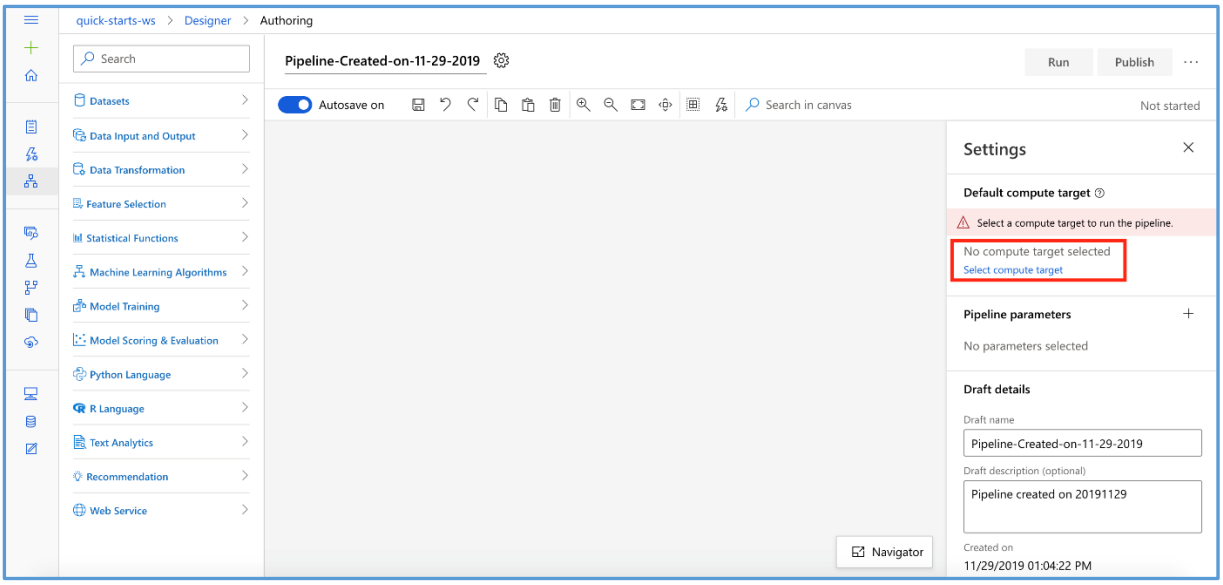
Task 1: Open Pipeline Authoring Editor

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



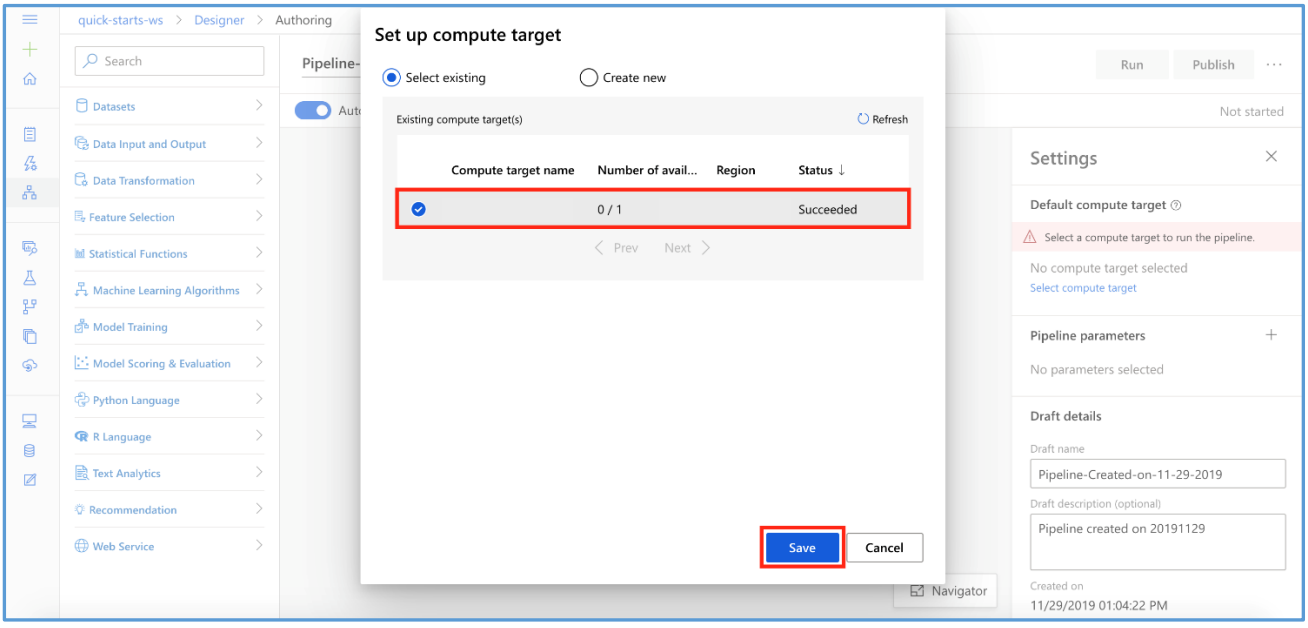
Task 2: Setup Compute Target

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



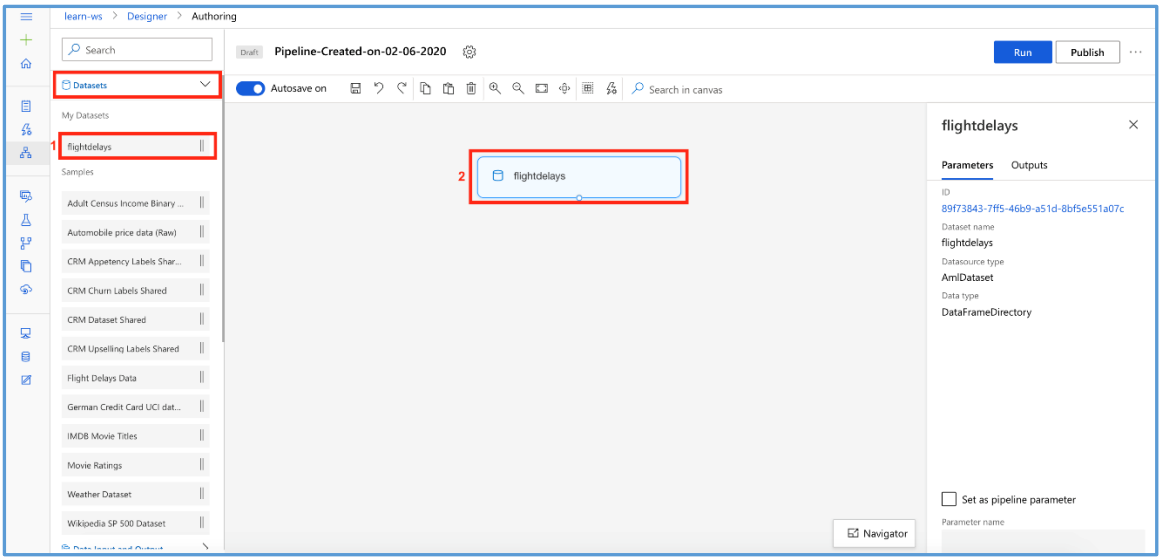
2. In the Set up compute target editor, select the available compute, and then select **Save**.

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:

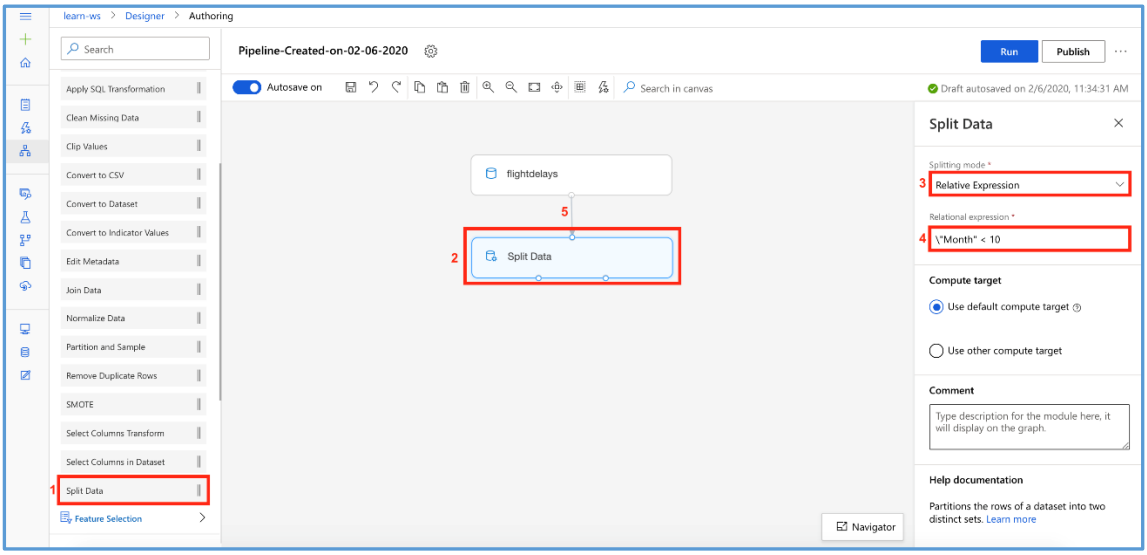
    1. Select the **Split Data** prebuilt module

    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

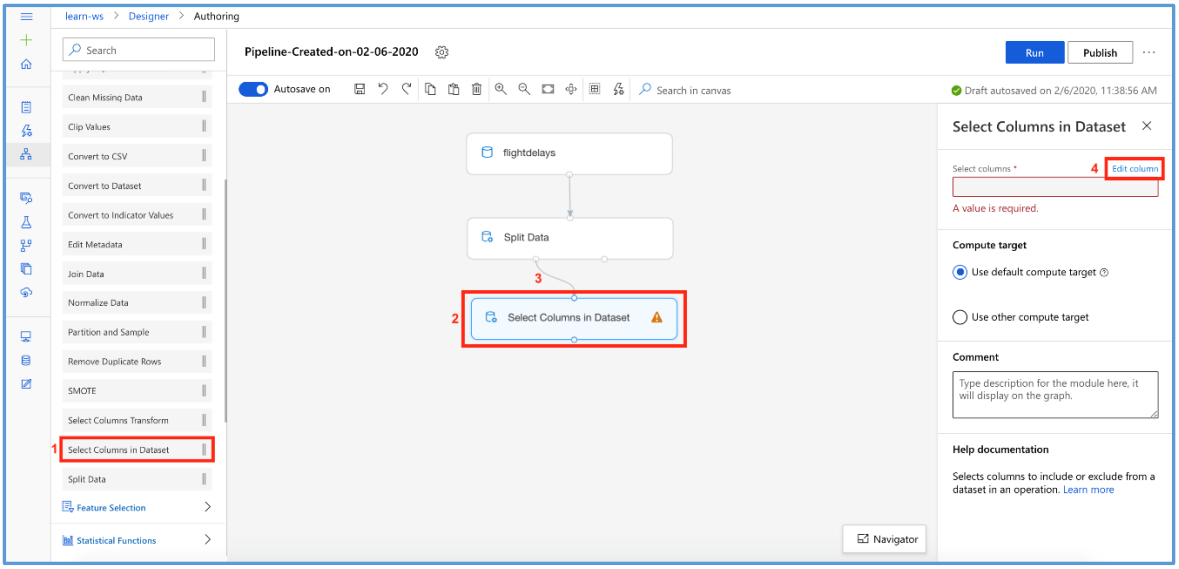
1. Select **Data Transformation** section in the left navigation. Follow the steps outlined below:

    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



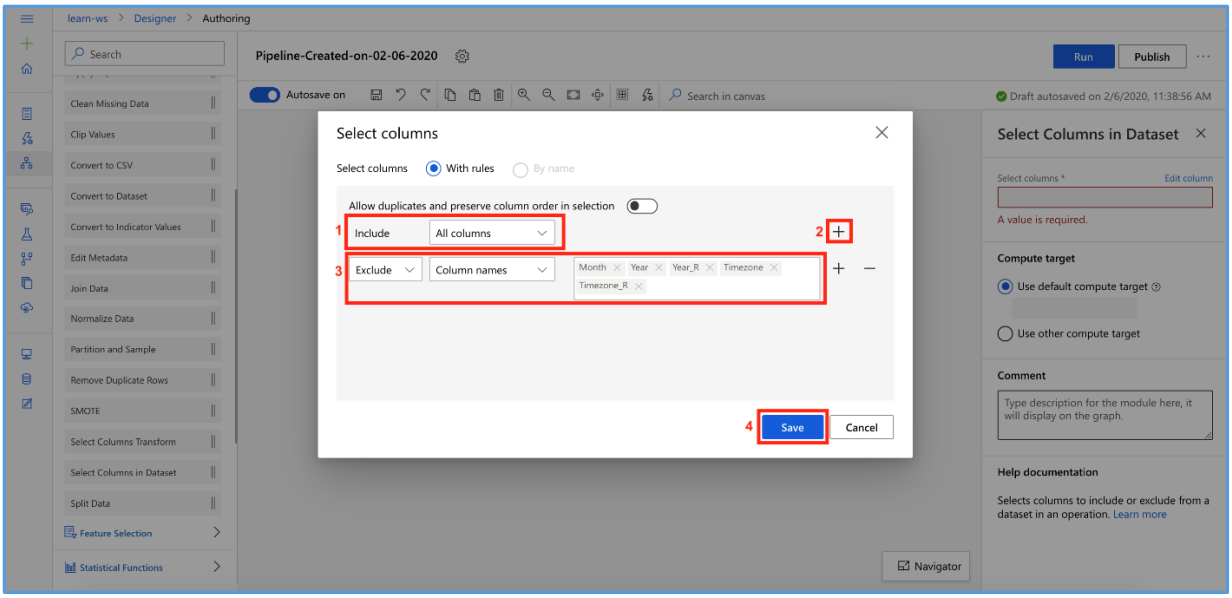
2. In the `Select columns` editor, follow the steps outlined below:

    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**

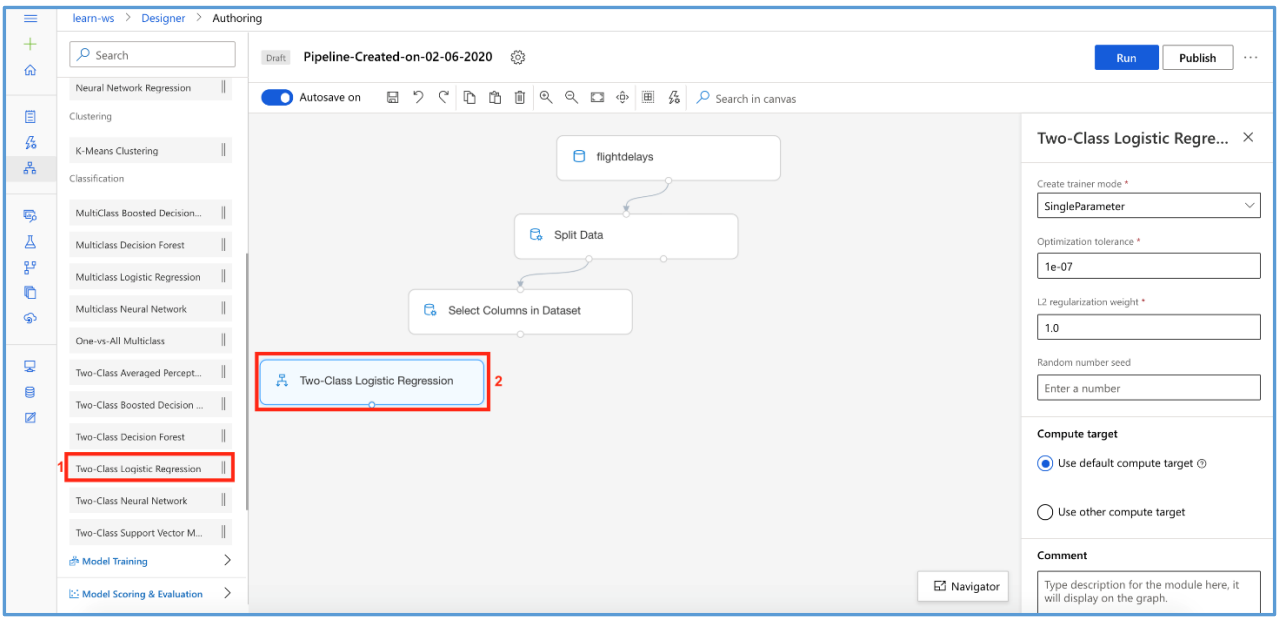


Task 6: Initialize Classification Model

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:

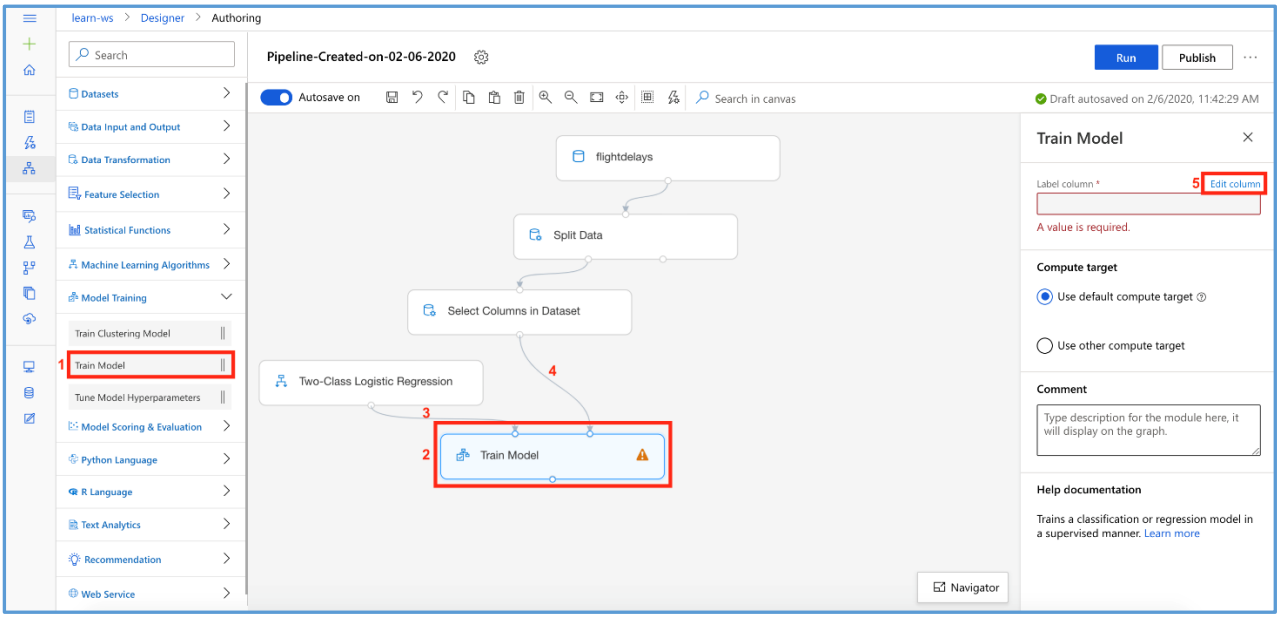
    1. Select the **Train Model** prebuilt module

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

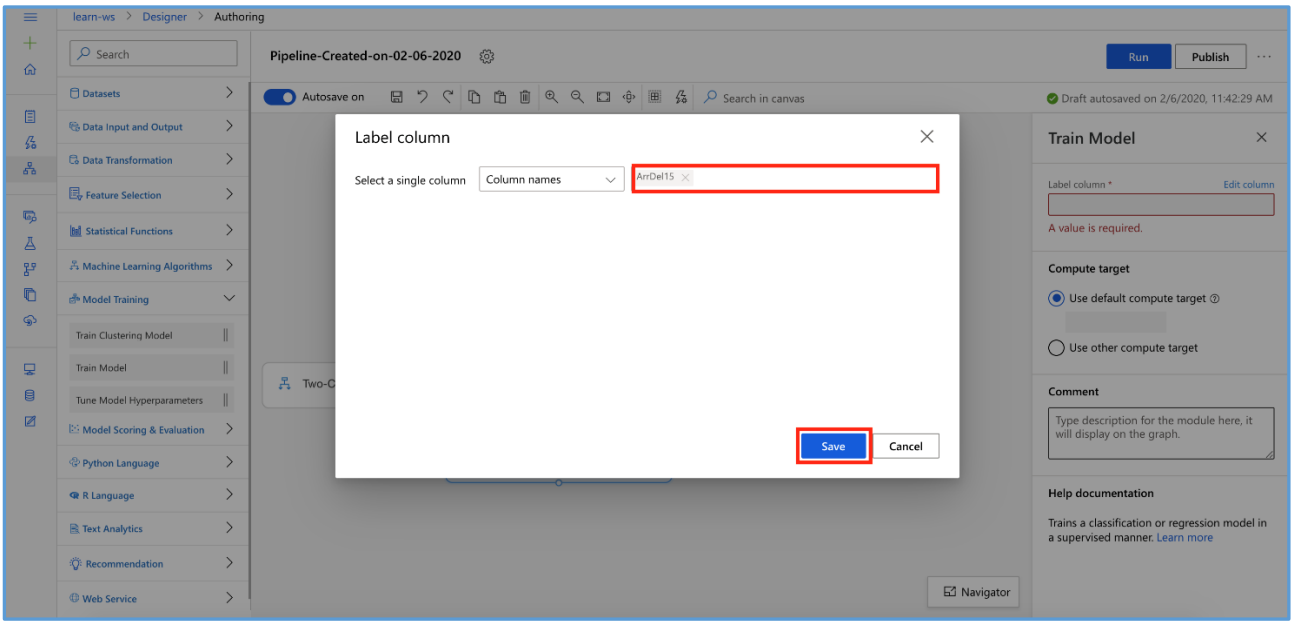
3. Connect the Two-Class Logistic Regression module to the first input of the Train Model module

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

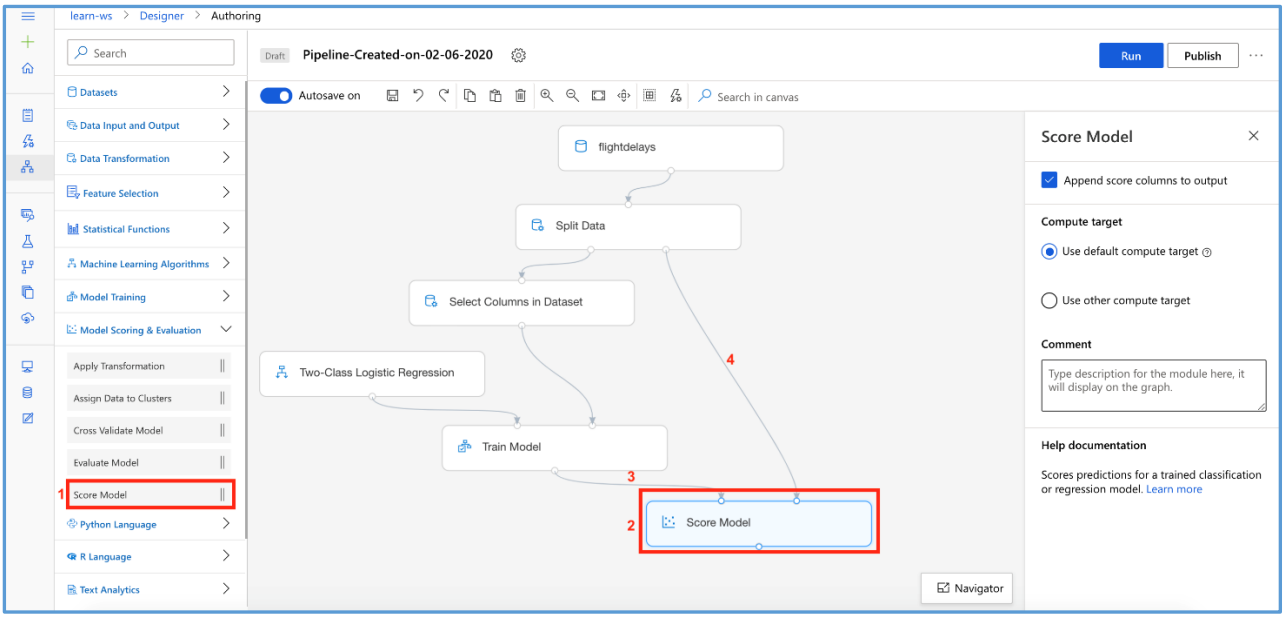
1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:

    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.

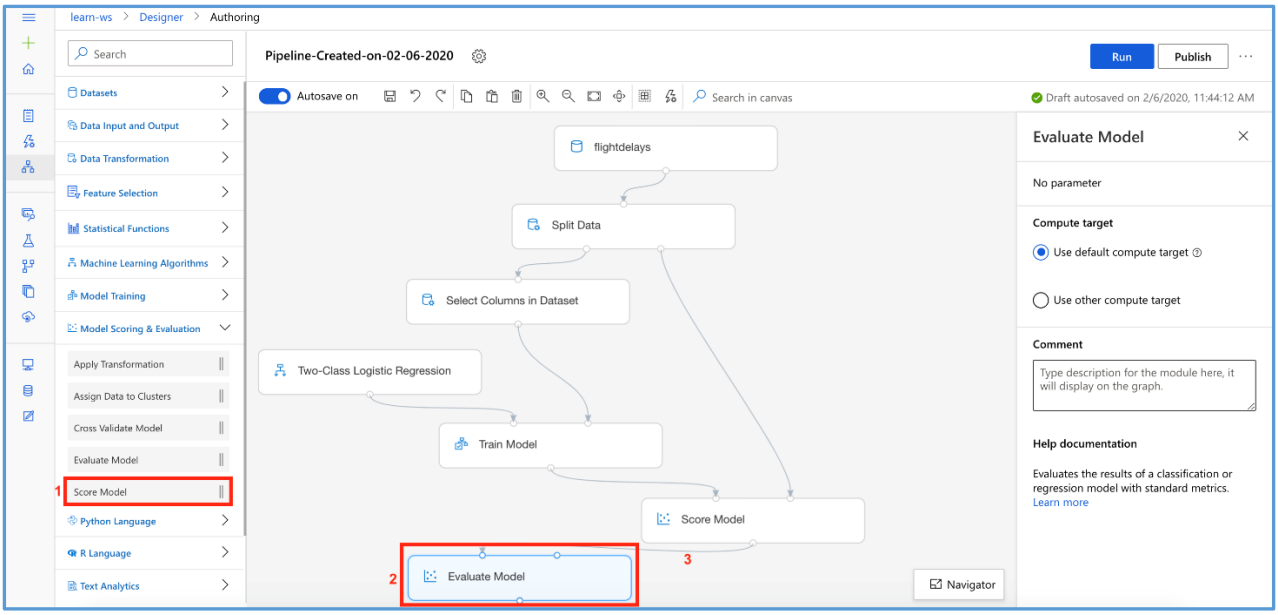
Task 9: Setup Evaluate Model Module

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

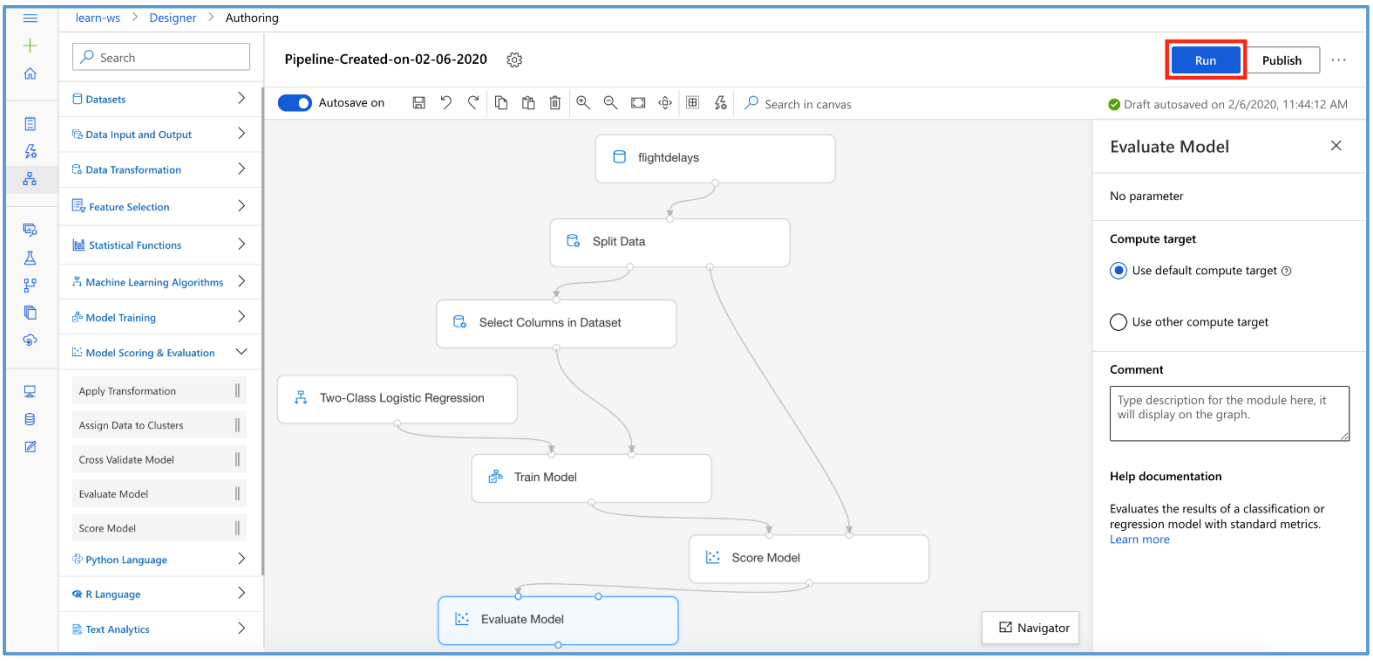
    3. Connect the Score Model module to the first input of the Evaluate Model module



Exercise 3: Submit Training Pipeline

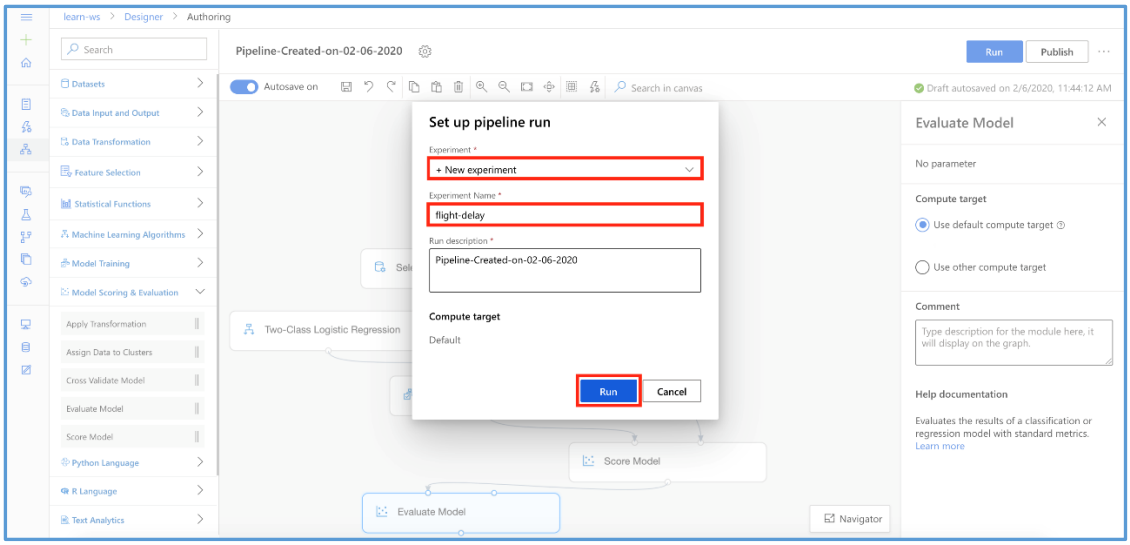
Task 1: Create Experiment and Submit Pipeline

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**.



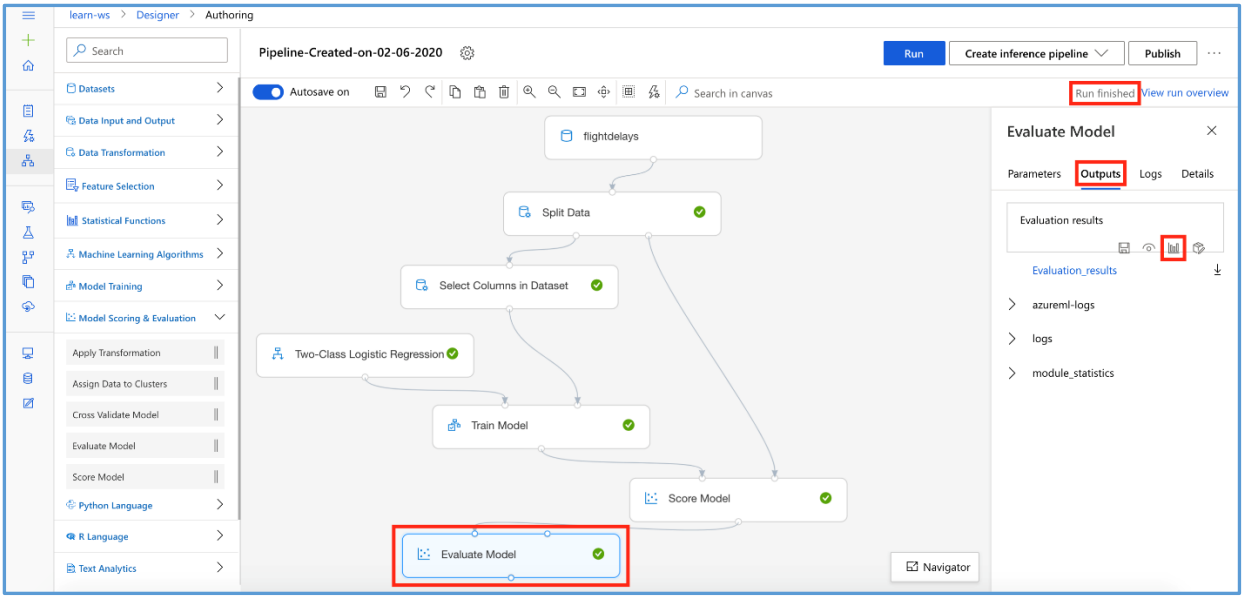
3. Wait for pipeline run to complete. It will take around **10 minutes** to complete the run.

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

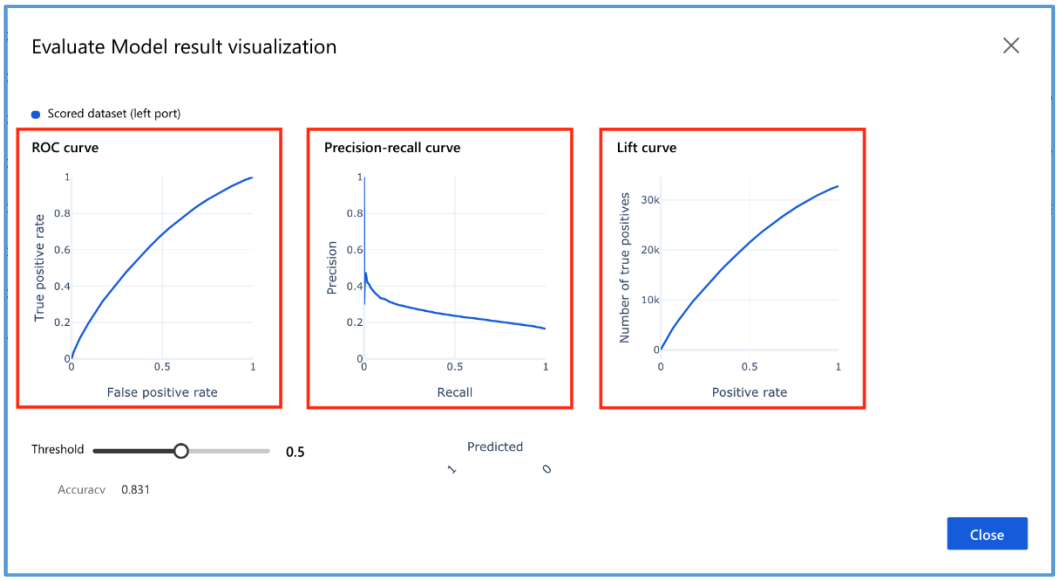
Task 1: Open the Result Visualization Dialog

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**

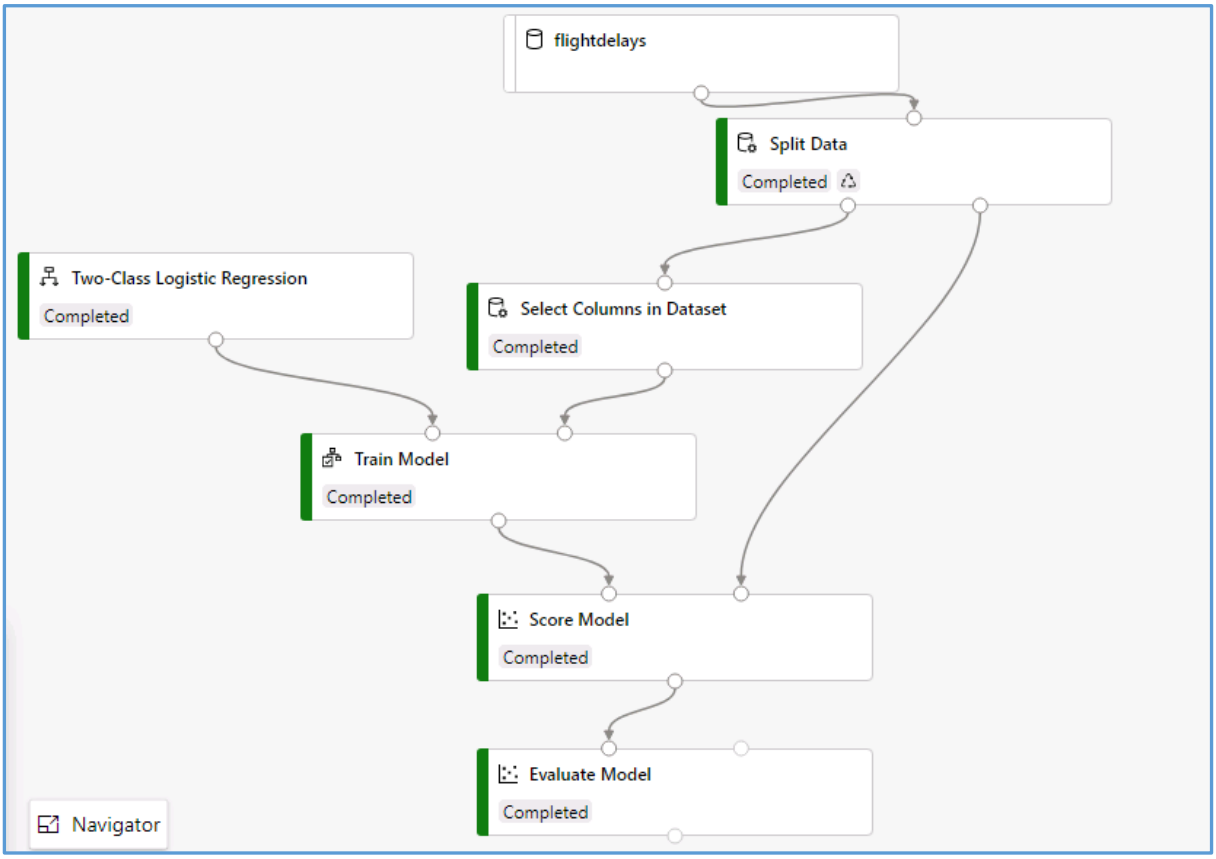
    2. Review the binary classifier's **Confusion Matrix**

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.

Chapter 28: Walkthrough – Train and Evaluate a Model

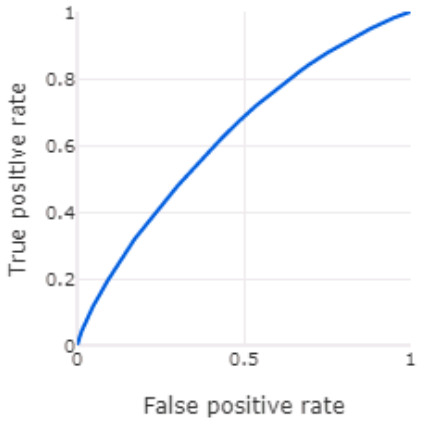
- Set the compute resource for the ML pipeline
- Register a new dataset with Azure ML. URL needs to be, specified for creating the dataset.
- Split the data using the "Relative Expression" splitting mode & Relational expression of "\"Month" < 10. Thus, all data less than 10 months would be Training data and remaining would be Test data.
- For the Training data, we are excluding a set of columns and for this, we are using "Select Column in dataset" task.
- ML algorithm used is Two Class Logistic Regression.
- Then we used the "Train Model", and connected the output of Algorithm to the first input and in the second input connect the output from "Select Column in dataset" task, which is the finalized version of the Training dataset.
- Then we used the "Score Model" task to score the Training and Test datasets.
- Finally, we used the "Evaluate Model" task for evaluation purpose.



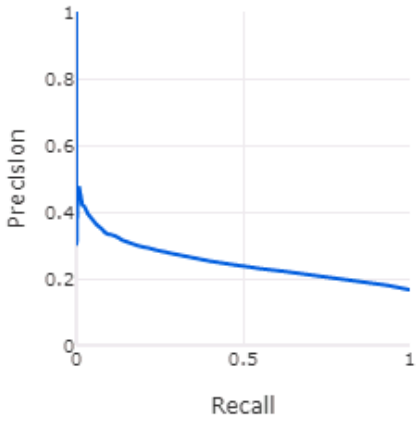
Evaluate Model result visualization

Scored dataset (left port)

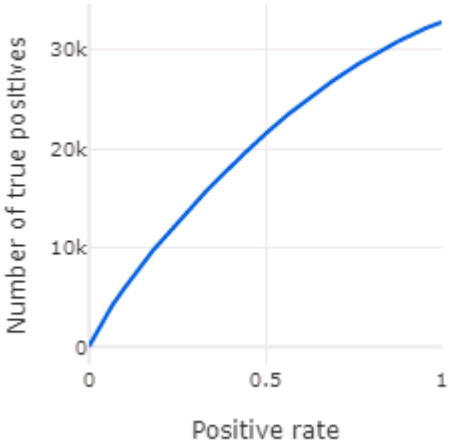
ROC curve



Precision-recall curve



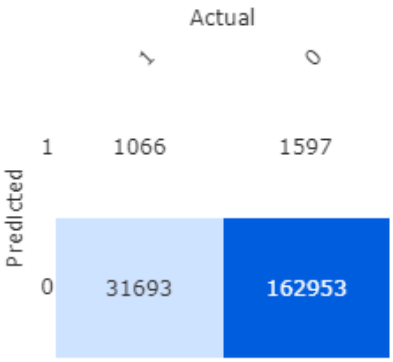
Lift curve



Evaluate Model result visualization

Threshold 0.5

Accuracy	0.831
Precision	0.4
Recall	0.033
F1 Score	0.06
AUC	0.631



Chapter 29: Prelaunch Lab

Prelaunch your lab environment.

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

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

Chapter 30: Strength in Numbers

Remember, no matter how well trained an individual model is, there is still a significant chance that it could perform poorly or produce incorrect results. Rather than relying on a single model, you can often get better results by training multiple models or using multiple algorithms and in some way capturing the collective results. As we mentioned, there are two main approaches to this: **Ensemble learning** and **automated machine learning**. Let us have a closer look at each of them.

Ensemble learning

- Ensemble algorithms is a powerful technique to combine multiple machine learning models to produce one predictive model

One ML model would definitely have issues in producing accurate predictions. Collective wisdom of a large population is statistically better than decision making of a single individual.

We take number of ML algorithms and then train them on the same dataset; finally, we apply some kind of combinations to their output to get the final prediction.

Remember, **ensemble learning** combines multiple machine learning models to produce one predictive model. There are three main types of ensemble algorithms:

Bagging or **bootstrap aggregation**

- Helps reduce overfitting for models that tend to have high variance (such as *decision trees*)
- Uses random subsampling of the training data to produce a *bag* of trained models.
- The resulting trained models are homogeneous
- The final prediction is an average prediction from individual models

Steps involved:

- Randomly select the samples of the data, & train ML models using the randomly selected datasets. **[Sampling data from input data & training multiple ML models]**
- Repeat the process several times.
- Result is number of trained ML models.
- Then we assign equal weights to the predicted outputs of each individual ML model.
- Combine the final output

Boosting

- Helps reduce bias for models.**[Same input data but train multiple ML models by using different values for their hyperparameters]**

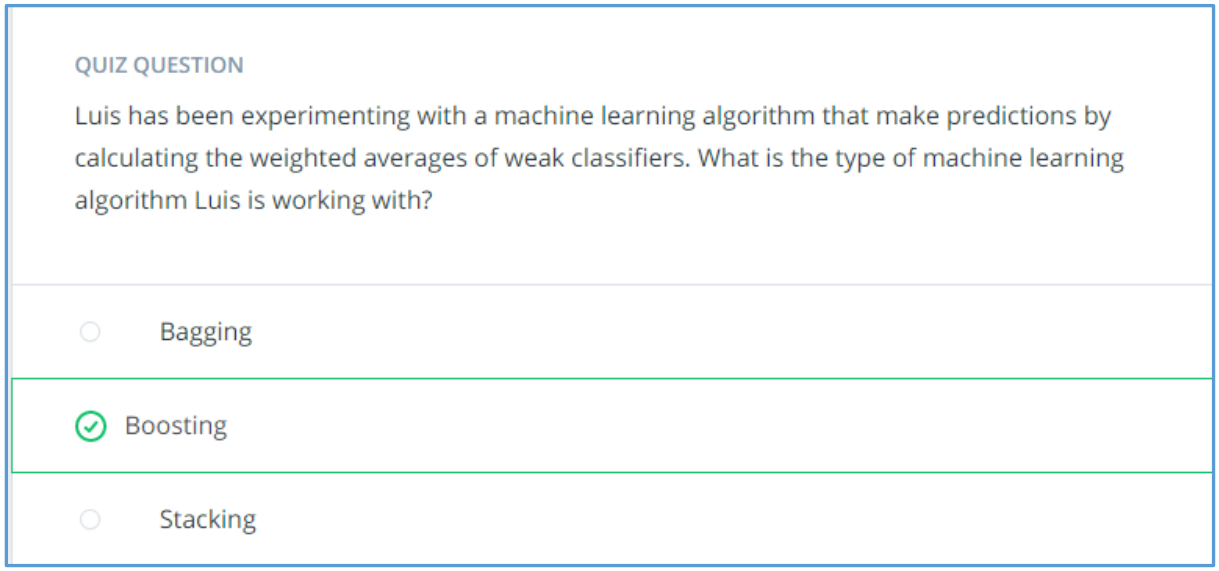
- In contrast to bagging, boosting uses the same input data to train multiple models using different hyperparameters.
- Boosting trains model in *sequence* by training weak learners one by one, with each new learner correcting errors from previous learners
- The final predictions are a *weighted* average from the individual models

Steps involved:

- Use same input data but train multiple ML models by using different values of hyperparameters.
- We are using same learners & since the method is sequential, thus performance is constantly improving for the resulting model. [Produce strong learners using Weak learners]

Stacking

- Trains a large number of completely different (heterogeneous) models
- Combines the outputs of the individual models into a meta-model that yields more accurate predictions



Automated machine learning, as the name suggests, automates many of the iterative, time-consuming, tasks involved in model development (such as selecting the best features, scaling features optimally, choosing the best algorithms, and tuning hyperparameters). Automated ML allows data scientists, analysts, and developers to build models with greater scale, efficiency, and productivity—all while sustaining model quality.

Hyperparameters are the components that can tune the behaviour of our model. Thus, we get to use same ML algorithm but different behaviour. Thus, we have to choose that which variation amongst all is best suited for our problem.

AutoML trains multiple ML model with different combinations of hyperparameters, features, & Algorithms. This increases the efficiency and productivity of the ML training process.

Chapter 31: Lab – Train a Two – Class – Boosted – Decision Tree

Lab Overview

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

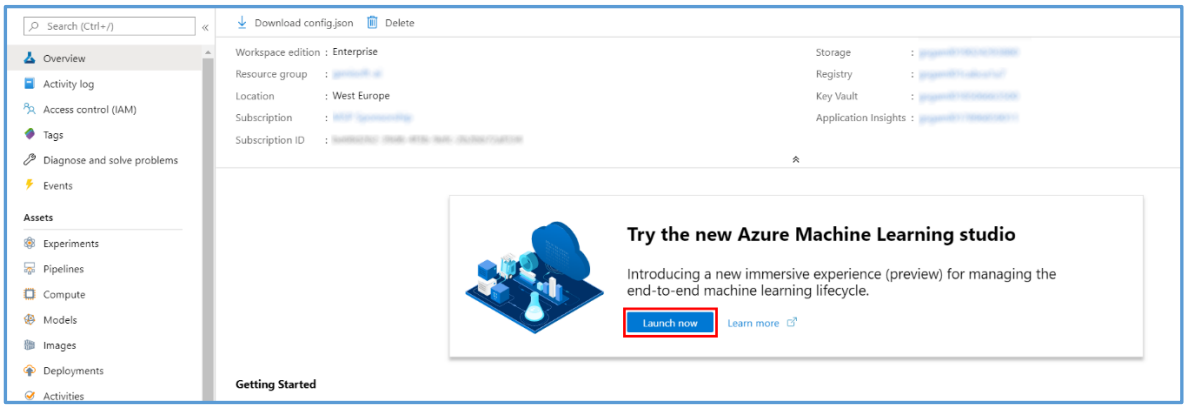
In this lab, we will be using the Flight Delays data set that is, enhanced with the weather data. Based on the enriched dataset, we will learn to use the Azure Machine Learning Graphical Interface to process data, build, train, score, and evaluate a classification model to predict if a particular, flight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

Exercise 1: Register Dataset with Azure Machine Learning studio

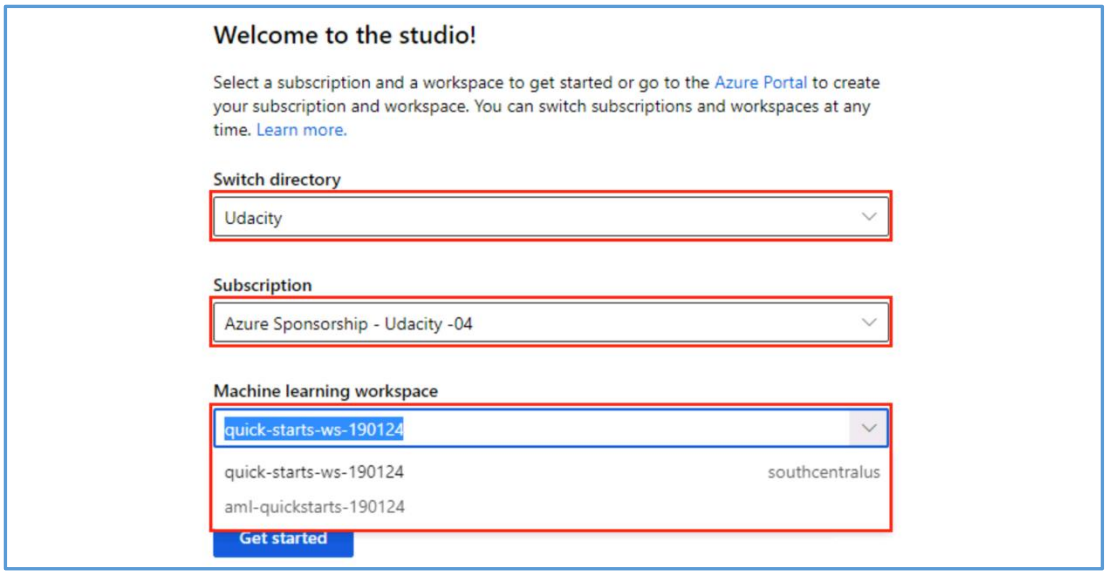
Task 1: Upload Dataset

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

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

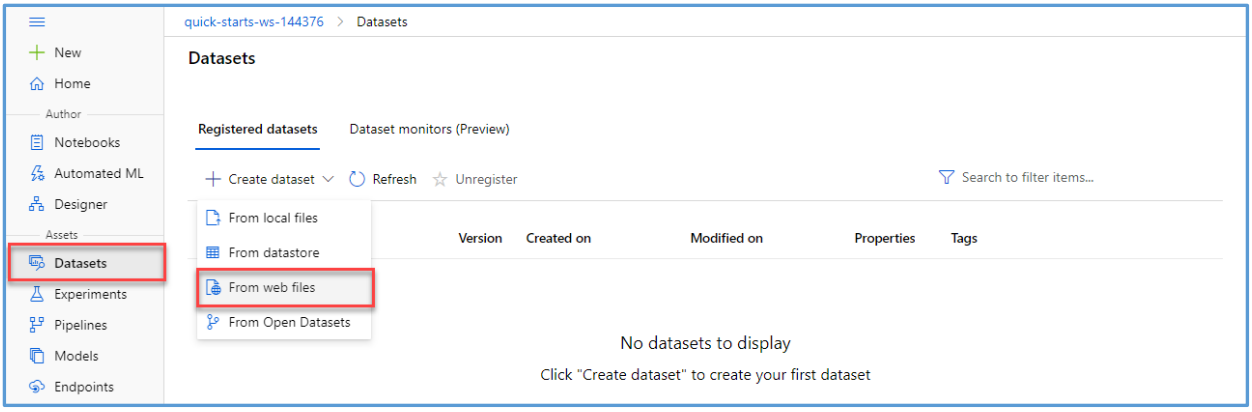


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



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

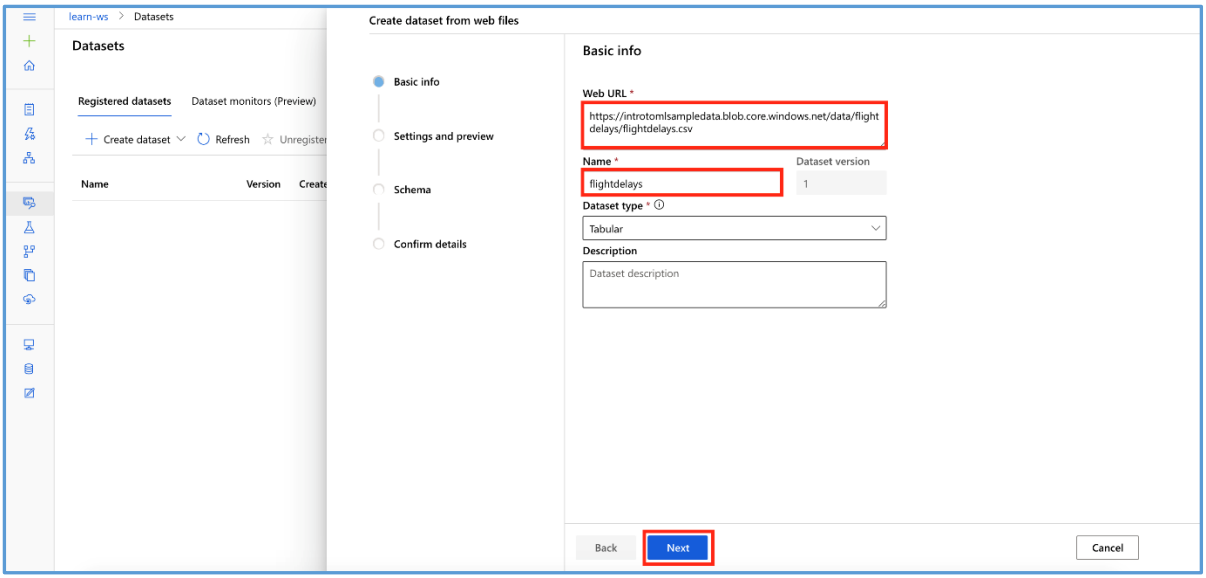
4. From the studio, select **Datasets, + Create dataset, from web files**. This will open the `Create dataset from web files` dialog on the right.



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

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

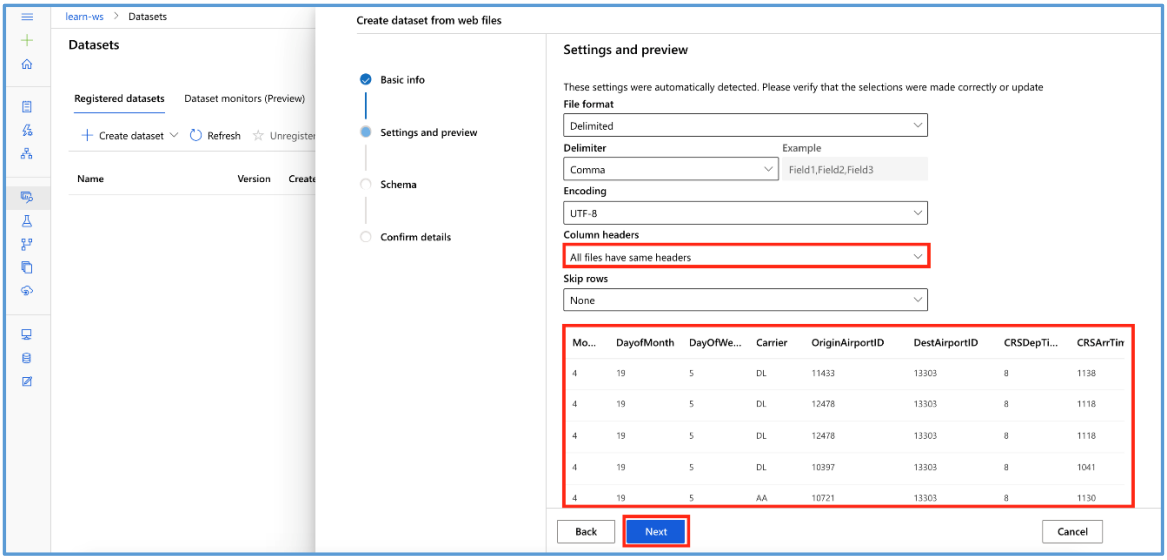
6. Provide `flightdelays` as the Name leave the remaining values at their defaults and select **Next**.



Task 2: Preview Dataset

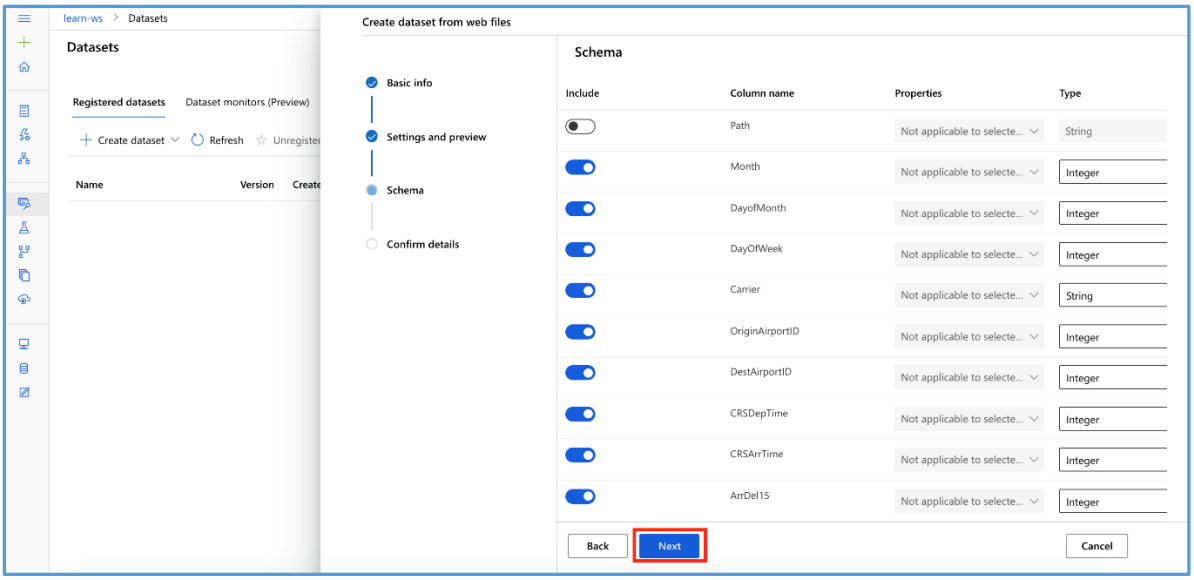
1. On the Settings and preview panel, set the column headers drop down to All files have same headers.

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



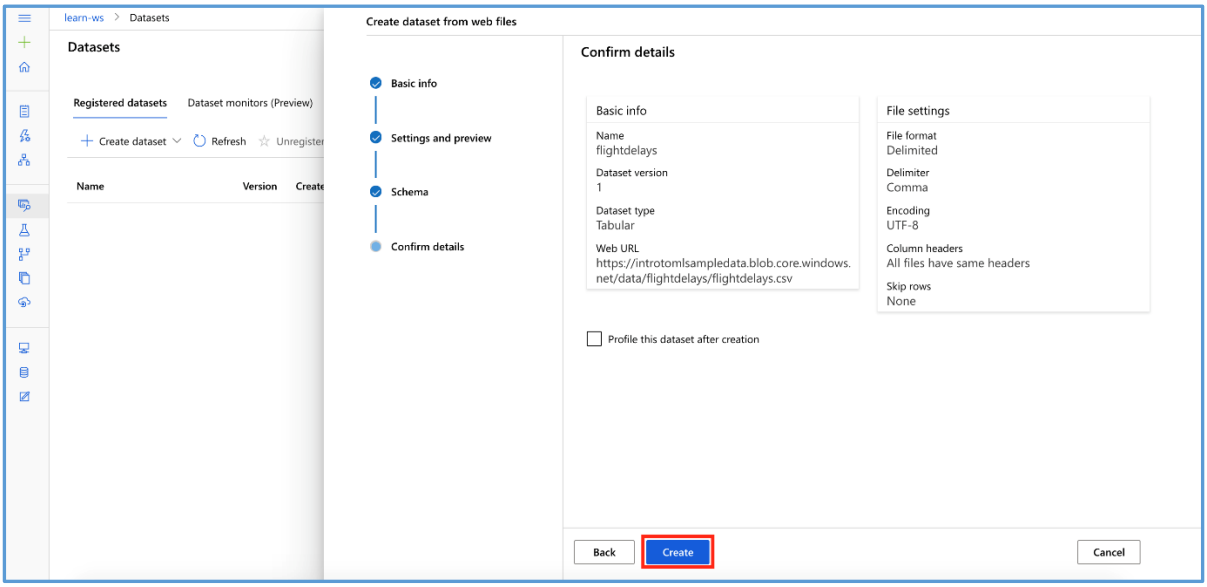
Task 3: Select Columns

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

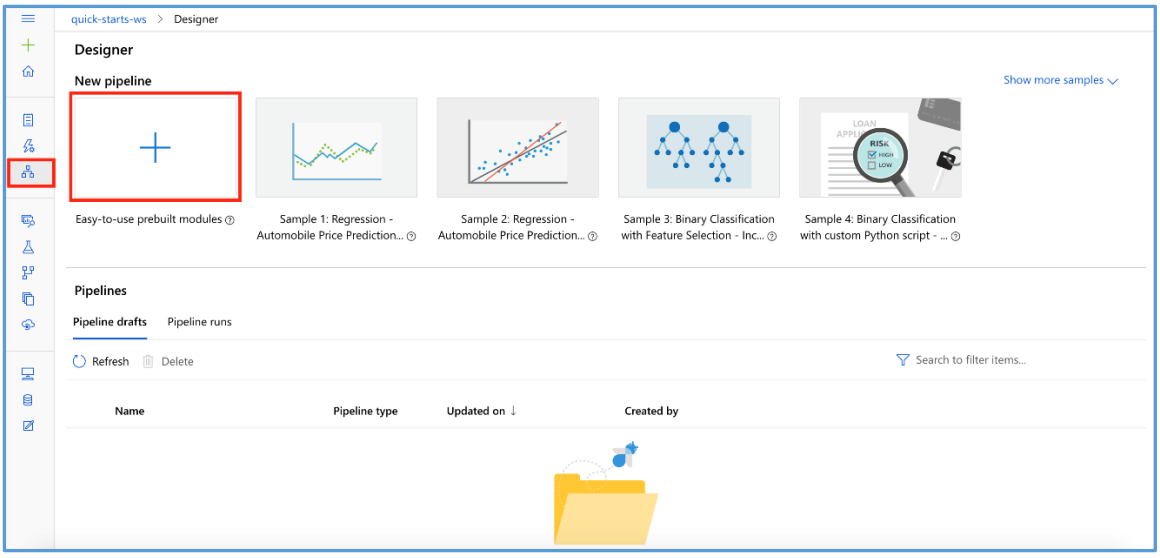
1. Confirm the dataset details and select **Create**



Exercise 2: Create New Training Pipeline

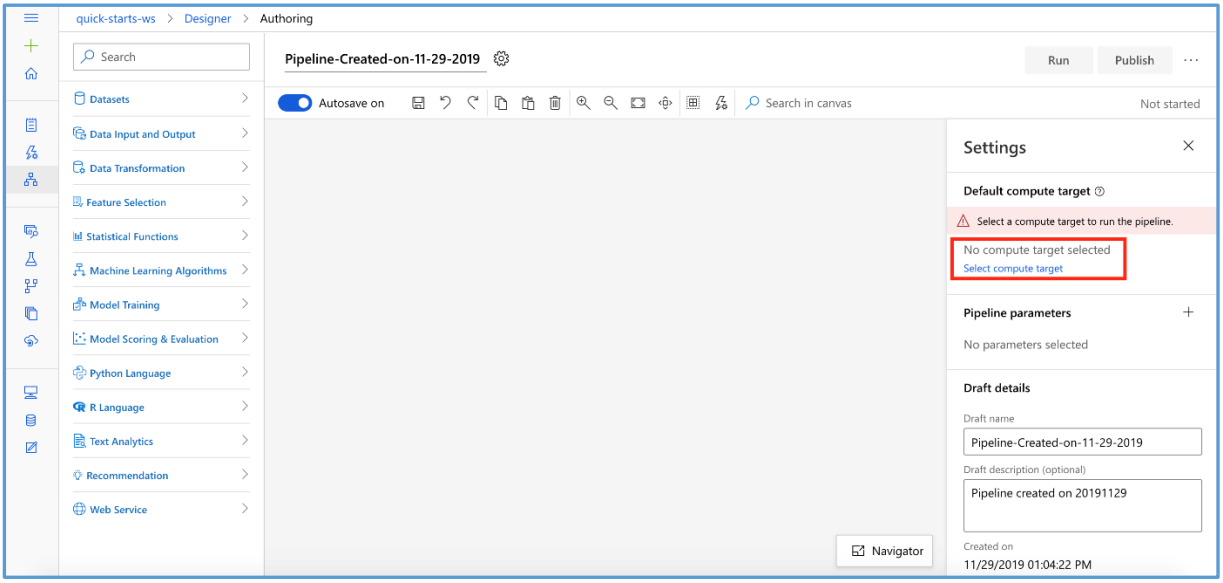
Task 1: Open Pipeline Authoring Editor

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



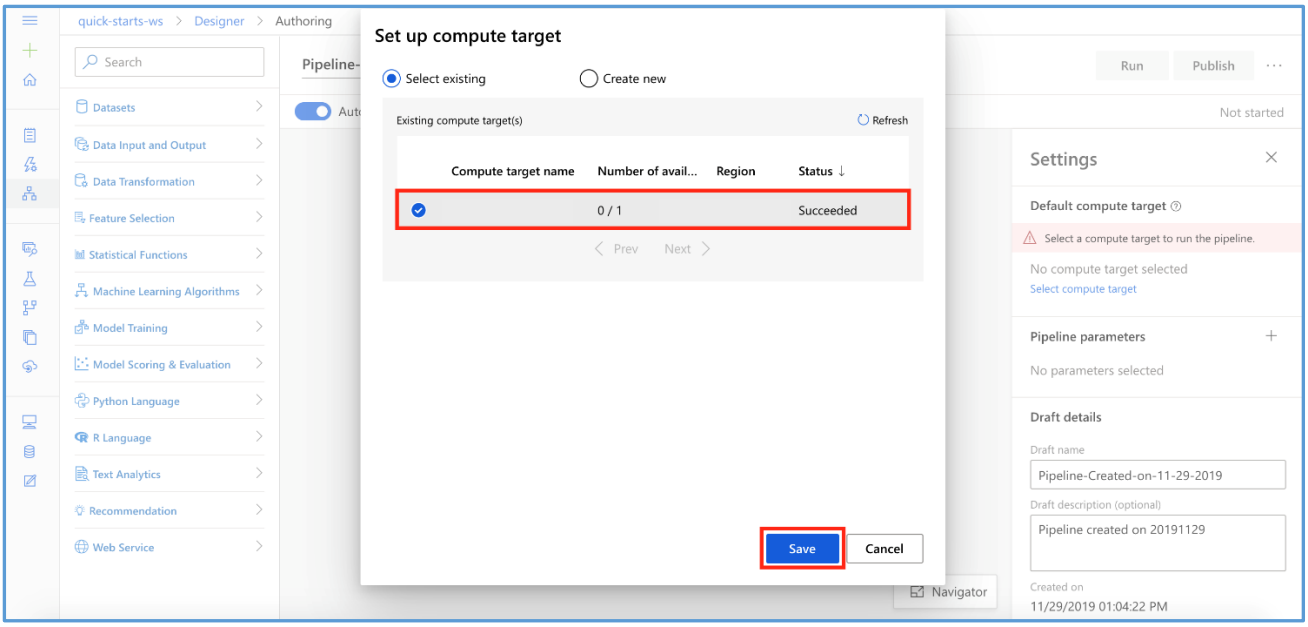
Task 2: Setup Compute Target

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



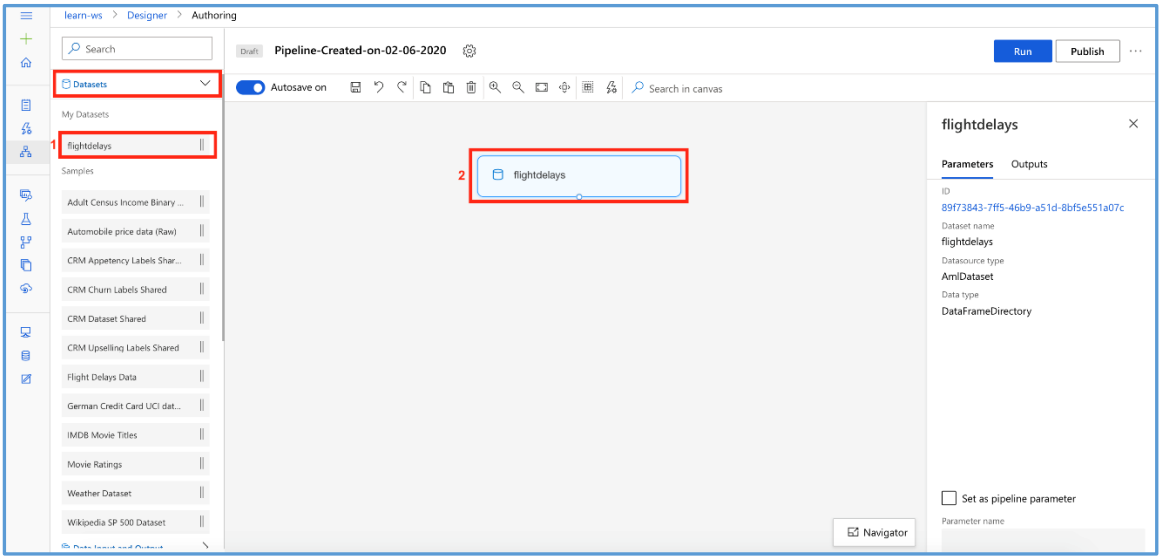
2. In the Set up compute target editor, select the available compute, and then select **Save**.

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

3. 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.

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

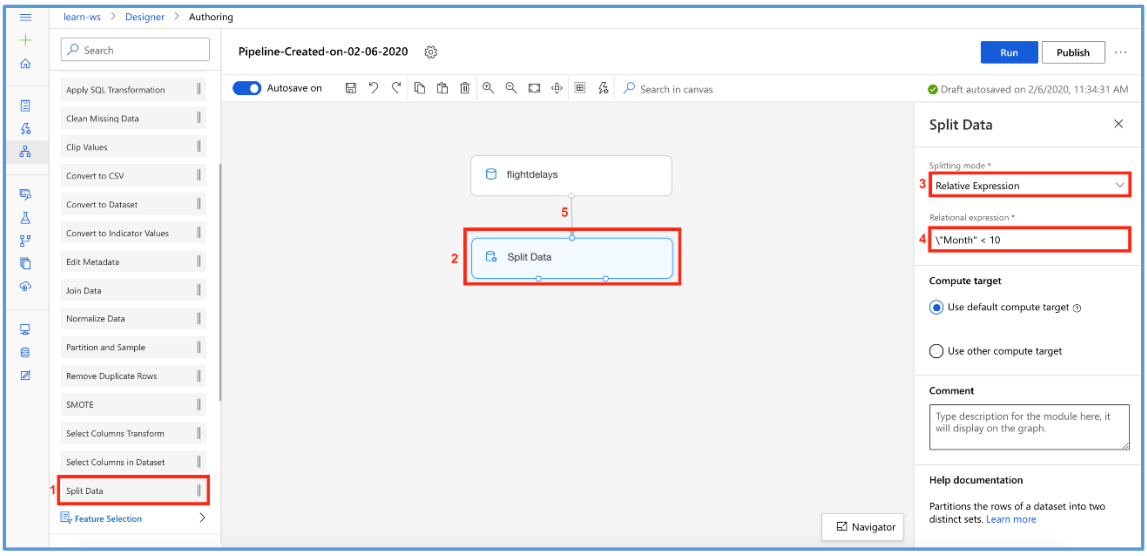
    1. Select the **Split Data** prebuilt module

    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

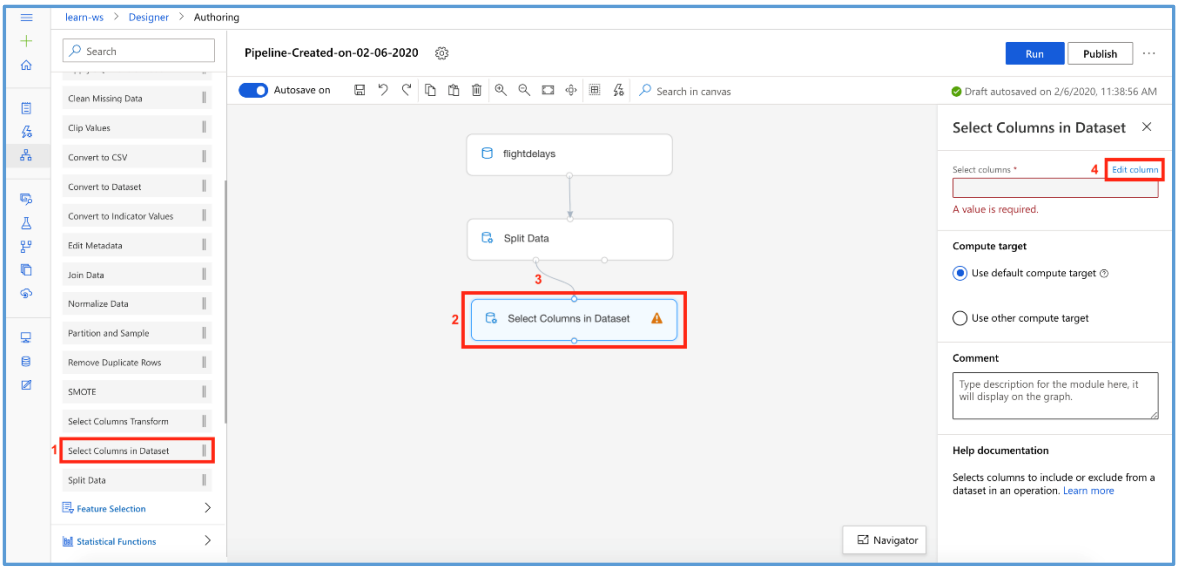
1. Select **Data Transformation** section in the left navigation. Follow the steps outlined below:

    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



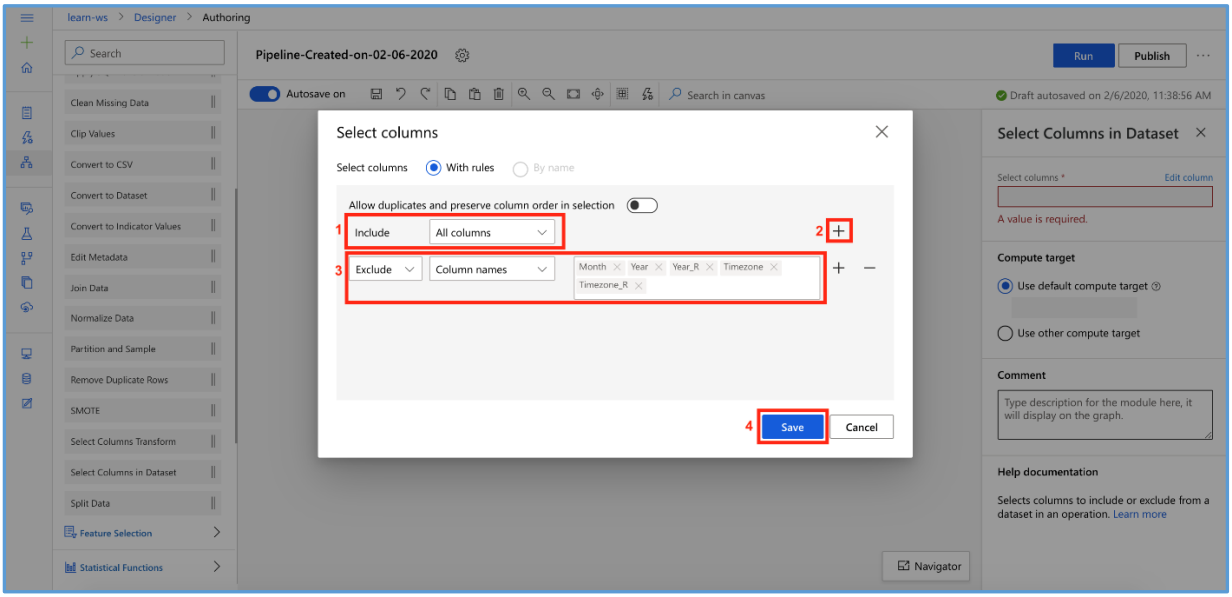
2. In the Select columns editor, follow the steps outlined below:

    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**

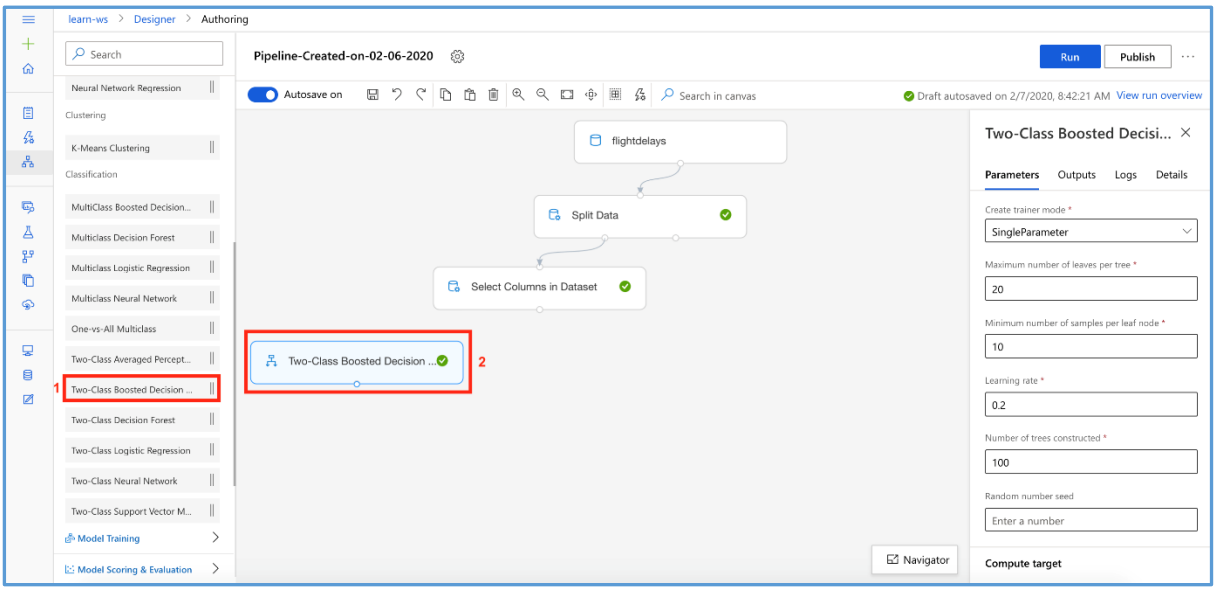


Task 6: Initialize Classification Model

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

1. Select the **Two-Class 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:

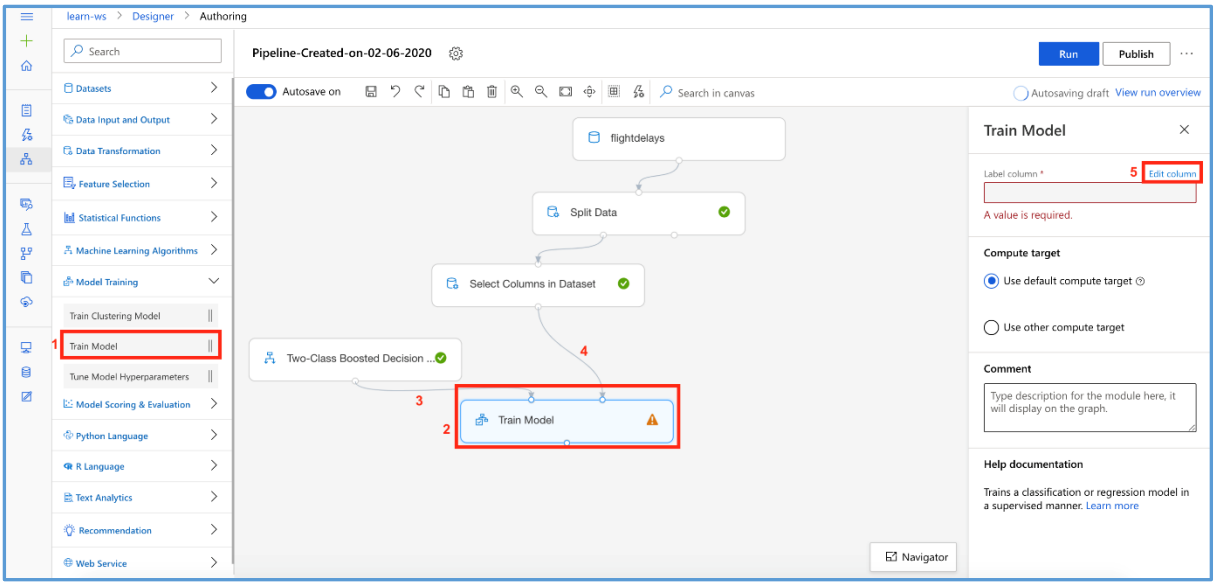
    1. Select the **Train Model** prebuilt module

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

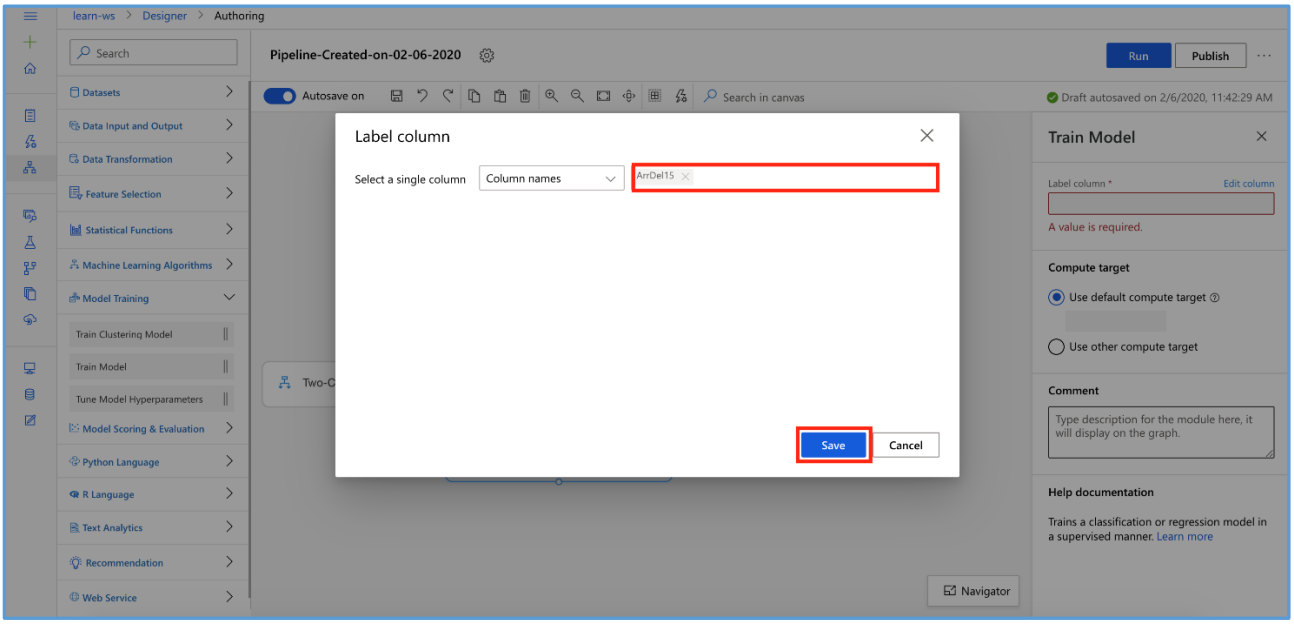
    3. Connect the `Two-Class Boosted Decision Tree` module to the first input of the `Train Model` module

    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

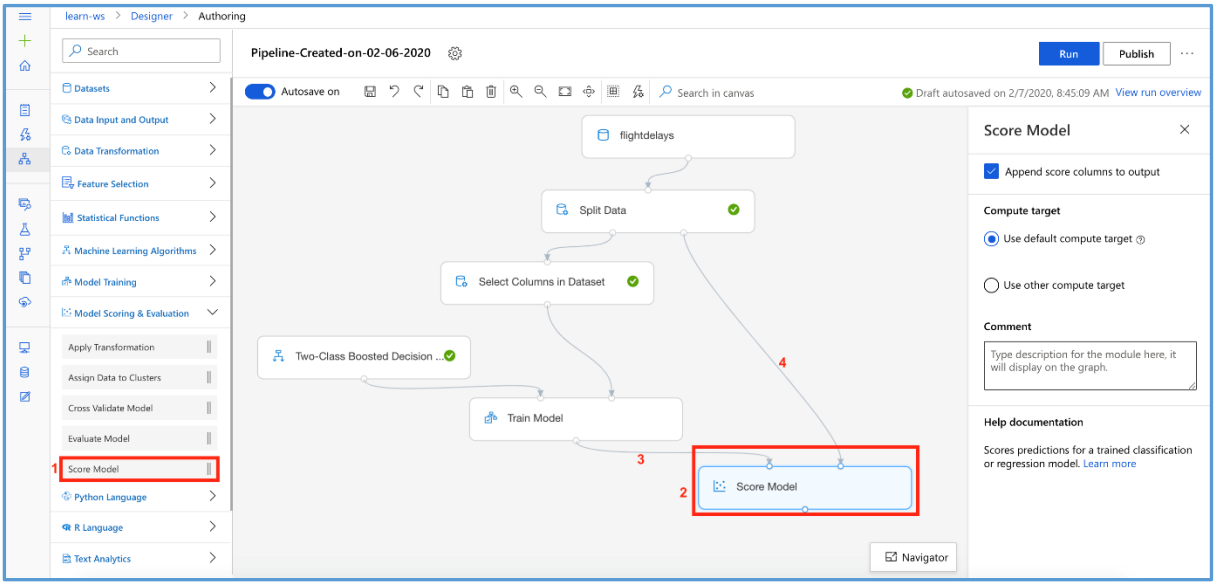
1. Select **Model Scoring & Evaluation** section in the left navigation. Follow the steps outlined below:

    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.

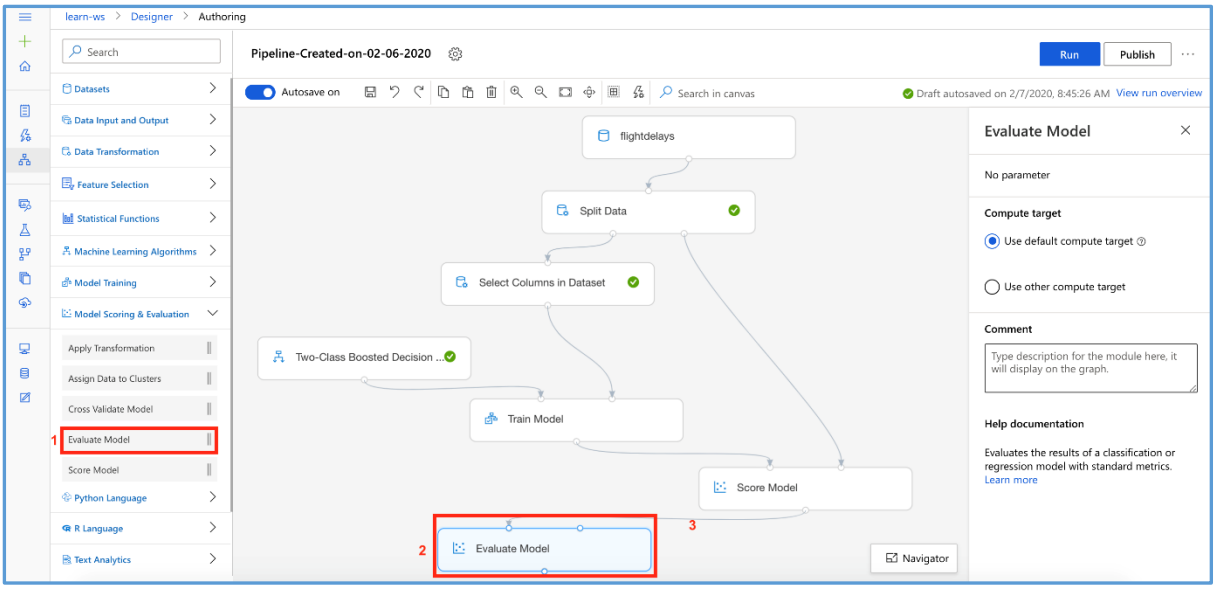
Task 9: Setup Evaluate Model Module

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

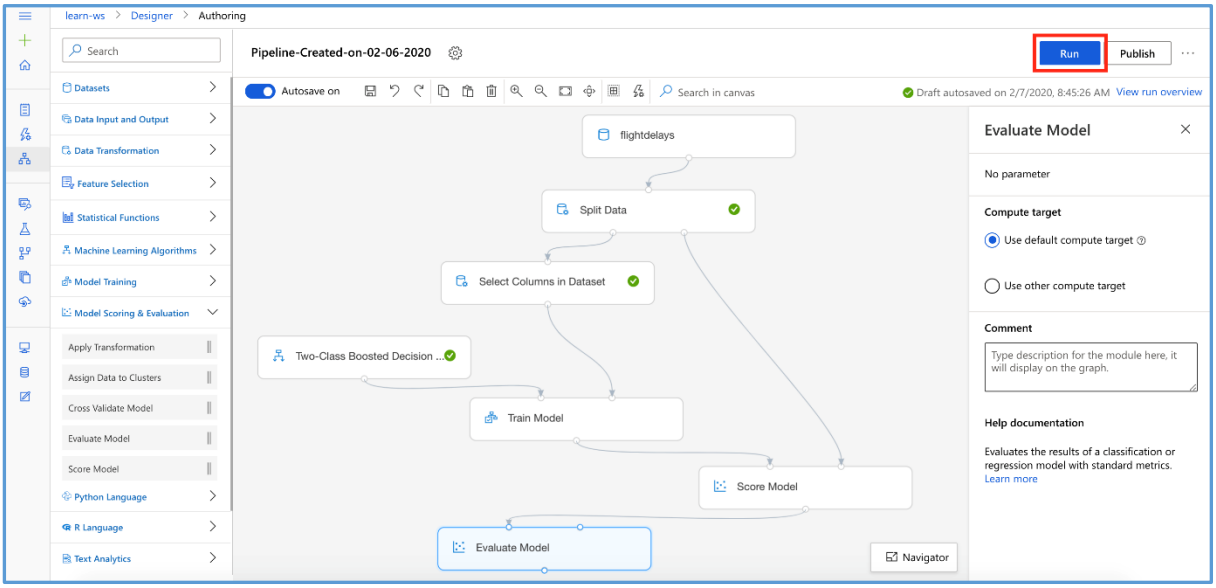
    3. Connect the Score Model module to the first input of the Evaluate Model module



Exercise 3: Submit Training Pipeline

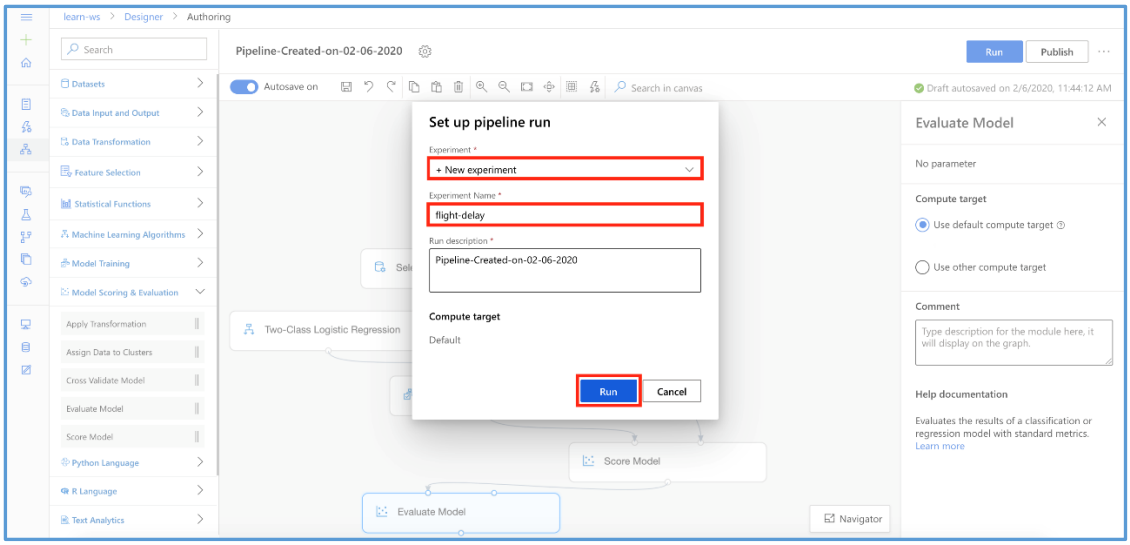
Task 1: Create Experiment and Submit Pipeline

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**.



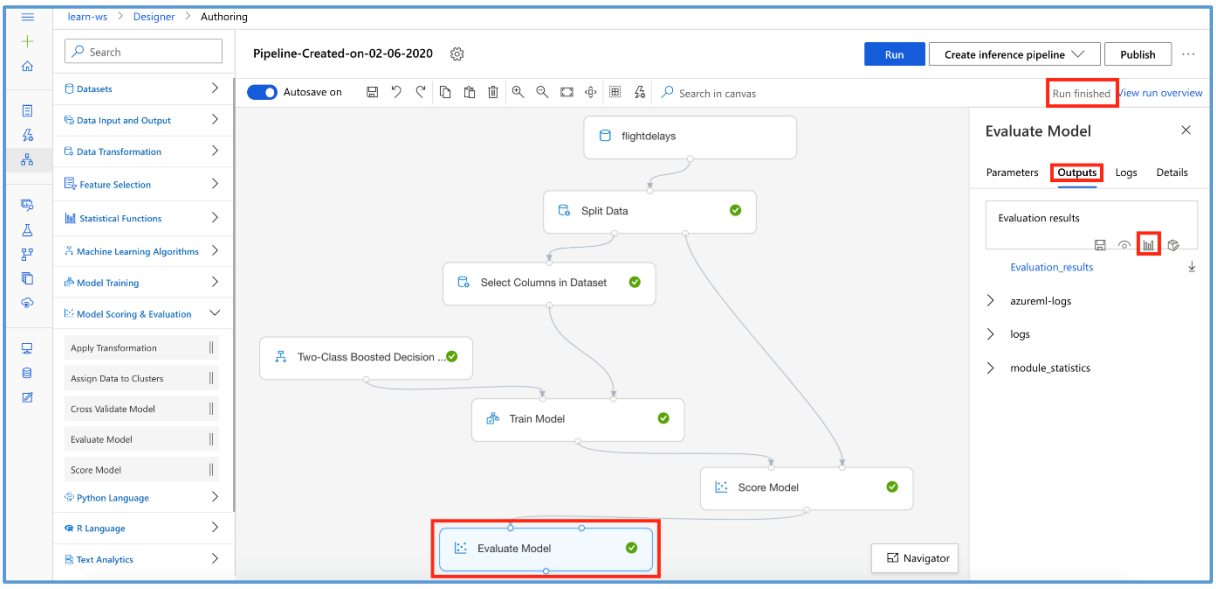
3. Wait for pipeline run to complete. It will take around **10 minutes** to complete the run.

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

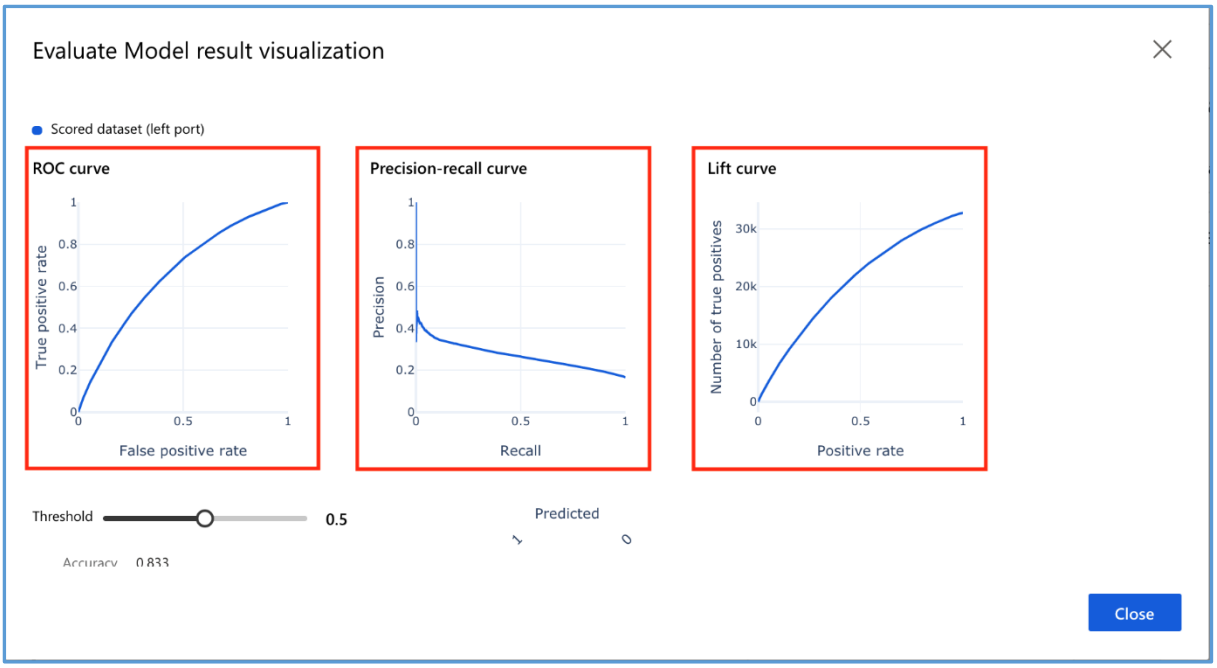
Task 1: Open the Result Visualization Dialog

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**

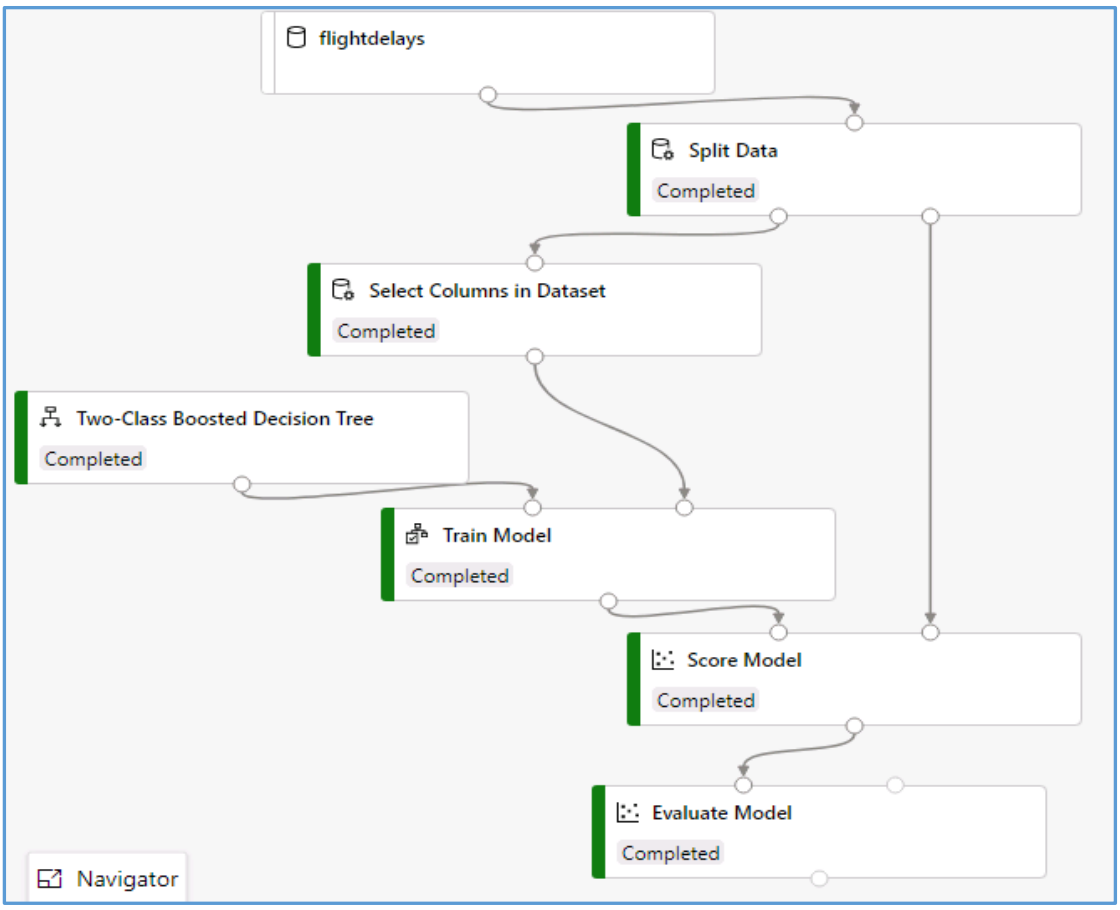
Score bin ↓	Positive exam...	Negative exam...	Fraction above thr...	Accur...	F1 Score	Precis...	Recall	Negat...	Negat...	Cumu...
(0.900,1.000]	0	0	0.000	0.834	0.000	1.000	0.000	0.834	1.000	0.000
(0.800,0.900]	1	2	0.000	0.834	0.000	0.333	0.000	0.834	1.000	0.000

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.

Chapter 32: Walkthrough – Train a Two – Class – Boosted – Decision Tree

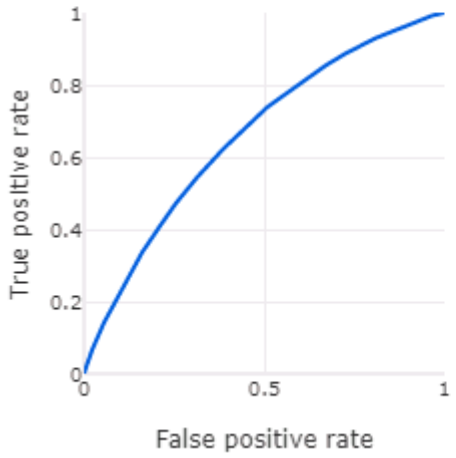
- Set the compute resource for the ML pipeline
- Register a new dataset with Azure ML. URL needs to be, specified for creating the dataset.
- Split the data using the "Relative Expression" splitting mode & Relational expression of "\"Month" < 10. Thus, all data less than 10 months would be Training data and remaining would be Test data.
- For the Training data, we are excluding a set of columns and for this, we are using "Select Column in dataset" task.
- ML algorithm used is Two Class Boosted decision tree. Here we are using "Number of trees constructed" = 100 instead of a single decision tree.
- Then we used the "Train Model", and connected the output of Algorithm to the first input and in the second input connect the output from "Select Column in dataset" task, which is the finalized version of the Training dataset.
- Then we used the "Score Model" task to score the Training and Test datasets.
- Finally, we used the "Evaluate Model" task for evaluation purpose.



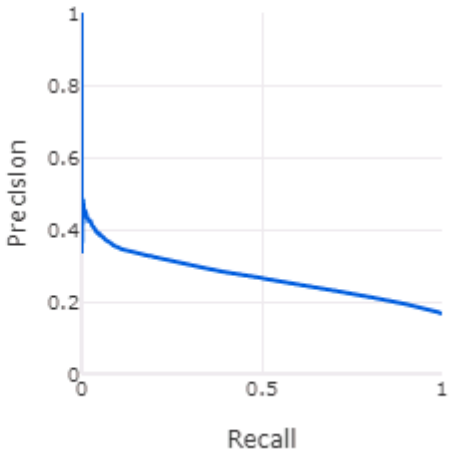
Evaluate Model result visualization

Scored dataset (left port)

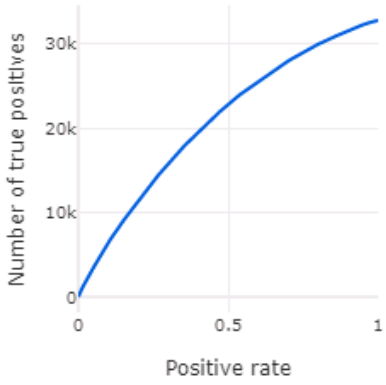
ROC curve



Precision-recall curve



Lift curve



Evaluate Model result visualization

Threshold 0.5

Accuracy	0.833
Precision	0.428
Recall	0.019
F1 Score	0.036
AUC	0.662

Predicted \ Actual	1	0
1	610	816
0	32149	163734

Chapter 33: Prelaunch Lab

Prelaunch your lab environment.

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

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

Chapter 34: Lab: Train a Simple Classifier with Automated ML

Lab Overview

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

Configuration options available in automated machine learning:

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

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

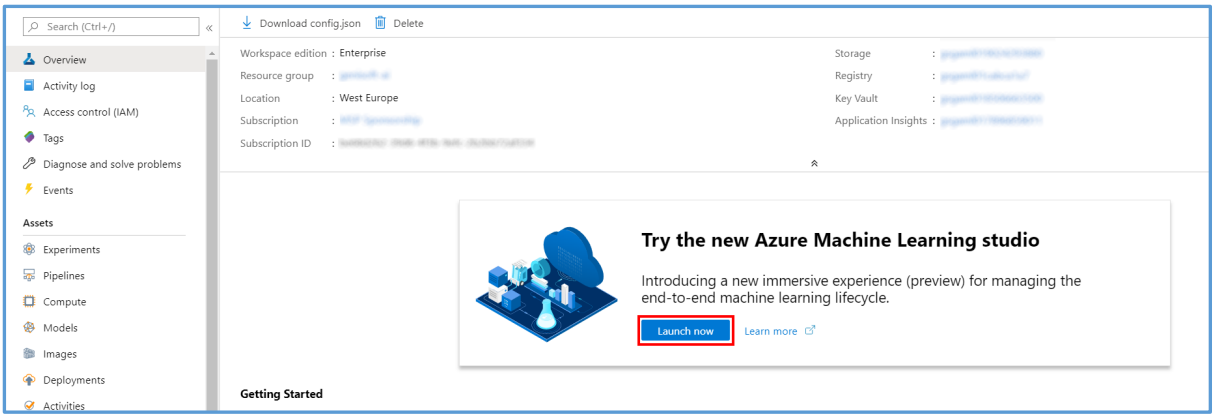
In this lab, you learn how to create, run, and explore automated machine learning experiments in the Azure Machine Learning Studio without a single line of code. As part of this lab, we will be using the Flight Delays data set that is, enhanced with the weather data. Based on the enriched dataset, we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

Exercise 1: Register Dataset with Azure Machine Learning studio

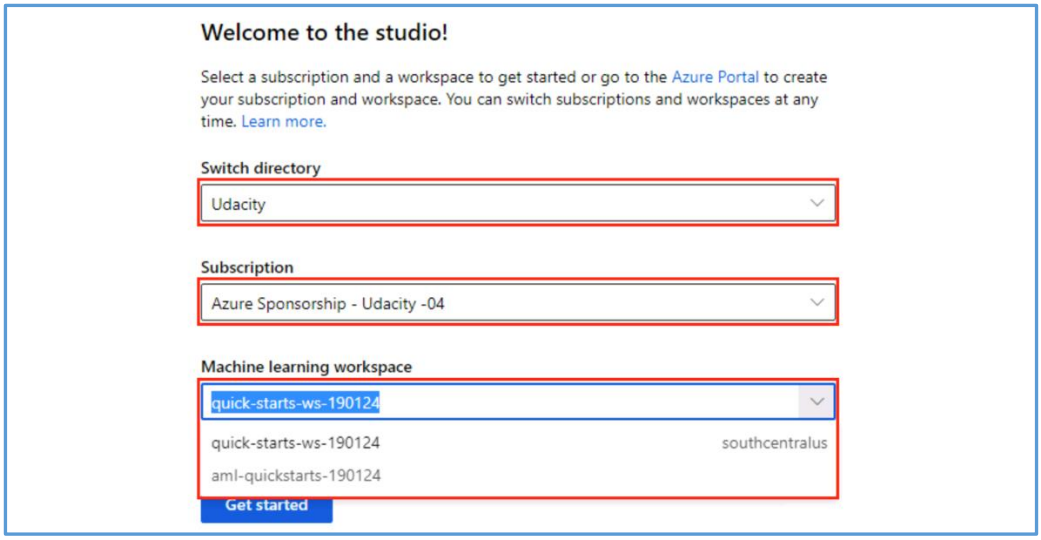
Task 1: Upload Dataset

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

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

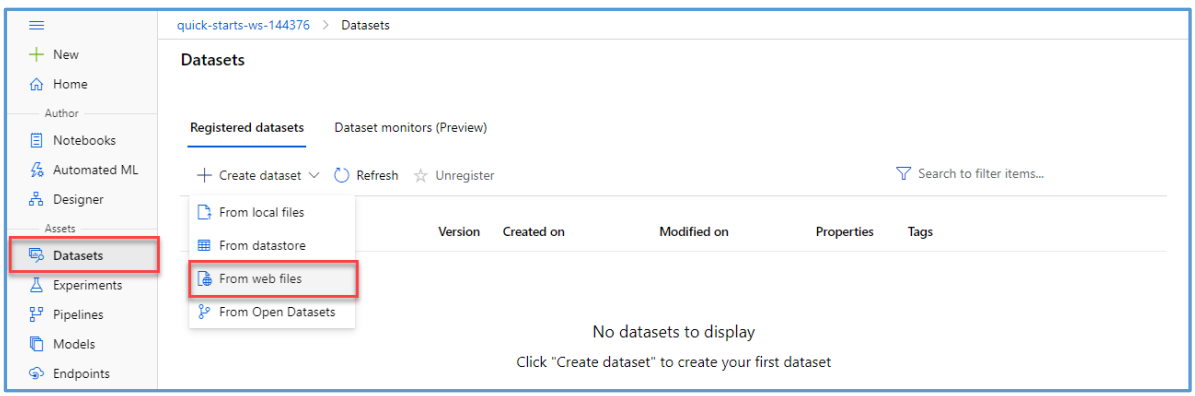


3. When you first launch the studio, you may need to set the directory and subscription. If so, you will see this screen:



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

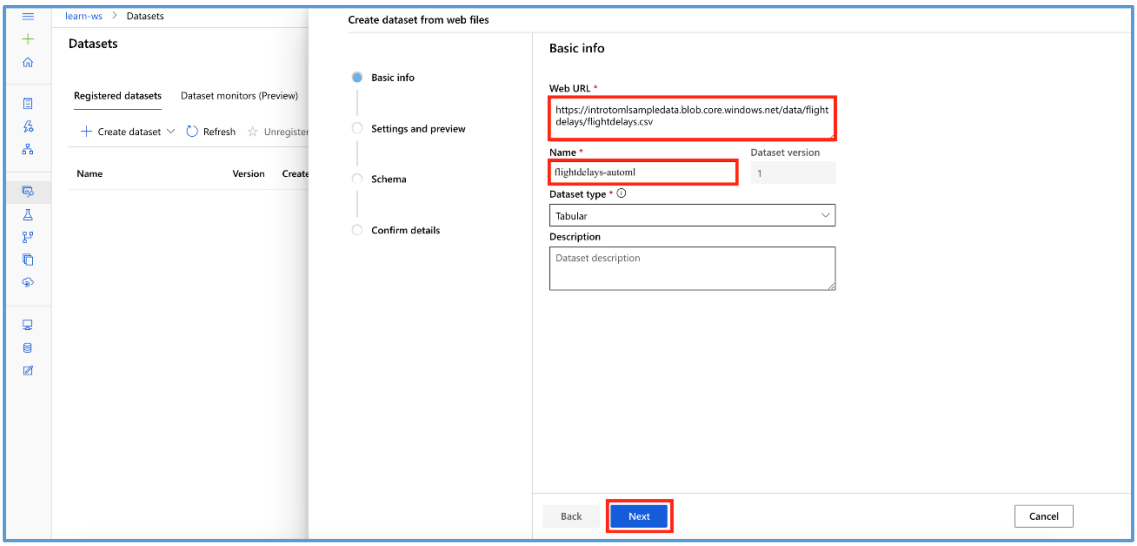
4. From the studio, select **Datasets, + Create dataset, from web files**. This will open the Create dataset from web files dialog on the right.



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

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

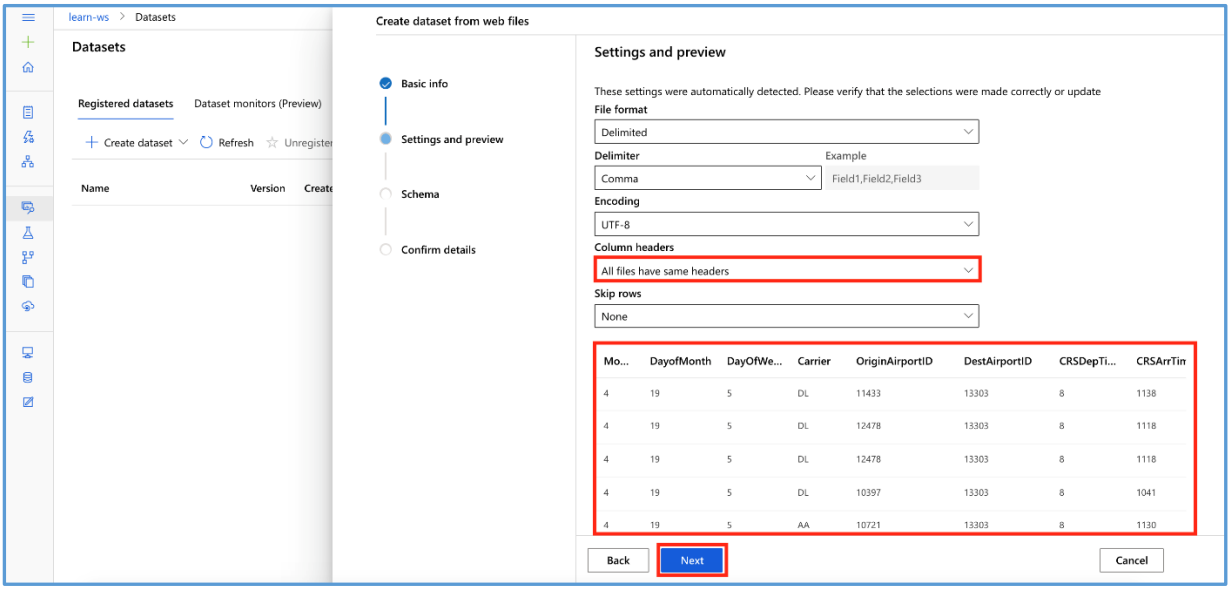
6. Provide `flightdelays-automl` as the Name, leave the remaining values at their defaults and select **Next**.



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**

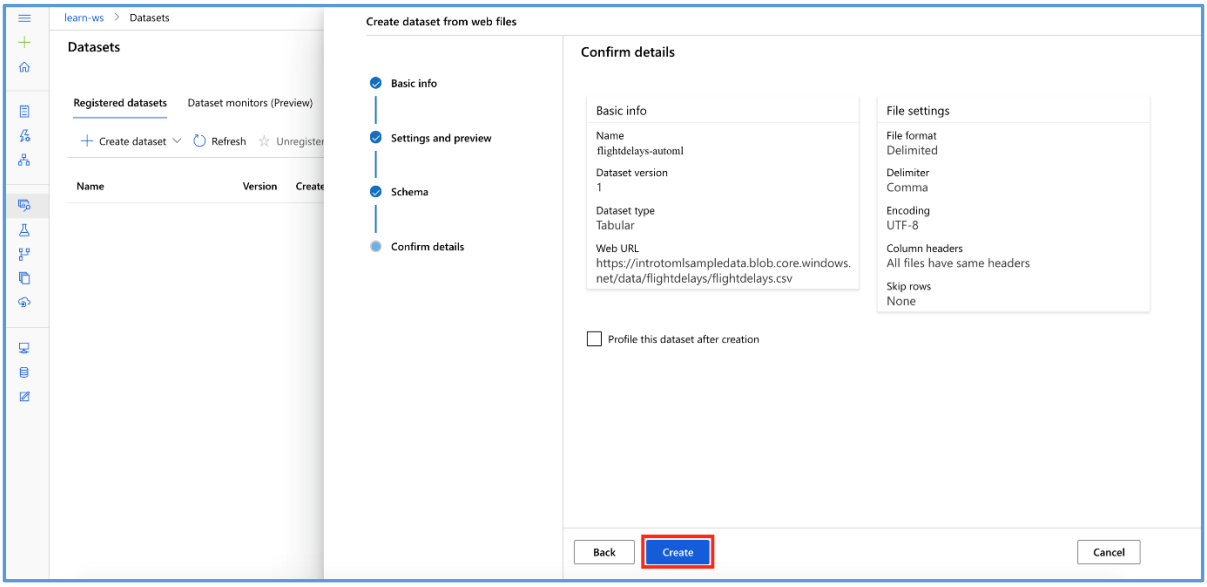


Task 3: Select Columns

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

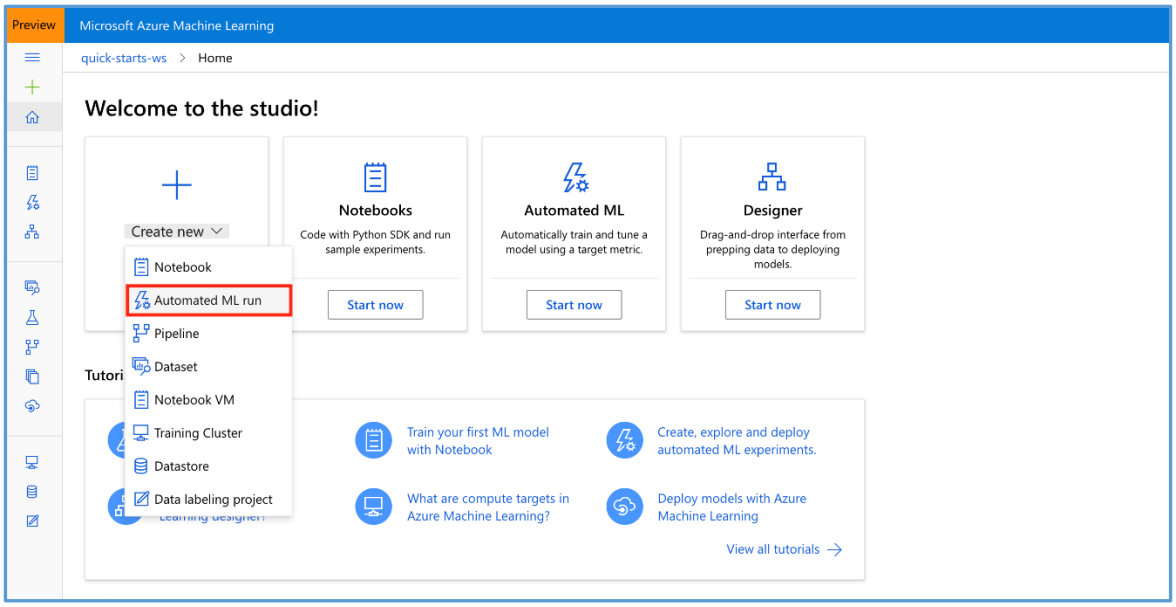
1. Confirm the dataset details and select **Create**



Exercise 2: Setup New Automated Machine Learning Experiment

Task 1: Create New Automated Machine Learning Experiment

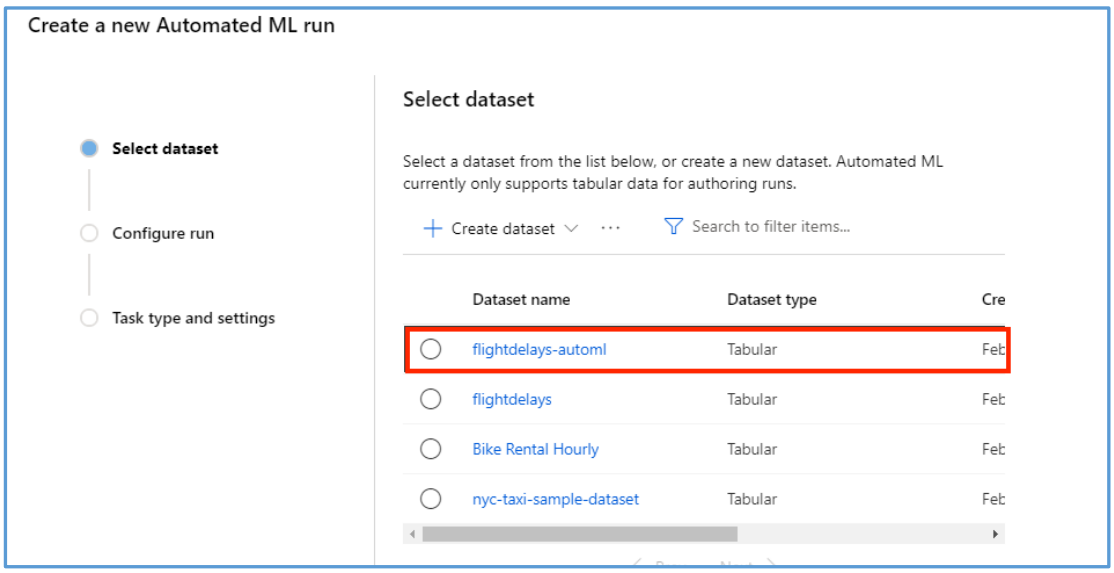
1. From the studio home, select **Create new, Automated ML run**



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

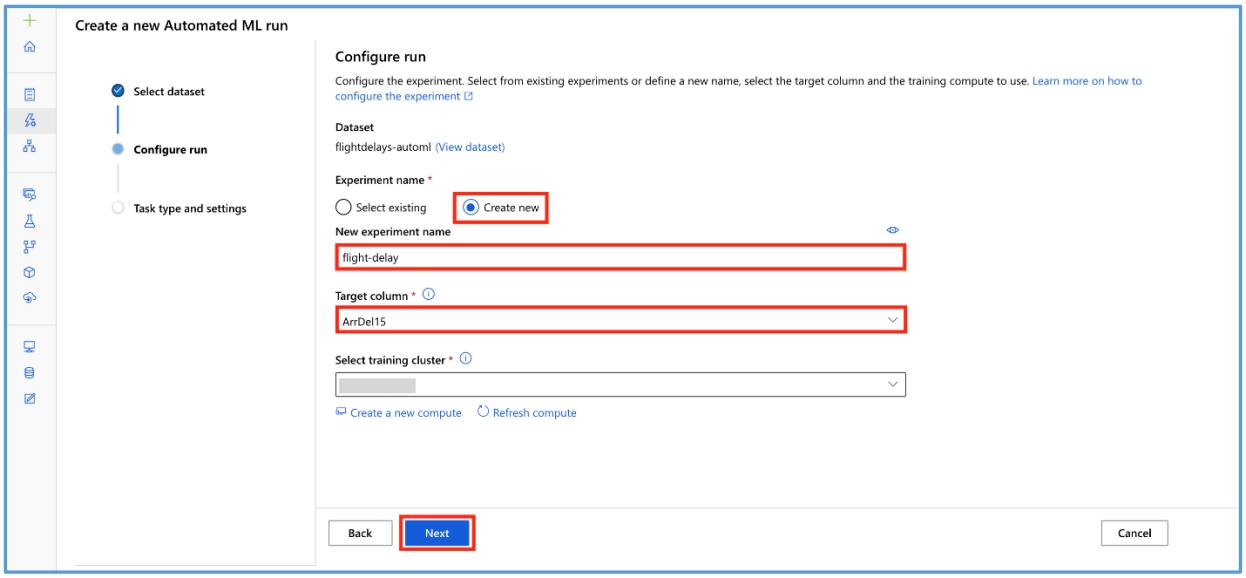
Task 2: Select Training Data

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



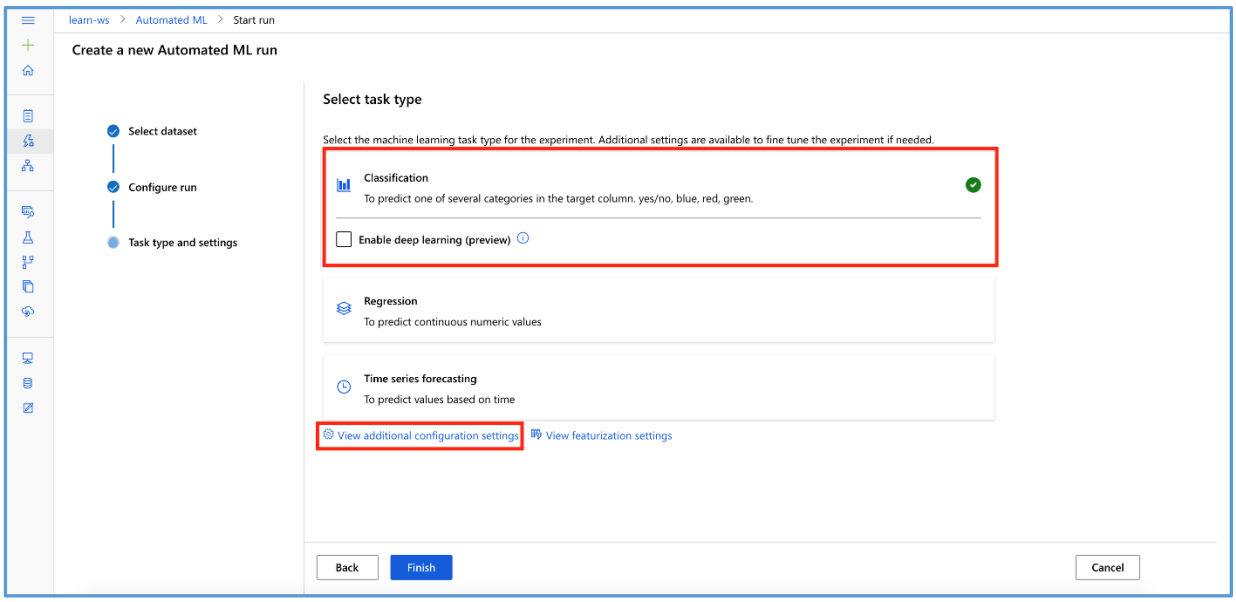
Task 3: Create a new Automated ML run

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

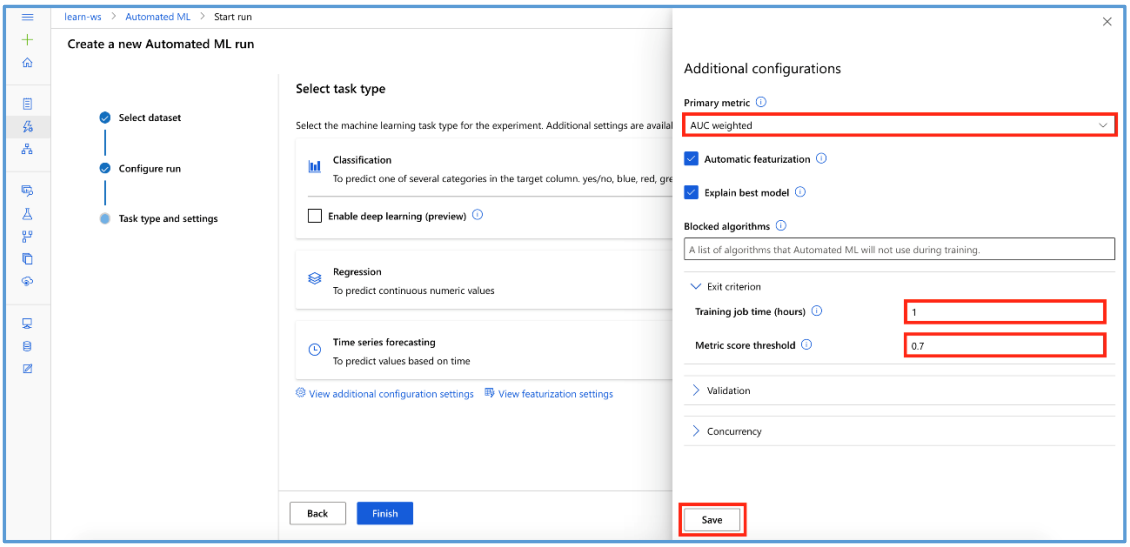
1. Select task type: **Classification**, and then select **View additional configuration settings**



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

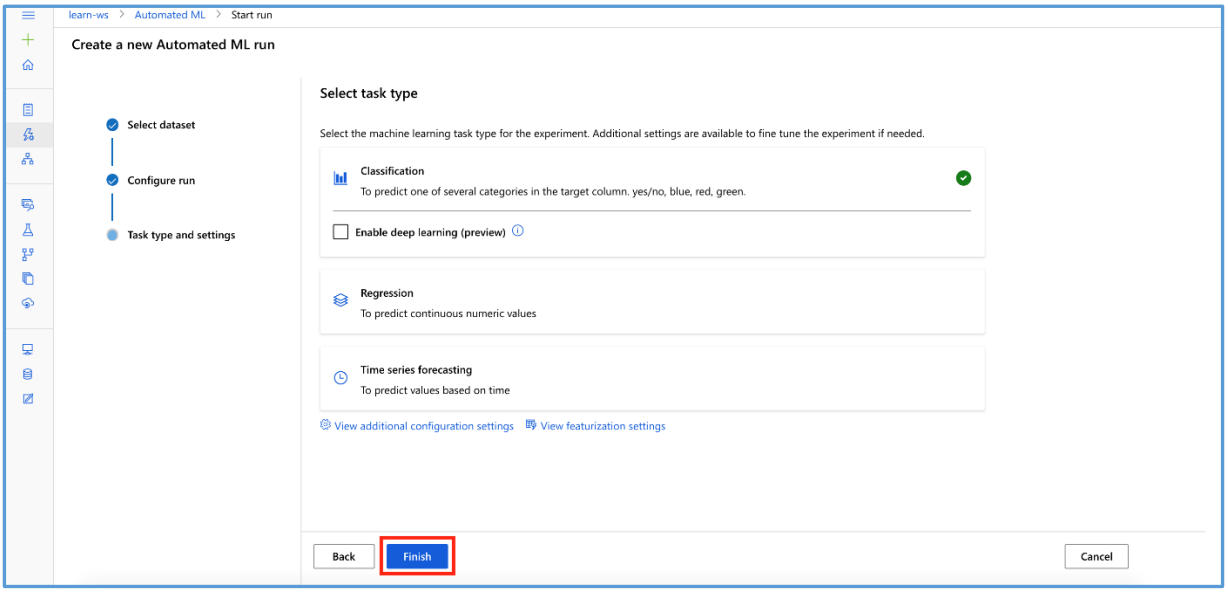


Note that we are setting a metric score threshold to limit the training time. In practice, for initial experiments, you will typically only set the training job time to allow AutoML to discover the best algorithm to use for your specific data.

Exercise 3: Start and Monitor Experiment

Task 1: Start Experiment

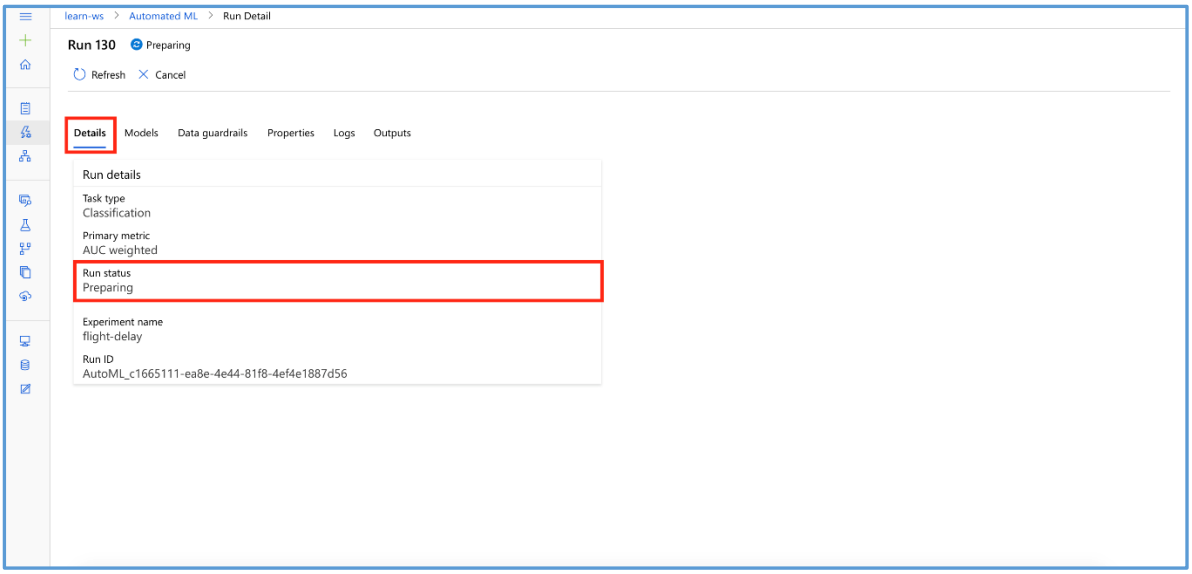
1. Select **Finish** to start running the experiment



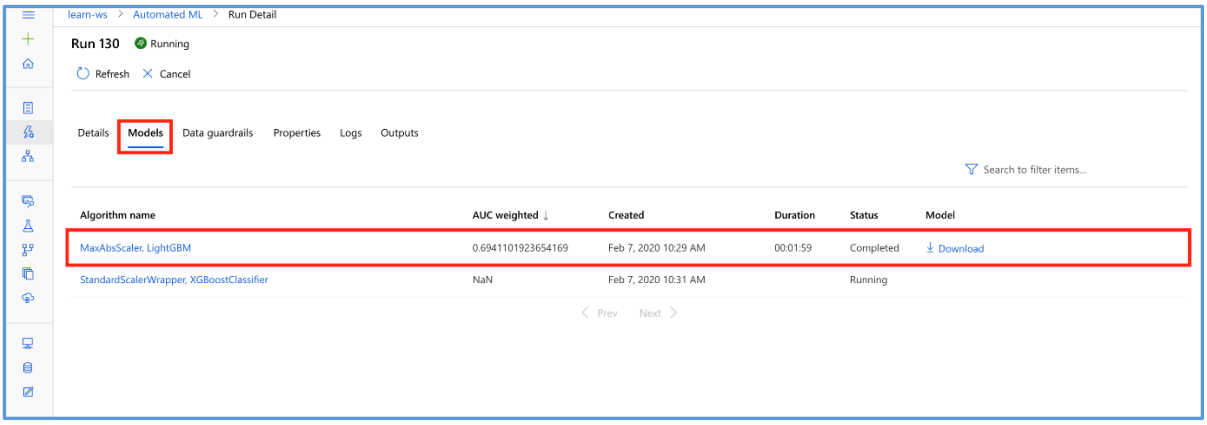
Task 2: Monitor Experiment

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

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

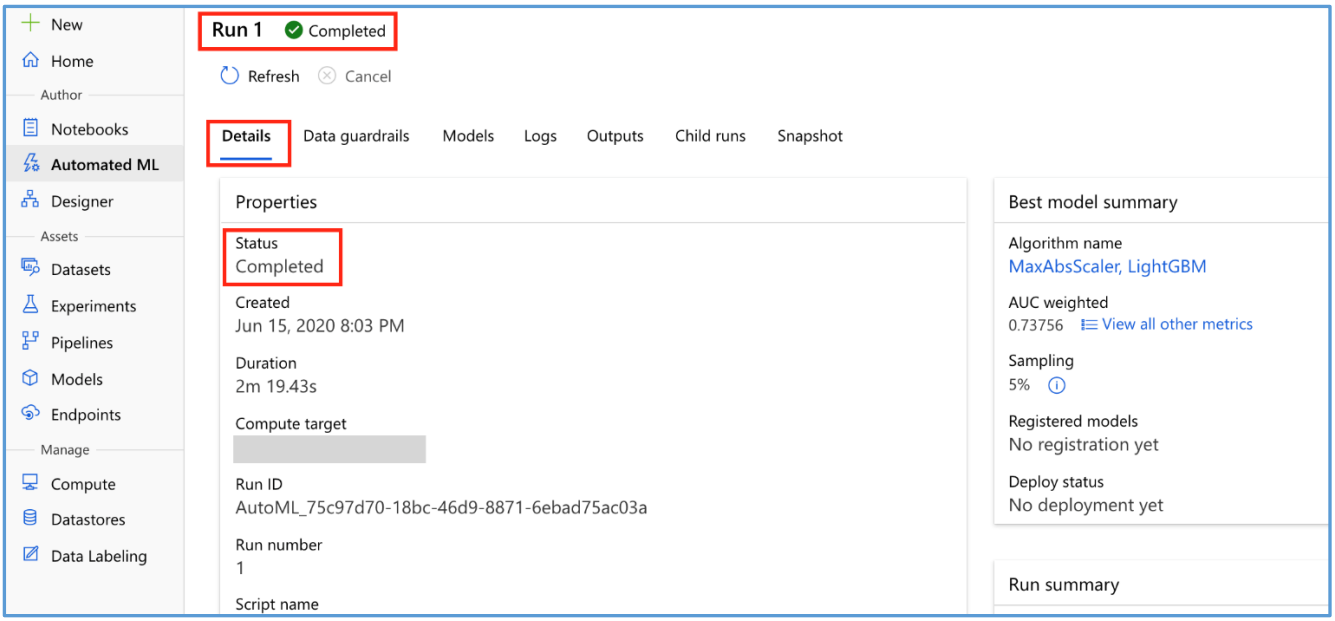


3. Select the **Models** tab, and observe the various algorithms the AutoML is evaluating. You can also observe the corresponding **AUC weighted** scores for each algorithm.



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**.

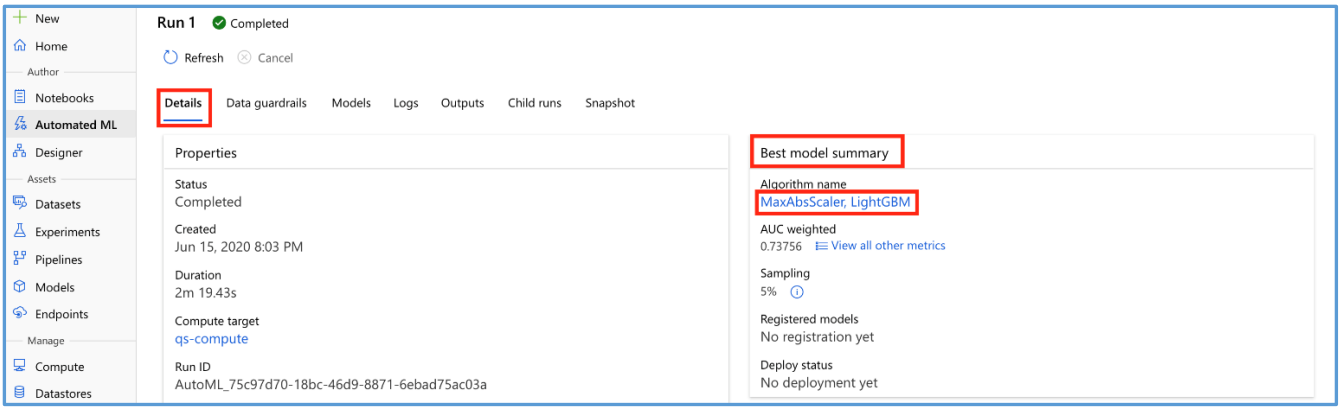


5. While you wait for the model training to complete, you can learn to view and understand the charts and metrics for your automated machine learning run by selecting Understand automated machine learning results.

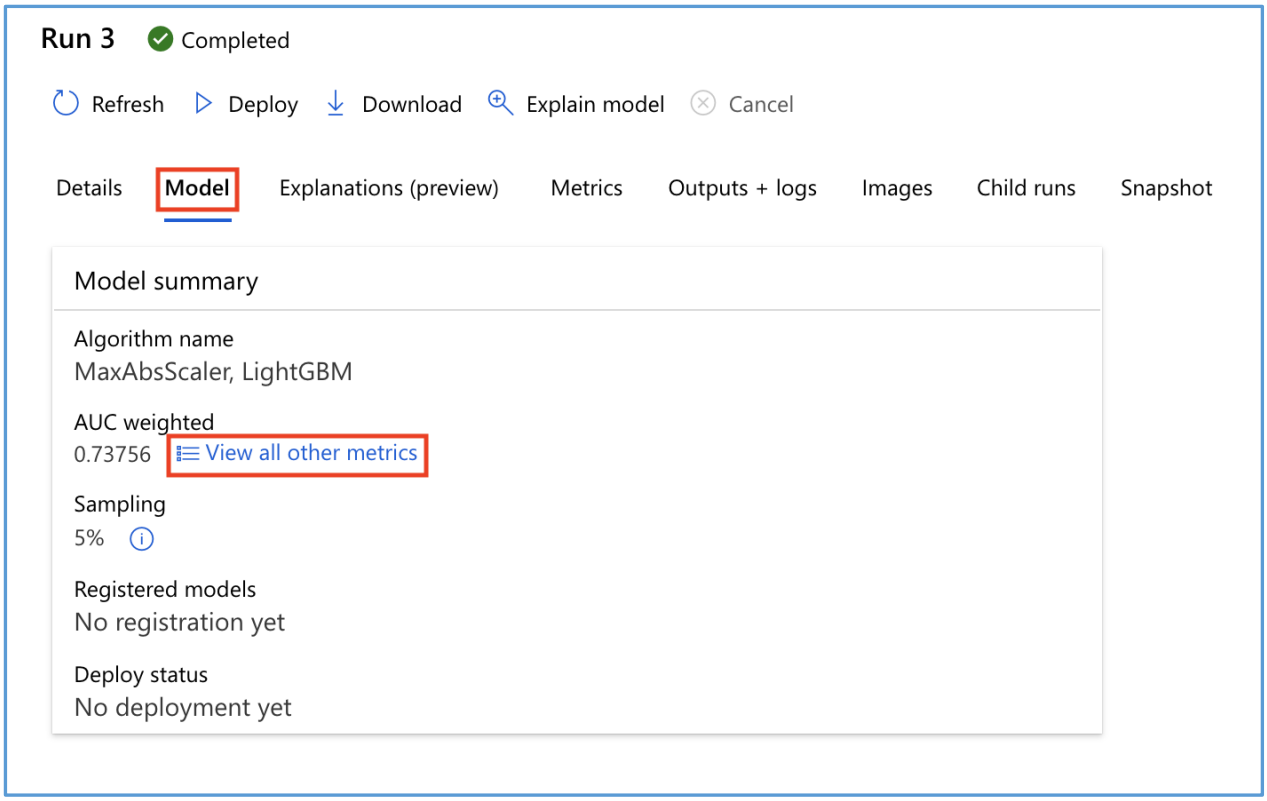
Exercise 4: Review Best Model's Performance

Task 1: Review Best Model Performance

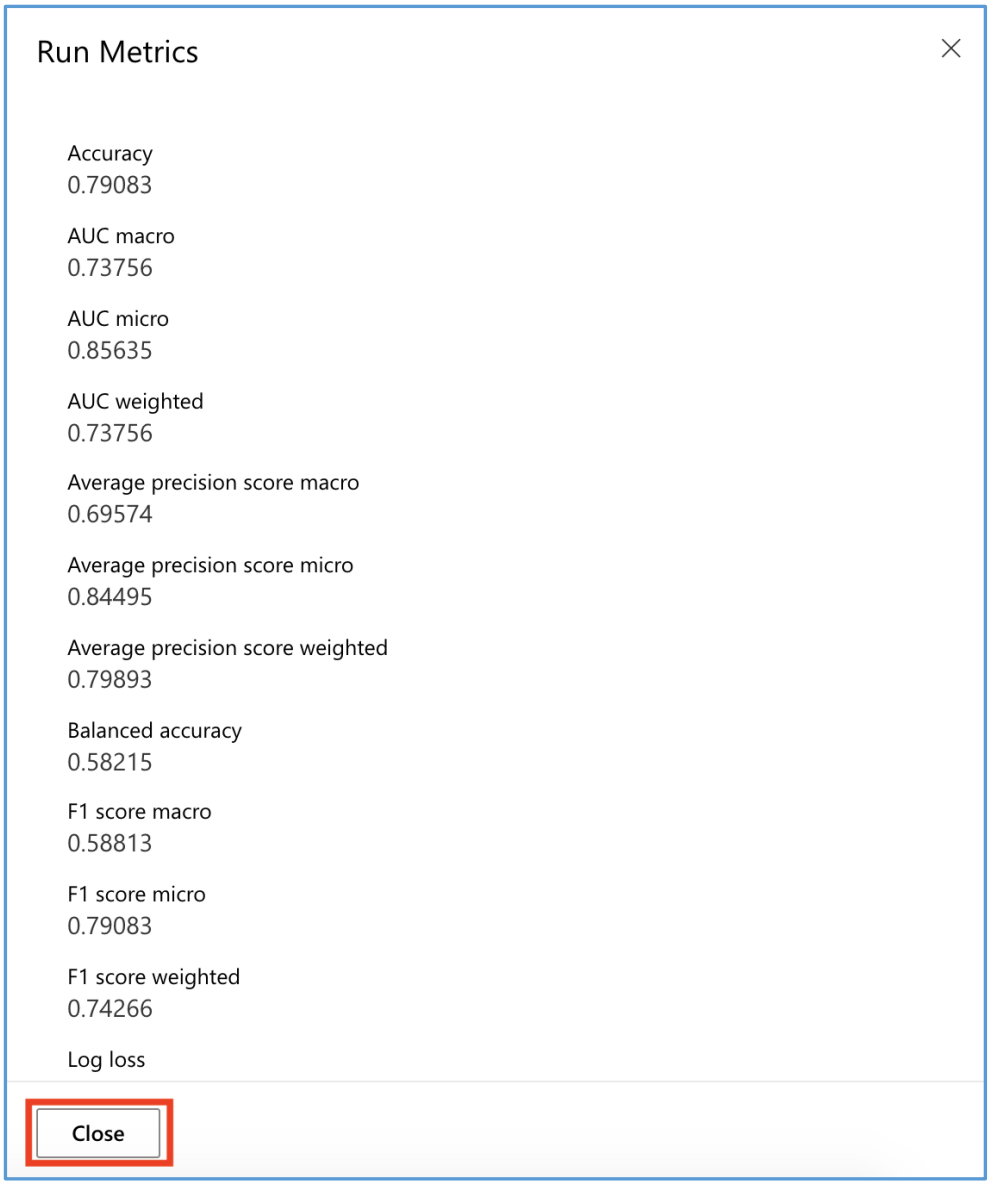
1. The Details tab shows the Best model summary. Next, select **Algorithm name** to review the model details.



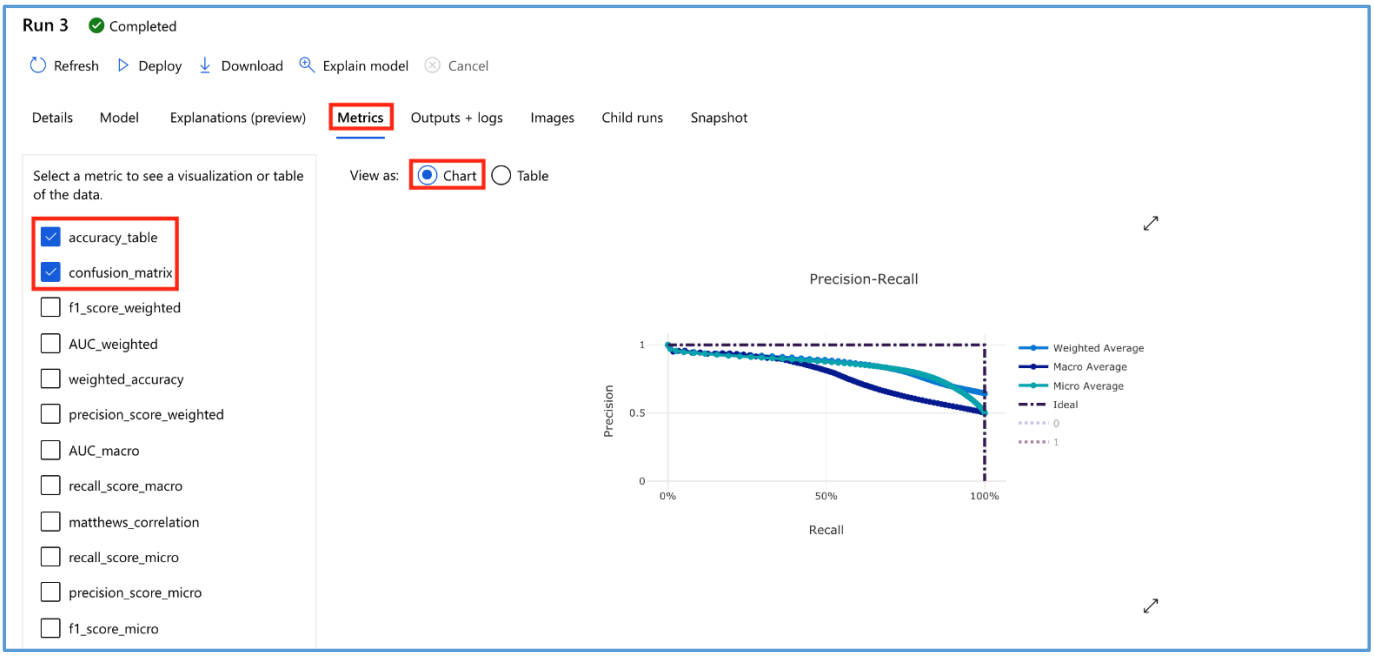
2. From the Model details tab, to view the various metrics to evaluate the best model performance, select **View all other metrics**.



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.

Chapter 35: Walkthrough: Train a Simple Classifier with Automated ML

- Set the compute resource for the ML pipeline
- Register a new dataset with Azure ML as part of Azure AutoML experiment. URL needs to be, specified for creating the dataset. Few columns need to be, excluded while getting the final form of the dataset.
- Target column for prediction needs to be, specified “ArrDel15” and we need to specify this to be classification task.
- Metric selected was AUC weighted & threshold for the metric is 0.7 where exploration process finishes.
- Once experiment is, run Auto ML started exploring various options of hyperparameters and ML algorithms as part of the process and finally it ends up with the best option. “Recommended Model”.

Chapter 36: Lesson Summary

In this lesson, you have learned to perform the essential **data preparation and management** tasks involved in machine learning:

- Data importing and transformation
- The use of *datastores* and *datasets*
- Versioning
- Feature engineering
- Monitoring for data drift

The second major area we covered in this lesson was **model training**, including:

- The core model training process
- Two of the fundamental machine learning models: *Classifier* and *regressor*
- The model evaluation process and relevant metrics

The final part of the lesson focused on how to get better results by using multiple trained models instead of a single one. In this context, you learned about **ensemble learning** and **automated machine learning**. You have learned how the two differ; yet apply the same general principle of "strength in numbers". In the process, you trained an ensemble model (a decision forest) and a straightforward classifier using automated Machine Learning.