**Graphical user interface, text, application

Description automatically generated**

***Agenda:***

* Create a new pipeline
* Set the default compute target
* Import data
* Visualize the data
* Prepare data
* Train a machine learning model
* Split the data
* Prepare data for training model
* Add Score Model component
* Add Evaluate Model component
* View scored labels
* Evaluate models
* Evaluation Results
* Submit the pipeline
* Clean up resources
* Delete everything
* Delete individual assets

**Create a new pipeline**

Azure Machine pipelines organize multiple machine learning and data processing steps into a single resource. Pipelines let you organize, manage, and reuse complex machine learning workflows across projects and users.

To create an Azure Machine Learning pipeline, you need an Azure Machine Learning workspace.

* **Create a new workspace**

You need an Azure Machine Learning workspace to use the designer. The workspace is the top-level resource for Azure Machine Learning, it provides a centralized place to work with all the artifacts you create in Azure Machine Learning.

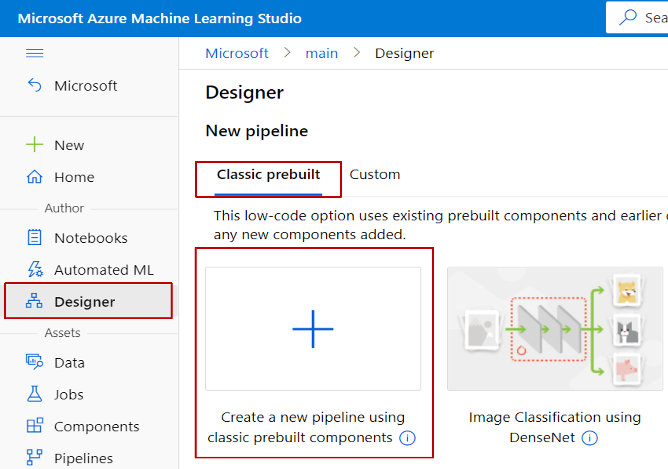
* **Create the pipeline**

**Designer** supports two types of components, **classic prebuilt components** and **custom components**. These two types of components are not compatible.

**Classic prebuilt components** provide prebuilt components majorly for data processing and traditional machine learning tasks like regression and classification. This type of component continues to be supported but will not have any new components added.

**Custom components** allow you to provide your own code as a component. It supports sharing across workspaces and seamless authoring across Studio, CLI, and SDK interfaces.

* Here, applying **classic prebuilt components**.
* Select **Designer 🡪 Classic prebuilt**



* Select **Create a new pipeline using classic prebuilt components**.
* Click the pencil icon beside the automatically generated pipeline draft name, rename it to *Bike Rental Hour counts prediction*. The name doesn't need to be unique.



**Set the default compute target**

A pipeline jobs on a compute target, which is a compute resource that's attached to your workspace. After you create a compute target, you can reuse it for future jobs.

Note: Attached compute is not supported, use [**compute instances or clusters**](https://learn.microsoft.com/en-us/azure/machine-learning/concept-compute-target#azure-machine-learning-compute-managed) instead.

You can set a **Default compute target** for the entire pipeline, which will tell every component to use the same compute target by default. However, you can specify compute targets on a per-module basis.

* Select  **Settings** to the right of the canvas to open the **Settings** pane or perform under **pipeline authoring editor**.
* Select **Create Azure ML compute instance**.

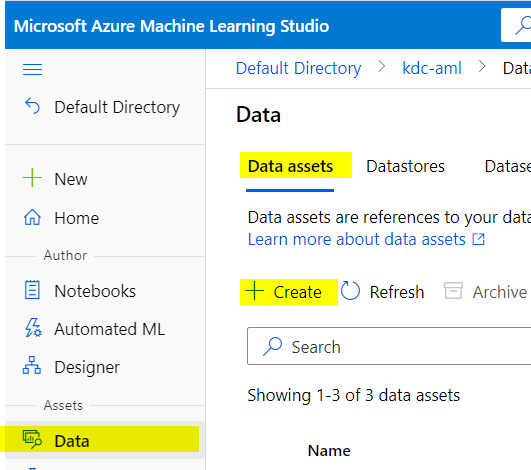
If you already have an available compute target, you can select it from the **Select Azure ML compute instance** drop-down to run this pipeline.

* Enter a name for the compute resource.
* Select **Create**.

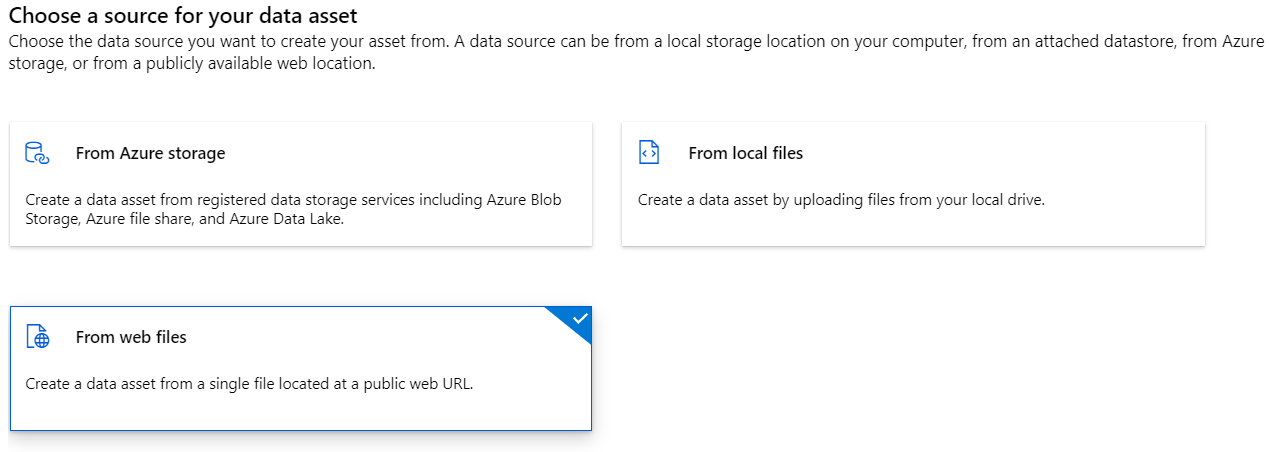
**Import data**

Using **Bike Rental Hour data (Raw)**.

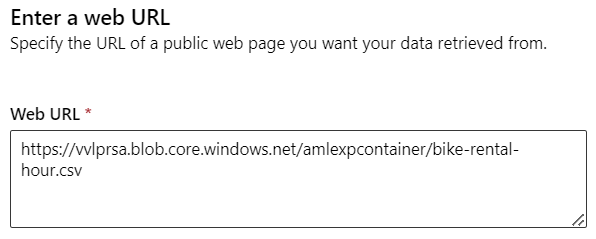
* Select **Data 🡪 Create** under **Data assets**



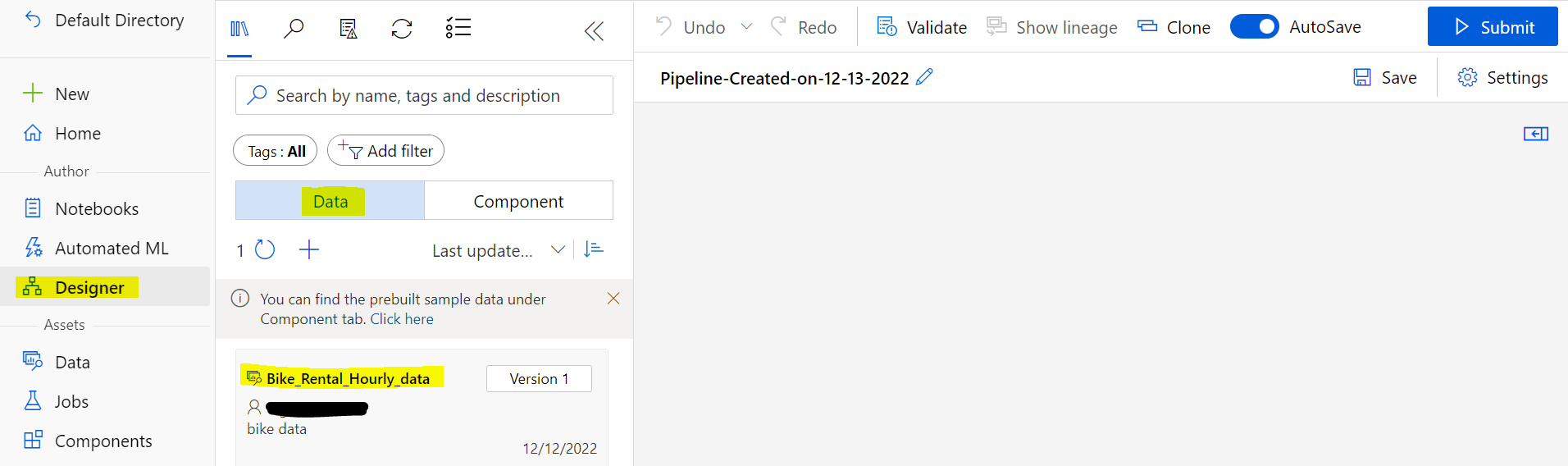
* Download bike-rental-hour.csv and create a blob object and copy URL
* Create a new data asset and **Name** it **Bike\_Rental\_Hourly\_data** and choose Type as **File**. Click **Next**.



* Choose **From web files** under **Data Source** and give the copied URL from previous step as **Web URL**. Click **Next**.



* To the left of the pipeline canvas is a palette of datasets and components. Select **Data 🡪 Bike\_Rental\_Hourly\_data**.



**Visualize the data**

You can visualize the data to understand the dataset that you'll use.

1. Right-click the **Bike\_Hourly\_Rental\_data (Raw)** and select **Preview Data**.
2. Select the different columns in the data window to view information about each one.

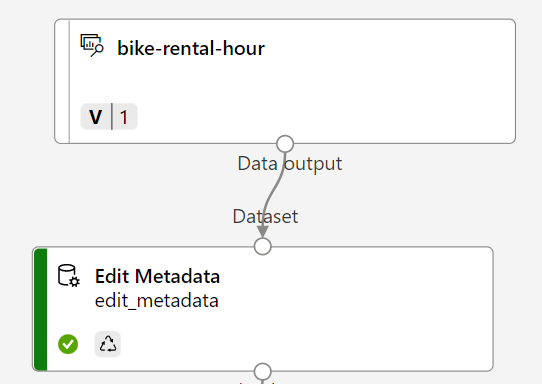
Each row represents an Bike Rental information, and the variables associated with each Bike Rental appear as columns. There are 17379 rows and 17 columns in this dataset.

## **Prepare data**

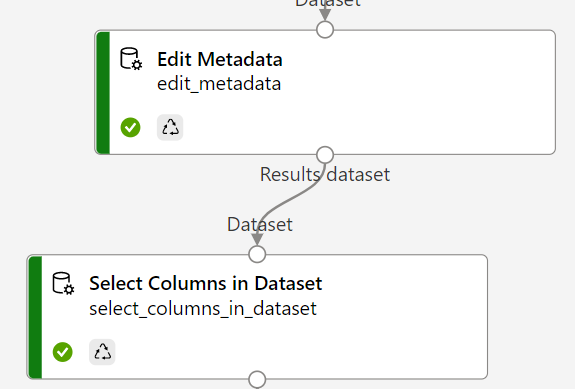
* Drag and drop on the canvas, the available **Bike\_Rental\_Hourly\_data** under the **Data**.
* In the datasets and component palette to the left of the canvas, click **Component** and search for the **Edit Metadata** module (Component/Data transformation category).

Connect the module to the dataset, and double click on the **Edit Metadata** to open Edit Metadata details.

Click on **Edit column** and add the **season and weathersit** column for **Column names**.



* Drag the **Select Columns in Dataset** (Data transformation category) component onto the canvas. connect this to the Edit Metadata module, and double click on the **Select Columns in Dataset**.
* Click **Edit Columns**, Configure the Select Columns in Dataset module as follows:
* **Include**: All columns
* Select **+**
* **Exclude** Column names: **instant, dteday, casual, registered**
* Select **Save**



* Drag and drop the **Execute Python Script** module (Python Language category) and connect it with the **Select Columns in Dataset**. (Make sure the connector is connected to the very first input of the Execute Python Script module)

Double Click on the **Execute Python Script** module,

Use 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. The script adds 12 new columns to the dataset containing the number of bikes that were rented in each of the previous 12 hours.

To add the following **python script,** click **Edit code** and paste copied script and **save** it:

```````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

# 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

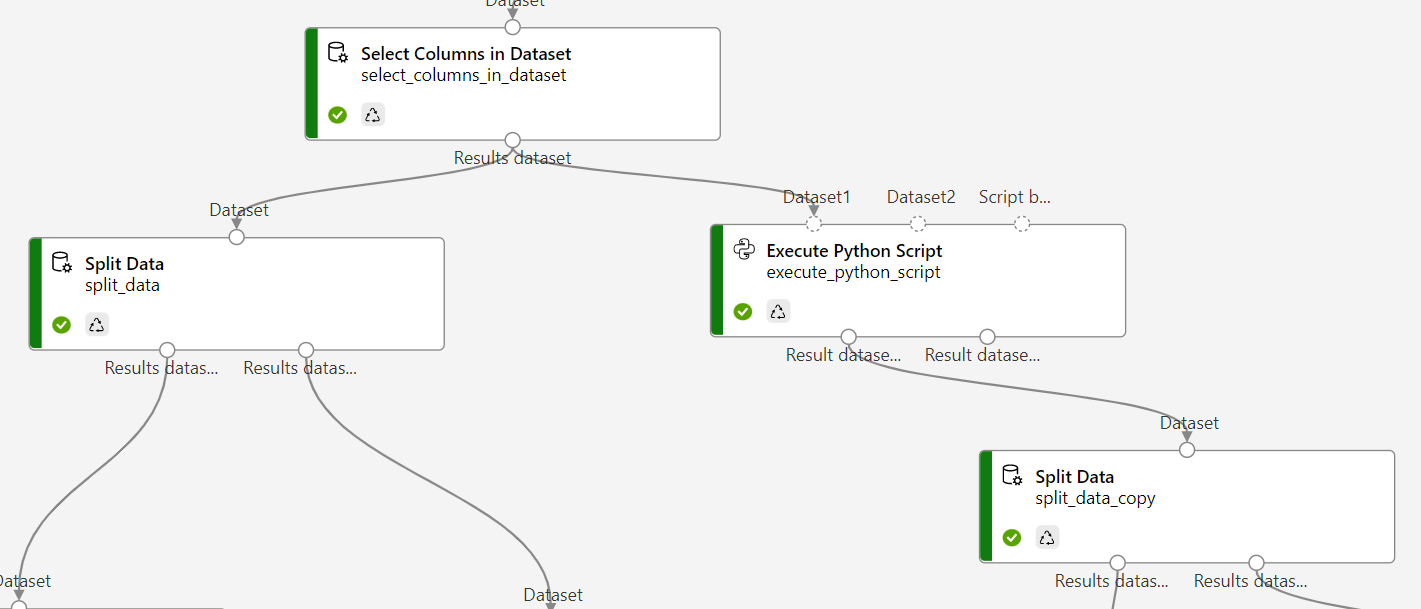
return dataframe1,

```````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

**Train a machine learning model**

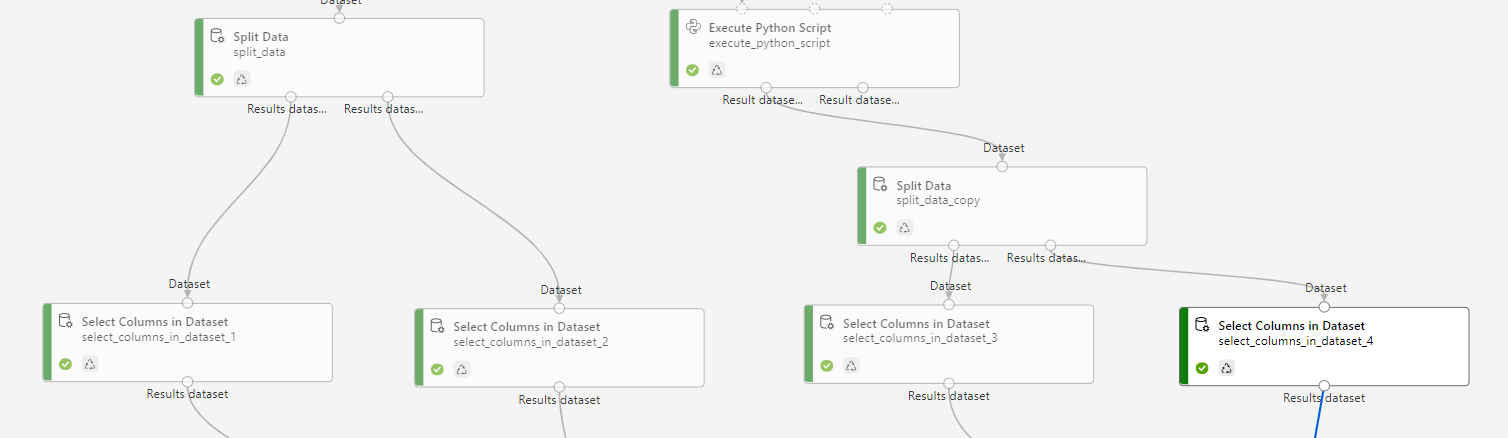
### Split the data

* Drag the **Split Data** component (Data Transformation module) to the pipeline canvas, 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
* Select the **Split Data** module block and use the menu buttons to Copy and Paste it on the canvas. Connect the secondly pasted **Split Data** to the output of the **Execute Python Script** module, which is the featured B set.

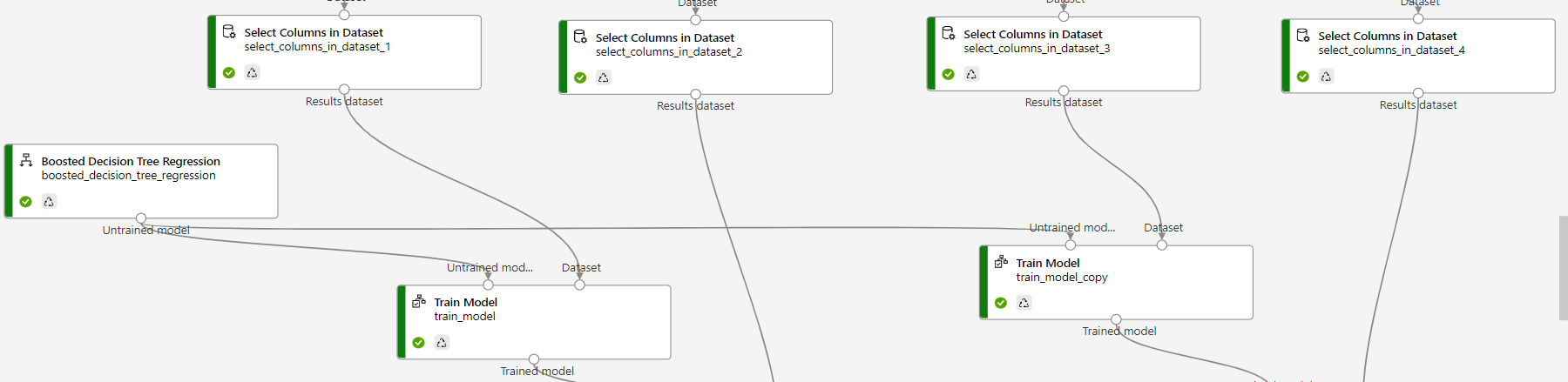


**Prepare data for training model**

* Drag and drop **Select columns in Dataset** module (Data transformation category), create **four identical modules to exclude the yr column** from all the outputs of **Train Model** and  **Score Model** module (these two model modules will be seen later).



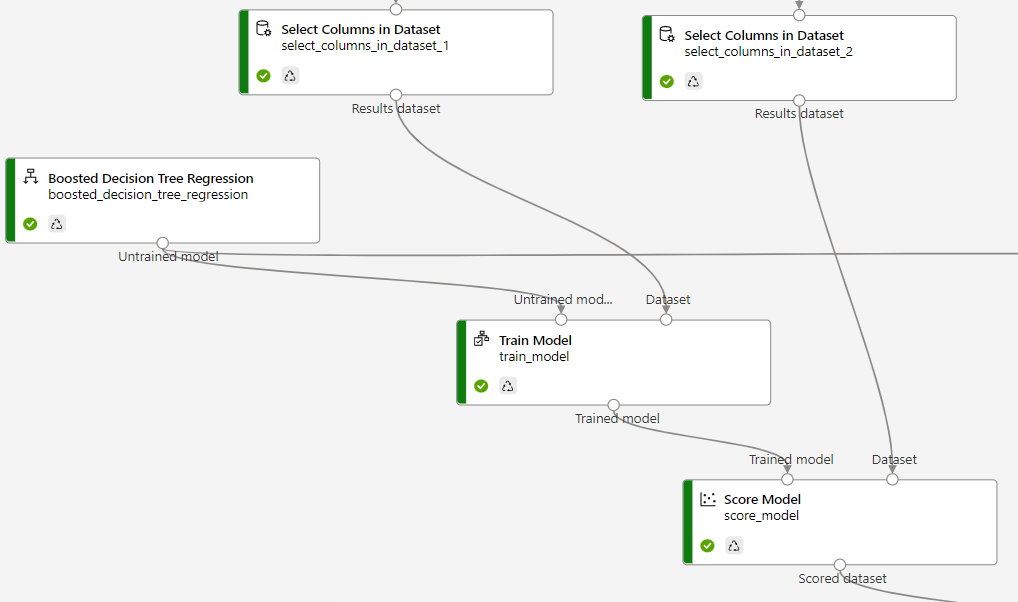
* Drag and drop **Boosted Decision Tree Regression** (Machine Learning Algorithms, Regression category) component to the pipeline canvas.
* Drag and drop **Train model** (Model training category) component and enter the **cnt** column in the **Label column field.**
* Connect the output of the **Linear Regression** component to the left input of the **Train Model** component.
* Connect the selected columns excluding target column of the **Select columns in Dataset\_1** component to the right input of the **Train Model** component.
* Select the **Train Model** module block and use the menu buttons to Copy and Paste it on the canvas. Connect the secondly pasted **Train Model** to the output of the **Select columns in Dataset\_3** module.
* Connect the **Boosted Decision Tree Regression** module as the first input and the training dataset as the second input to the **Train Model** module.



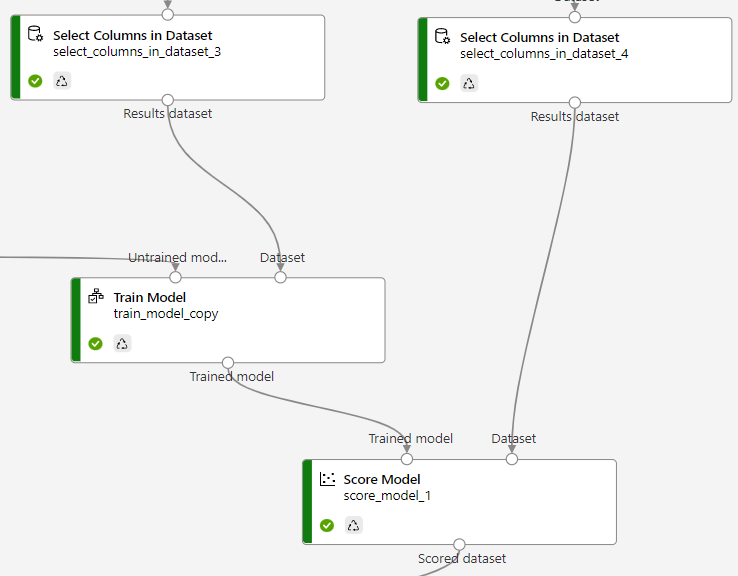
**Add Score Model component**

After training the model using \"yr" == 0 of the data, now using it to score the other remaining “yr”==1 to see how well the model functions.

* Use two **Score Model** modules (Model Scoring and Evaluation category) and link on the input of two trained models and the two test datasets.
* Drag the **Score Model** component to the pipeline canvas.
* Connect the output of the **Train Model** component to the left input port of **Score Model**. Connect the test data output, **Select columns in Dataset\_2** (right port) to the right input port of **Score Model**.



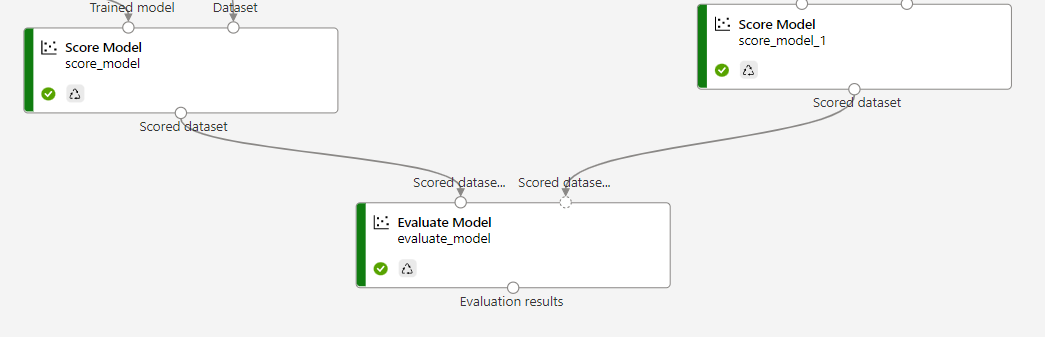
* Again, drag and drop the **Score Model** component to the pipeline canvas. Connect this to the output of the **Train Model** component to the left input port of **Score Model**. Connect the test data output, **Select columns in Dataset\_4** (right port) to the right input port of **Score Model**.



**Add Evaluate Model component**

Use the **Evaluate Model** component to evaluate how well your model scored the test dataset.

* Drag and drop the **Evaluate Model** module (Model Scoring and Evaluation category) and connect it to the **two Score Model modules – (score\_model & score\_model\_1)**.



**Submit the pipeline**

Now that pipeline is all setup, and submitting a pipeline job to train the machine learning model. You can submit a valid pipeline job at any point, which can be used to review changes to your pipeline during development.

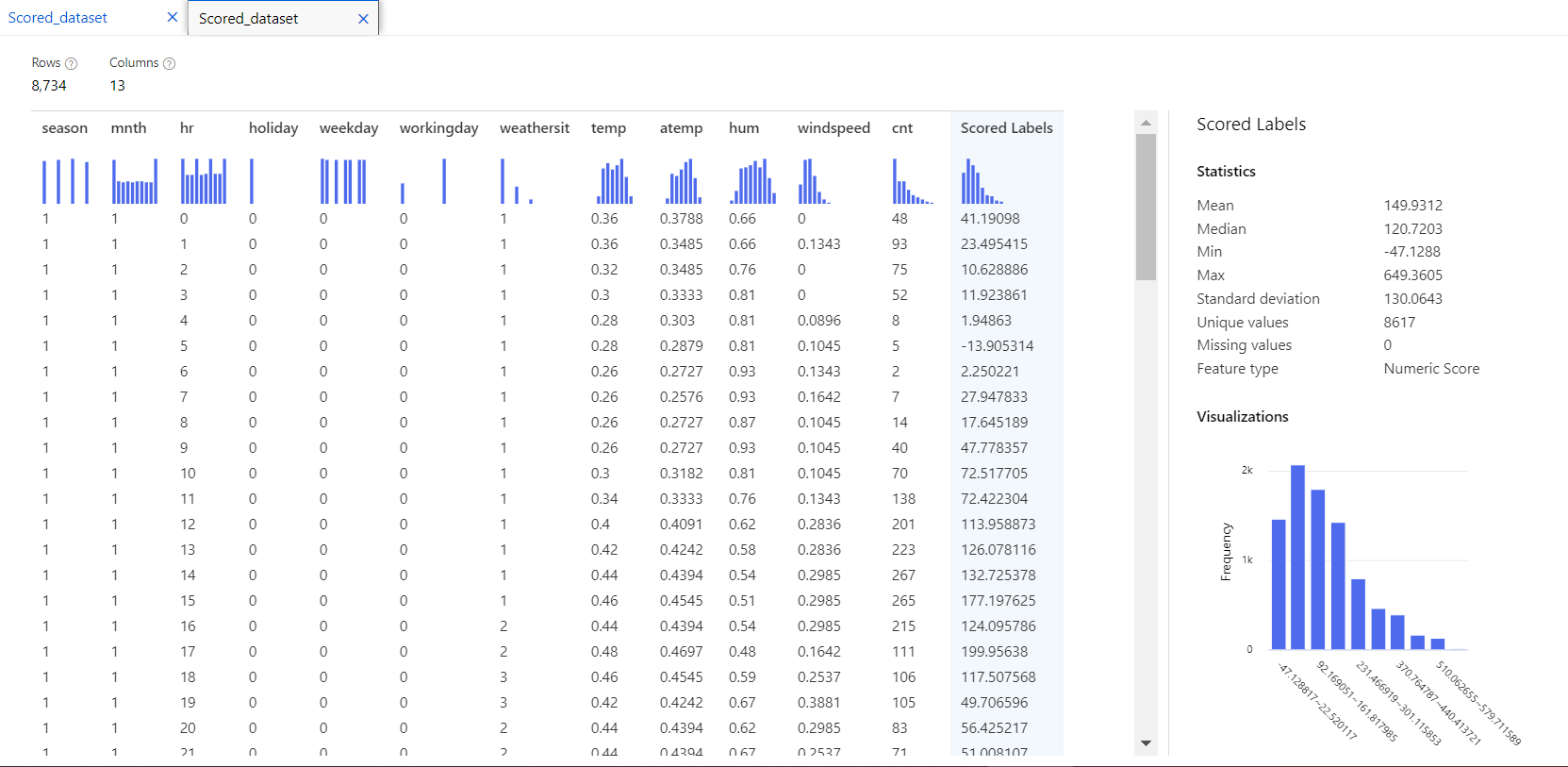
* Select **Submit** to open **the Setup pipeline job** run editor
* In the **Set up pipeline job** dialog box, select **Create new**.
* For **New experiment** Name, enter  **Biker Hourly Rental** Counts.
* Select **Submit**.
* You'll see a submission list in the left pane of the canvas, and a notification will pop up at the top right corner of the page. You can select the **Job detail** link to go to job detail page for debugging.

**View scored labels**

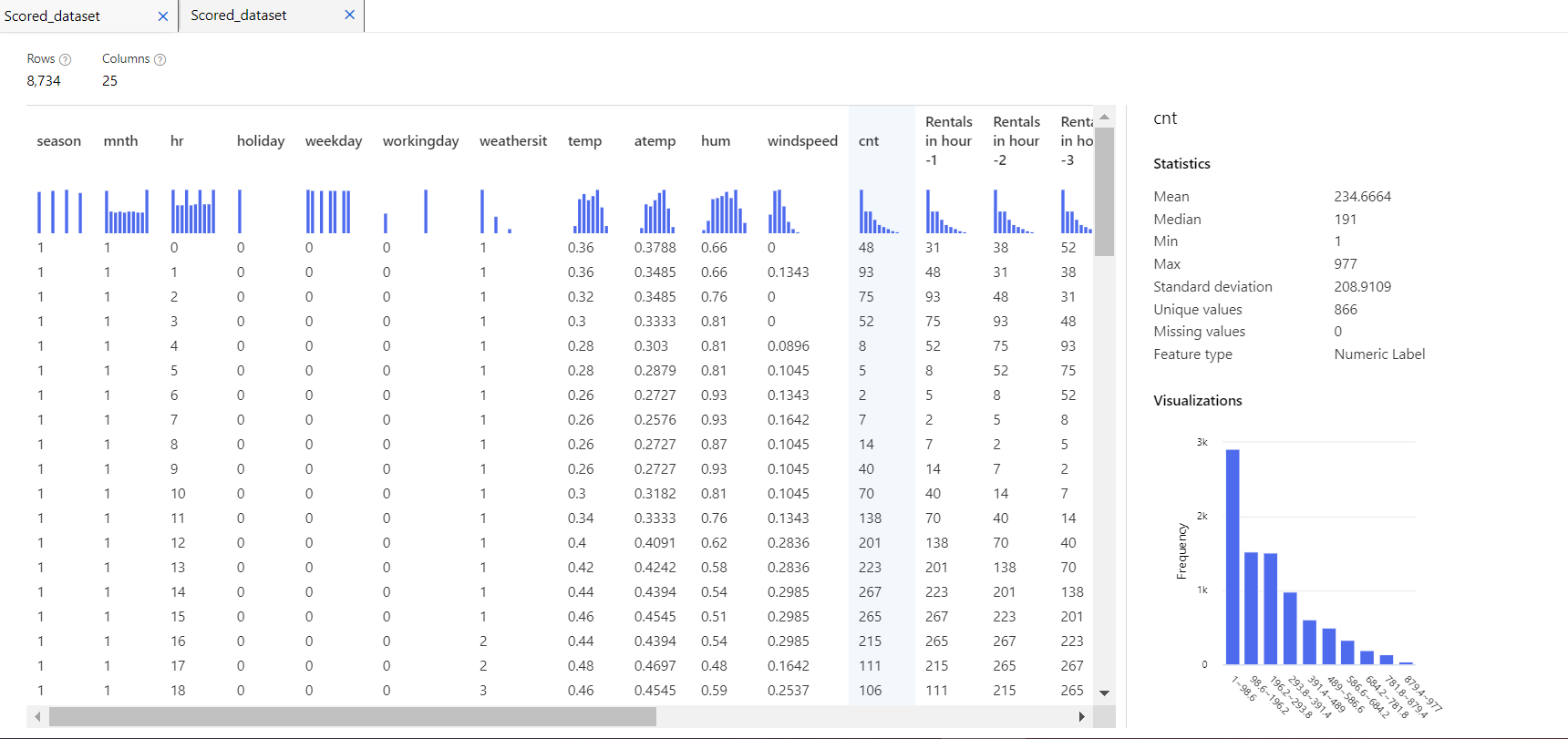
In the job detail page, you can check the pipeline job status, results and logs.

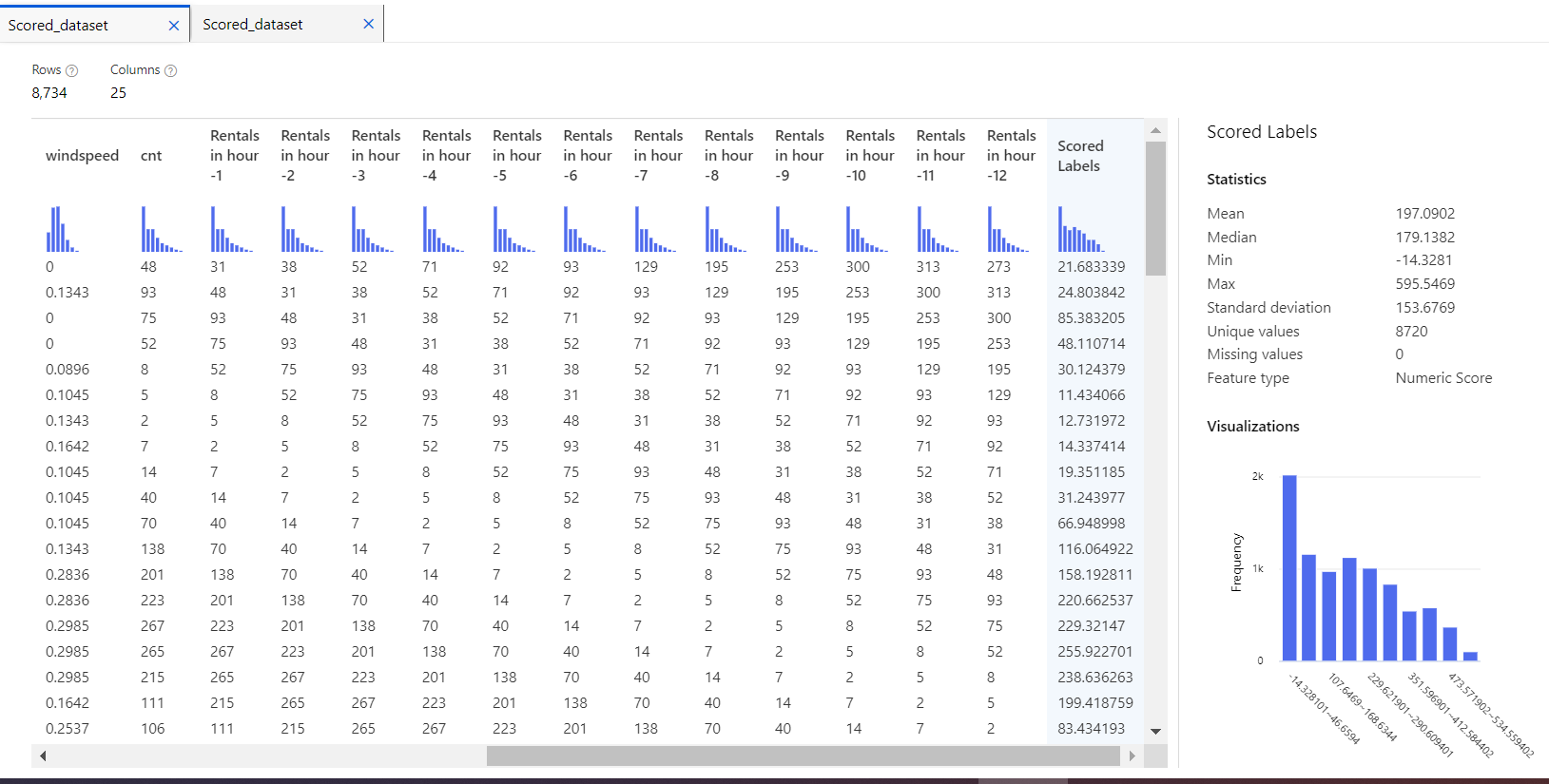
* Select and right click on the **Score Model** module and choose **Preview data** 🡪 **Scored dataset**.

**Scored dataset** result without python script:



**Scored dataset** result with python script:





### Evaluate models

Use the **Evaluate Model** to see how well the trained model performed on the test dataset.

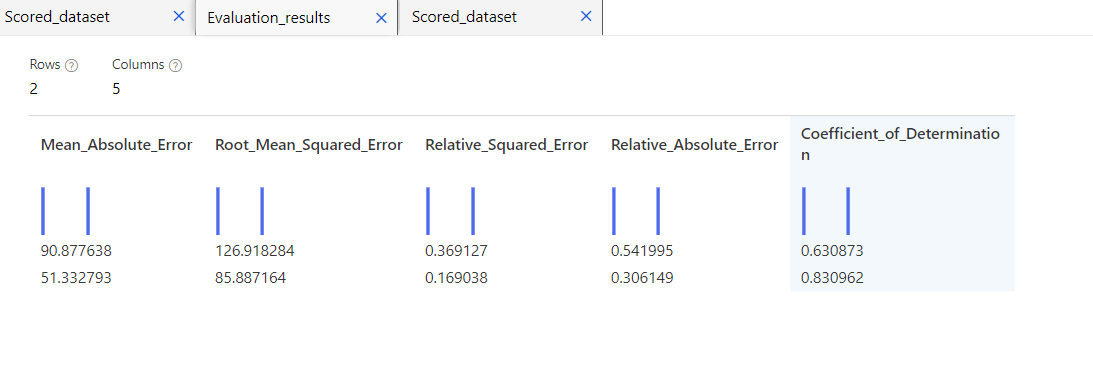
1. Right-click the **Evaluate Model** component and select **Preview data** > **Evaluation results** to view its output.

The following statistics are shown for your model:

* **Mean Absolute Error (MAE)**: The average of absolute errors. An error is the difference between the predicted value and the actual value.
* **Root Mean Squared Error (RMSE)**: The square root of the average of squared errors of predictions made on the test dataset.
* **Relative Absolute Error**: The average of absolute errors relative to the absolute difference between actual values and the average of all actual values.
* **Relative Squared Error**: The average of squared errors relative to the squared difference between the actual values and the average of all actual values.
* **Coefficient of Determination**: Also known as the R squared value, this statistical metric indicates how well a model fits the data.

For each of the error statistics, smaller is better. A smaller value indicates that the predictions are closer to the actual values. For the coefficient of determination, the closer its value is to one (1.0), the better the predictions.

The **Evaluation results** for the comparison of **two Score Model** is:



**Evaluation Results**

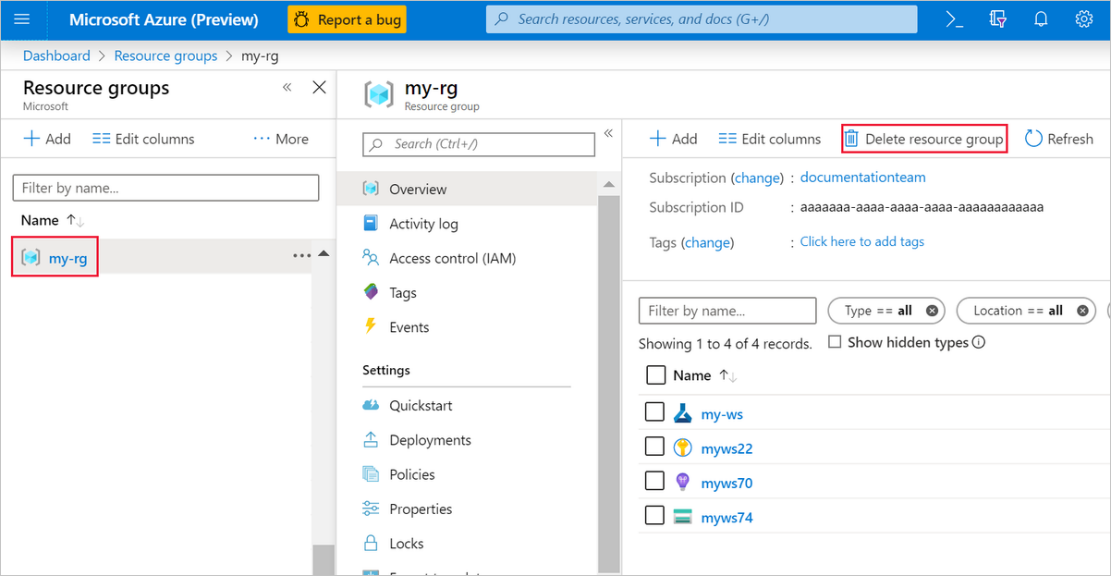
* Observe the values for the Mean\_Absolute\_Error, RMSE and R-Squared and other metrics. The first values correspond to the model trained on feature set A. The second values correspond to the model trained on feature sets A + B.
* Using simple feature engineering to derive new features from the existing data set allows the model to better understand the dynamics of the data and hence, produce a better prediction.

**Clean up resources**

### Delete everything

If you don't plan to use anything that you created, delete the entire resource group so you don't incur any charges.

* In the Azure portal, select **Resource groups** on the left side of the window.



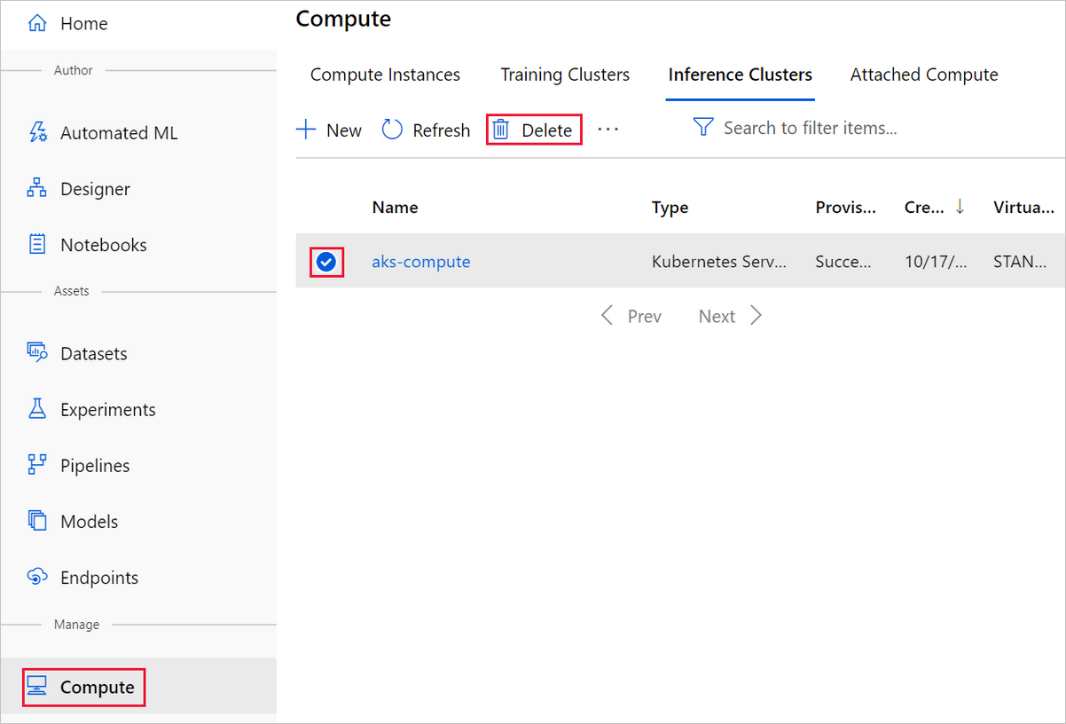
* In the list, select the resource group that you created.
* Select **Delete resource group**.

Deleting the resource group also deletes all resources that you created in the designer.

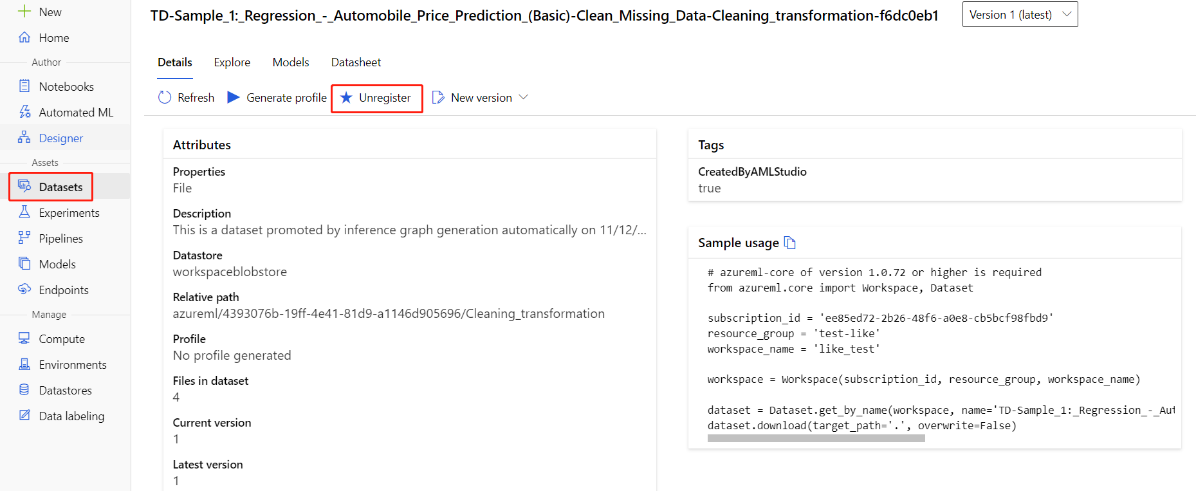
### Delete individual assets

In the designer where you created your experiment, delete individual assets by selecting them and then selecting the **Delete** button.

The compute target that you created here automatically autoscales to zero nodes when it's not being used. This action is taken to minimize charges. If you want to delete the compute target, take these steps:



You can unregister datasets from your workspace by selecting each dataset and selecting **Unregister**.



To delete a dataset, go to the storage account by using the Azure portal or Azure Storage Explorer and manually delete those assets.

\*\*\*Thank you! \*\*\*