Exercise Part 1: Create a Microsoft Azure Machine Learning Workspace

Microsoft Azure Machine Learning is a cloud-based platform for building and operating machine learning solutions in Azure. It includes a wide range of features and capabilities that help data scientists prepare data, train models, publish predictive services, and monitor their usage. One of these features is a visual interface called *designer*, that you can use to train, test, and deploy machine learning models without writing any code.

Create an Azure Machine Learning workspace

To use Azure Machine Learning, you create a *workspace* in your Microsoft Azure subscription. You can then use this workspace to manage data, compute resources, code, models, and other artifacts related to your machine learning workloads.

Note

This module is one of many that make use of an Azure Machine Learning workspace. If you are completing this module in preparation for the Azure AI Fundamentals or Azure Data Scientist certification, consider creating the workspace once and reusing it in other modules. After completing each module, be sure to follow the Clean Up instructions at the end of the module to stop compute resources.

If you do not already have one, follow these steps to create a workspace:

1. Sign into the Microsoft Azure portal using your Microsoft credentials.

2. Select＋**Create a resource**, search for Machine Learning, and create a new **Machine Learning** resource with the following settings:

- **Subscription:** Your Azure subscription
- **Resource group:** Create or select a resource group
- **Workspace name:** Enter a unique name for your workspace
- **Region:** Select the geographical region closest to you
- **Storage account:** Note the default new storage account that will be created for your workspace
- **Key vault:** Note the default new key vault that will be created for your workspace
- **Application insights:** Note the default new application insights resource that will be created for your workspace
- **Container registry:** None (one will be created automatically the first time you deploy a model to a container)

3. Wait for your workspace to be created (it can take a few minutes). Then go to it in the portal.

4. On the **Overview** page for your workspace, launch Microsoft Azure Machine Learning Studio (or open a new browser tab and navigate to https://ml.azure.com), and sign into Azure Machine Learning studio using your Microsoft account.

5. In Azure Machine Learning studio, toggle the ☰ icon at the top left to view the various pages in the interface. You can use these pages to manage the resources in your workspace.

You can manage your workspace using the Azure portal, but for data scientists and Machine Learning operations engineers, Azure Machine Learning studio provides a more focused user interface for managing workspace resources.

Exercise Part 2: Create Compute Resources

To train and deploy models using Microsoft Azure Machine Learning designer, you need compute power on which to run the training process, and to test the trained model after deploying it.

Create compute targets

Compute targets are cloud-based resources on which you can run model training and data exploration processes.

1. In Microsoft https://ml.azure.com/, view the **Compute** page (under **Manage**). This is where you manage the compute targets for your data science activities. There are four kinds of compute resources you can create:

- **Compute Instances**: Development workstations that data scientists can use to work with data and models.
- **Compute Clusters**: Scalable clusters of virtual machines for on-demand processing of experiment code.
- **Inference Clusters**: Deployment targets for predictive services that use your trained models.
- **Attached Compute**: Links to existing Microsoft Azure compute resources, such as Virtual Machines or Azure Databricks clusters.

2. On the **Compute Instances** tab, add a new compute instance with the following settings. You'll use this as a workstation from which to test your model:

- **Virtual Machine type**: CPU
- **Virtual Machine size**: Standard_DS11_v2 (Choose **Select from all options** to search for and select this machine size)
- **Compute name**: *enter a unique name*
- **Enable SSH access**: Unselected

3. While the compute instance is being created, switch to the **Compute Clusters** tab, and add a new compute cluster with the following settings. You'll use this to train a machine learning model:

- **Virtual Machine priority**: Dedicated
- **Maximum number of nodes**: 2
- **Idle seconds before scale down**: 120
- **Enable SSH access**: Unselected
- **Compute name**: *enter a unique name*
- **Minimum number of nodes**: 0
- **Virtual Machine type**: CPU

- **Virtual Machine size**: Standard_DS11_v2 (Choose **Select from all options** to search for and select this machine size)

Note

If you decide not to complete this module, be sure to stop your compute instance to avoid incurring unnecessary charges to your Microsoft Azure subscription.

The compute targets will take some time to be created. You can move onto the next unit while you wait.

Exercise Part 3: Explore Data

To train a regression model, you need a dataset that includes historical *features* (characteristics of the entity for which you want to make a prediction) and known *label* values (the numeric value that you want to train a model to predict).

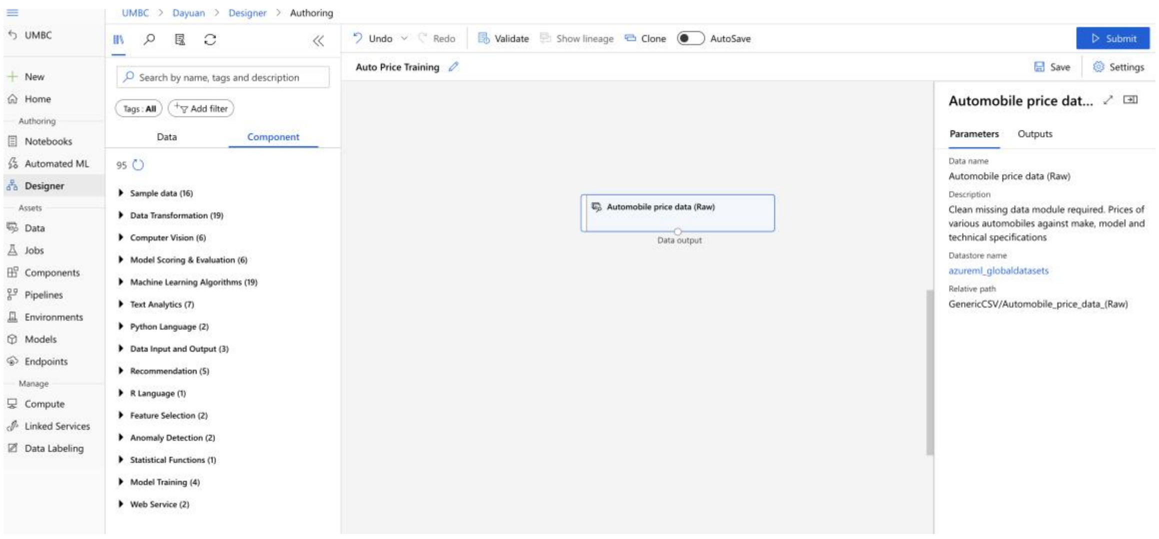
Create a pipeline

To use the Microsoft Azure Machine Learning designer, you create a *pipeline* that you will use to train a machine learning model. This pipeline starts with the dataset from which you want to train the model.

1. In Microsoft https://ml.azure.com, view the **Designer** page (under **Author**), and select **+** to create a new pipeline.

2. In the **Settings** pane, change the default pipeline name (**Pipeline-Created-on-*date***) to **Auto Price Training** (if the **Settings** pane is not visible, select the ⚙ icon next to the pipeline name at the top).

3. Observe that you need to specify a compute target on which to run the pipeline. In the **Settings** pane, use **Select compute target** to select the compute cluster you created previously.



Data name
Automobile price data (Raw)

Description
Clean missing data module required. Prices of various automobiles against make, model and technical specifications

Datastore name
azureml_globaldatasets

Relative path
GenericCSV/Automobile_price_data_(Raw)

Add and explore a dataset

In this module, you'll train a regression model that predicts the price of an automobile based on its characteristics. Azure Machine Learning includes a sample dataset that you can use for this model.

1. On the left side of the designer, expand the **Sample datasets** section, and drag the **Automobile price data (Raw)** dataset from the **Samples** section onto the canvas.

2. Right-click (Ctrl+click on a Mac) the **Automobile price data (Raw)** dataset on the canvas, and on the **Visualize** menu, select **Dataset output**.

3. Review the schema of the data, noting that you can see the distributions of the various columns as histograms.

4. Scroll to the right of the dataset until you see the **Price** column. This is the label your model will predict.

5. Select the column header for the **price** column and view the details that are displayed in the pane to the right. These include various statistics for the column values, and a histogram showing the distribution of the column values.

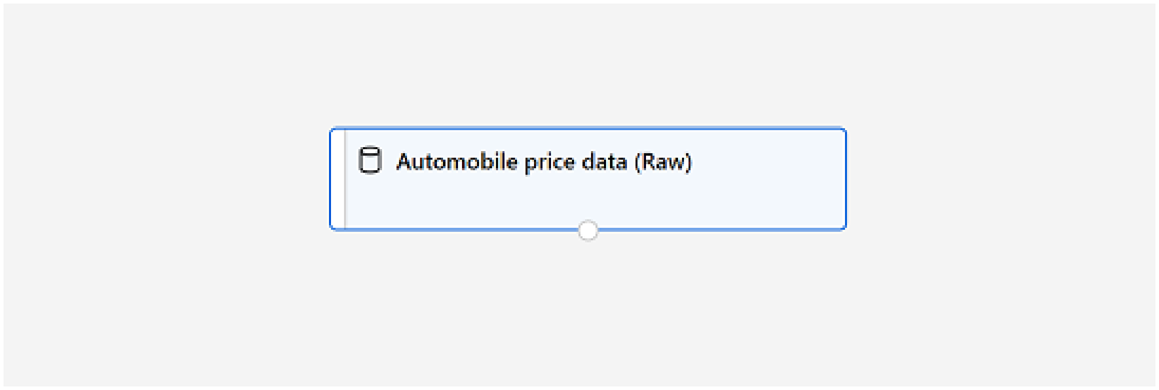
6. Scroll back to the left and select the **normalized-losses** column header. Then review the statistics for this column noting, there are quite a few missing values in this column. This will limit its usefulness in predicting the **price** label; so you might want to exclude it from training.

7. View the statistics for the **bore**, **stroke**, and **horsepower** columns, noting the number of missing values. These columns have significantly fewer missing values than **normalized-losses**, so they may still be useful in predicting **price** if you exclude the rows where the values are missing from training.

8. Compare the values in the **stroke**, **peak-rpm**, and **city-mpg** columns. These are all measured in different scales, and it's possible that the larger values for **peak-rpm** might bias the training

algorithm and create an over-dependency on this column compared to columns with lower values, such as **stroke**. Typically, data scientists mitigate this possible bias by *normalizing* the numeric columns so they're on the similar scales.

9. Close the **Automobile price data (Raw) result visualization** window so that you can see the dataset on the canvas like this:



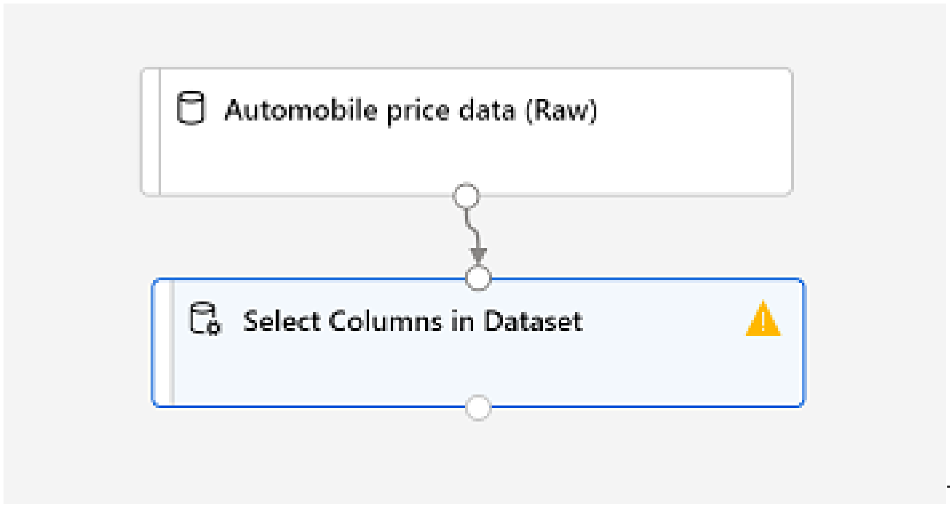
The Automobile price data (Raw) dataset on the designer canvas

Add data transformations

You typically apply data transformations to prepare the data for modeling. In the case of the automobile price data, you'll add transformations to address the issues you identified when exploring the data.

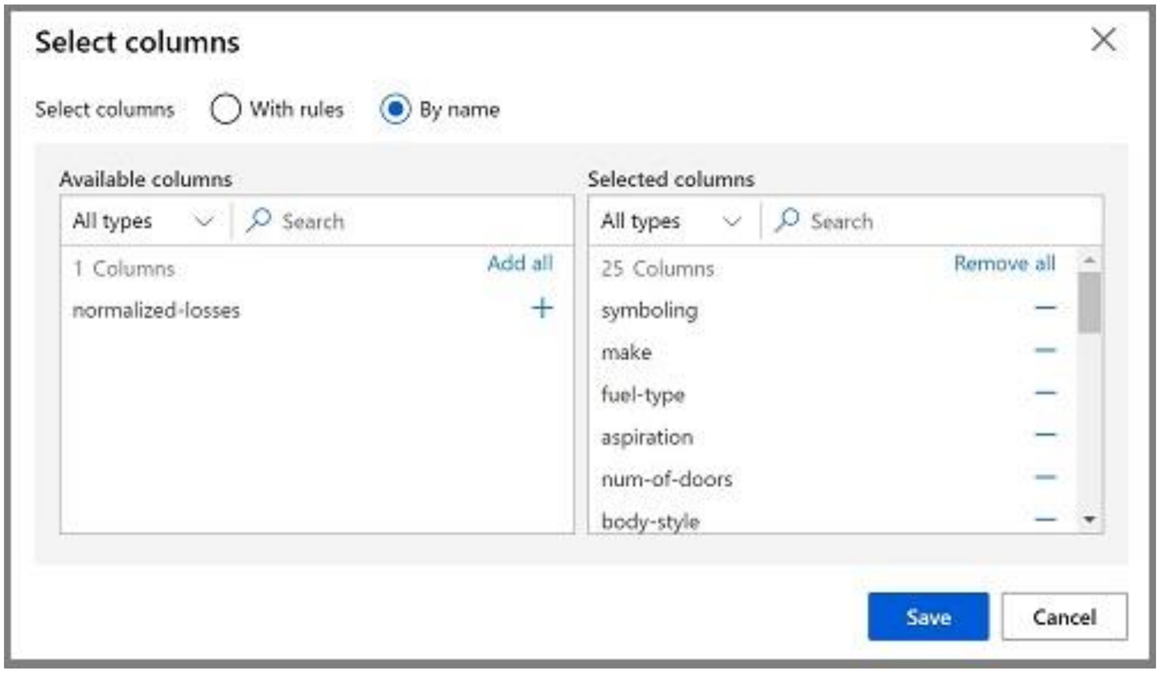
1. In the pane on the left, expand the **Data Transformation** section, which contains a wide range of modules you can use to transform data before model training.

2. Drag a **Select Columns in Dataset** module to the canvas, below the **Automobile price data (Raw)** module. Then connect the output at the bottom of the **Automobile price data (Raw)** module to the input at the top of the **Select Columns in Dataset** module, like this:



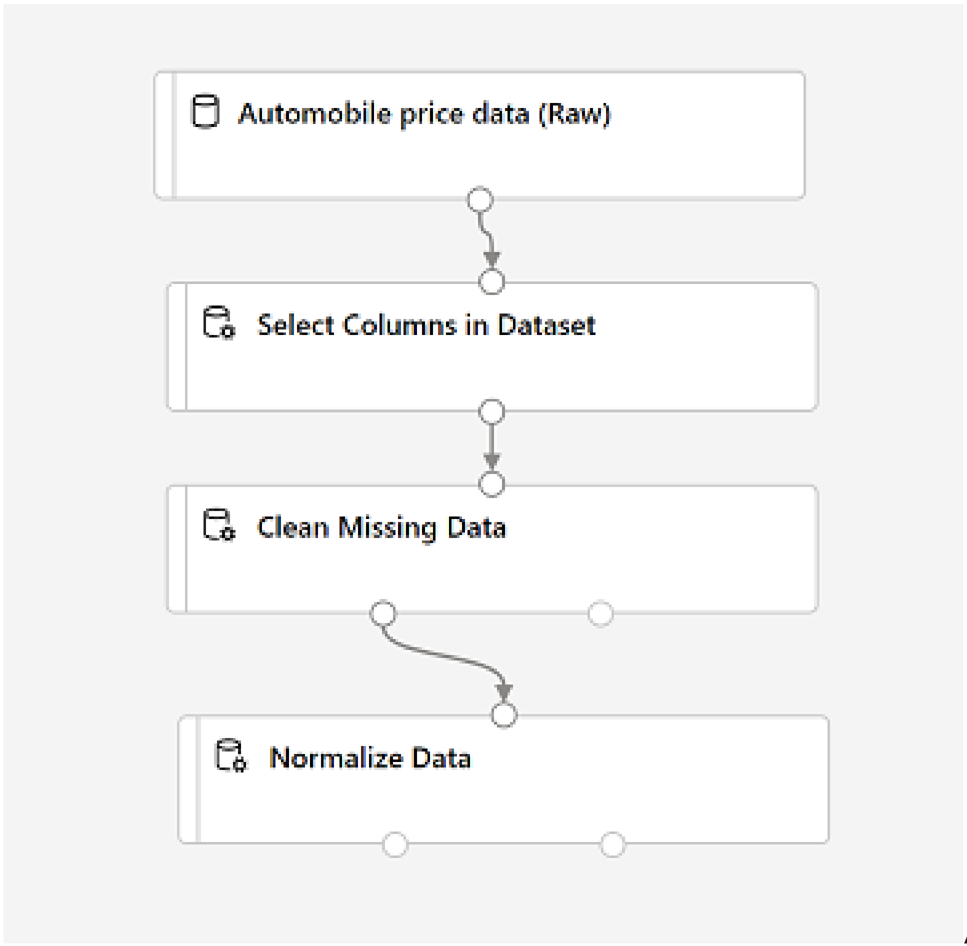
The Automobile price data (Raw) dataset connected to the Select Columns in Dataset module

3. Select the **Select Columns in Dataset** module, and in its **Settings** pane on the right, select **Edit column**. Then in the **Select columns** window, select **By name** and use the **+** links to add all columns other than **normalized-losses**, like this:



all columns other than normalized losses

In the rest of this exercise, you're going to create a pipeline that looks like this:

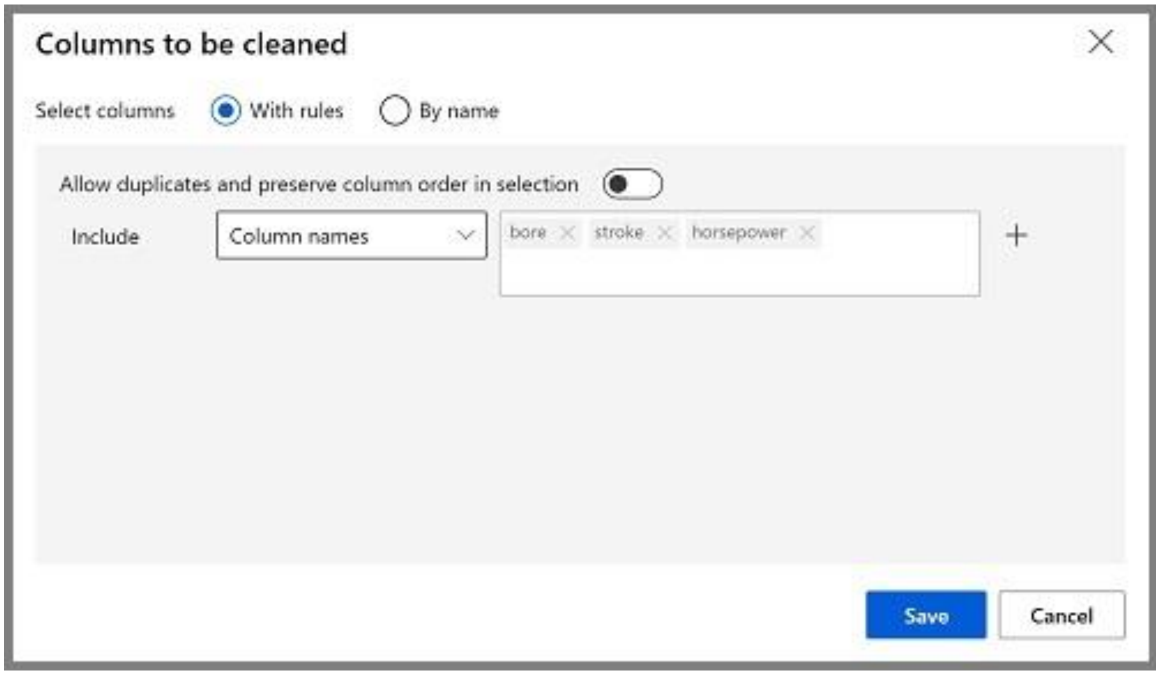


Automobile price data (Raw) dataset with Select Columns in Dataset, Clean Missing Data, and Normalize Data modules

Follow the remaining steps, using the image above for reference as you add and configure the required modules.

4. Drag a **Clean Missing Data** module from the **Data Transformations** section, and place it under the **Select Columns in Dataset** module. Then connect the output from the **Select Columns in Dataset** module to the input of the **Clean Missing Data** module.

5. Select the **Clean Missing Data** module, and in the settings pane on the right, click **Edit column**. Then in the **Select columns** window, select **With rules**, in the **Include** list select **Column names**, in the box of column names enter **bore**, **stroke**, and **horsepower** (making sure you match the spelling and capitalization exactly), like this:


Columns to be cleaned

Select columns [✓ With rules] By name

Allow duplicates and preserve column order in selection

Include: Column names — bore, stroke, horsepower

bore, stroke, and horsepower columns are selected

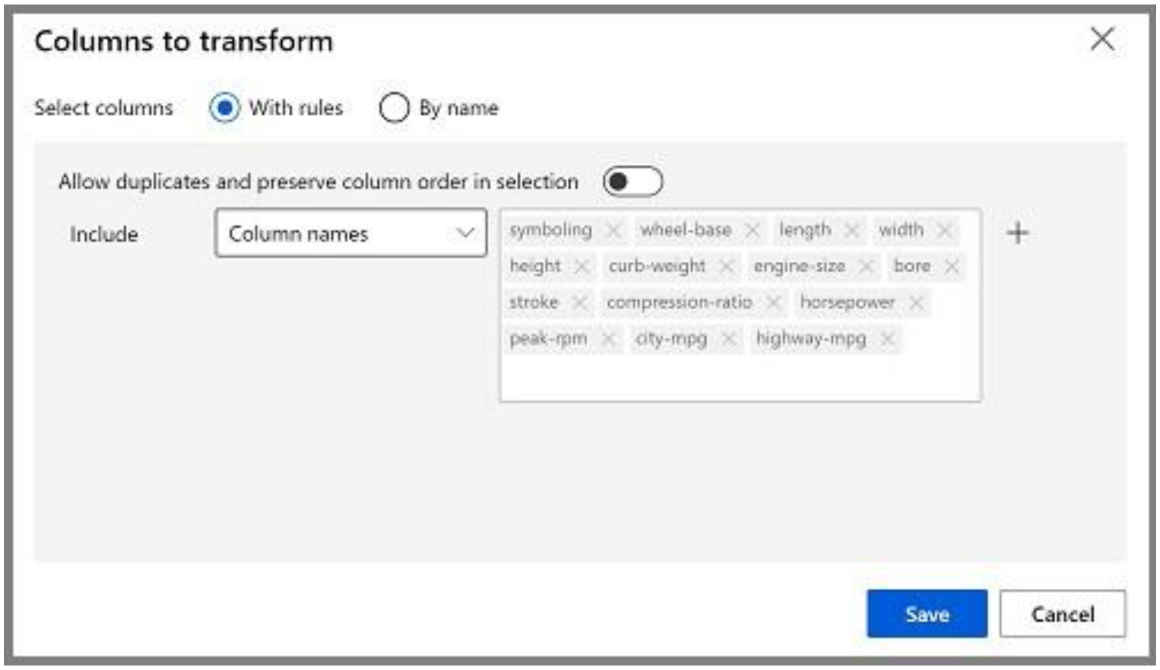
6. With the **Clean Missing Data** module still selected, in the settings pane, set the following configuration settings:

- **Minimum missing value ratio**: 0.0
- **Maximum missing value ratio**: 1.0
- **Cleaning mode**: Remove entire row

7. Drag a **Normalize Data** module to the canvas, below the **Clean Missing Data** module. Then connect the left-most output from the **Clean Missing Data** module to the input of the **Normalize Data** module.

8. Select the **Normalize Data** module and view its settings, noting that it requires you to specify the transformation method and the columns to be transformed. Then, set the transformation to **MinMax** and edit the columns by applying a rule to include the following **Column names** (ensuring you match the spelling, capitalization, and hyphenation exactly):

- **symboling**
- **wheel-base**
- **length**
- **width**
- **height**
- **curb-weight**
- **engine-size**
- **bore**
- **stroke**
- **compression-ratio**
- **horsepower**
- **peak-rpm**
- **city-mpg**
- **highway-mpg**


Columns to transform

Select columns: (● With rules) ( ) By name

Allow duplicates and preserve column order in selection

Include: Column names

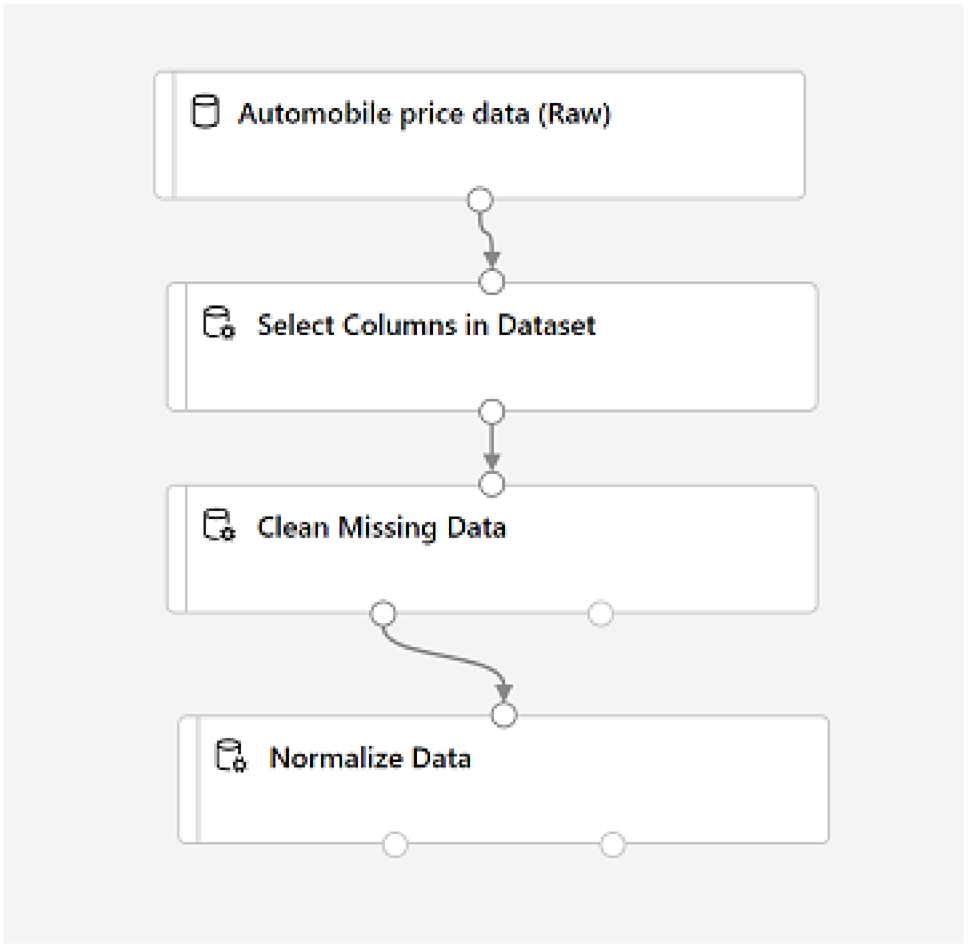
symboling, wheel-base, length, width, height, curb-weight, engine-size, bore, stroke, compression-ratio, horsepower, peak-rpm, city-mpg, highway-mpg

all numeric columns other than price are selected

Run the pipeline

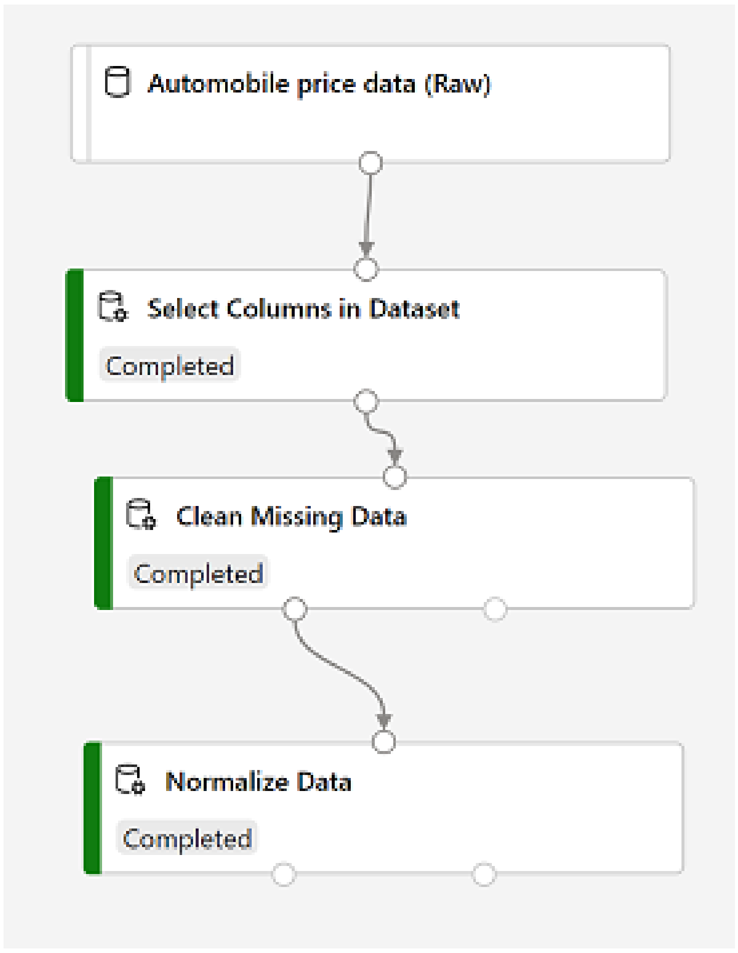
To apply your data transformations, you need to run the pipeline as an experiment.

1. Ensure your pipeline looks similar to this:



2. Select **Submit**, and run the pipeline as a new experiment named **mslearn-auto-training** on your compute cluster.

3. Wait for the run to finish. This may take 5 minutes or more. When the run has completed, the modules should look like this:



View the transformed data

The dataset is now prepared for model training.

1. Select the completed **Normalize Data** module, and in its **Settings** pane on the right, on the **Outputs + logs** tab, select the **Visualize** icon for the **Transformed dataset**.

2. View the data, noting that the **normalized-losses** column has been removed, all rows contain data for **bore**, **stroke**, and **horsepower**, and the numeric columns you selected have been normalized to a common scale.

3. Close the normalized data result visualization.

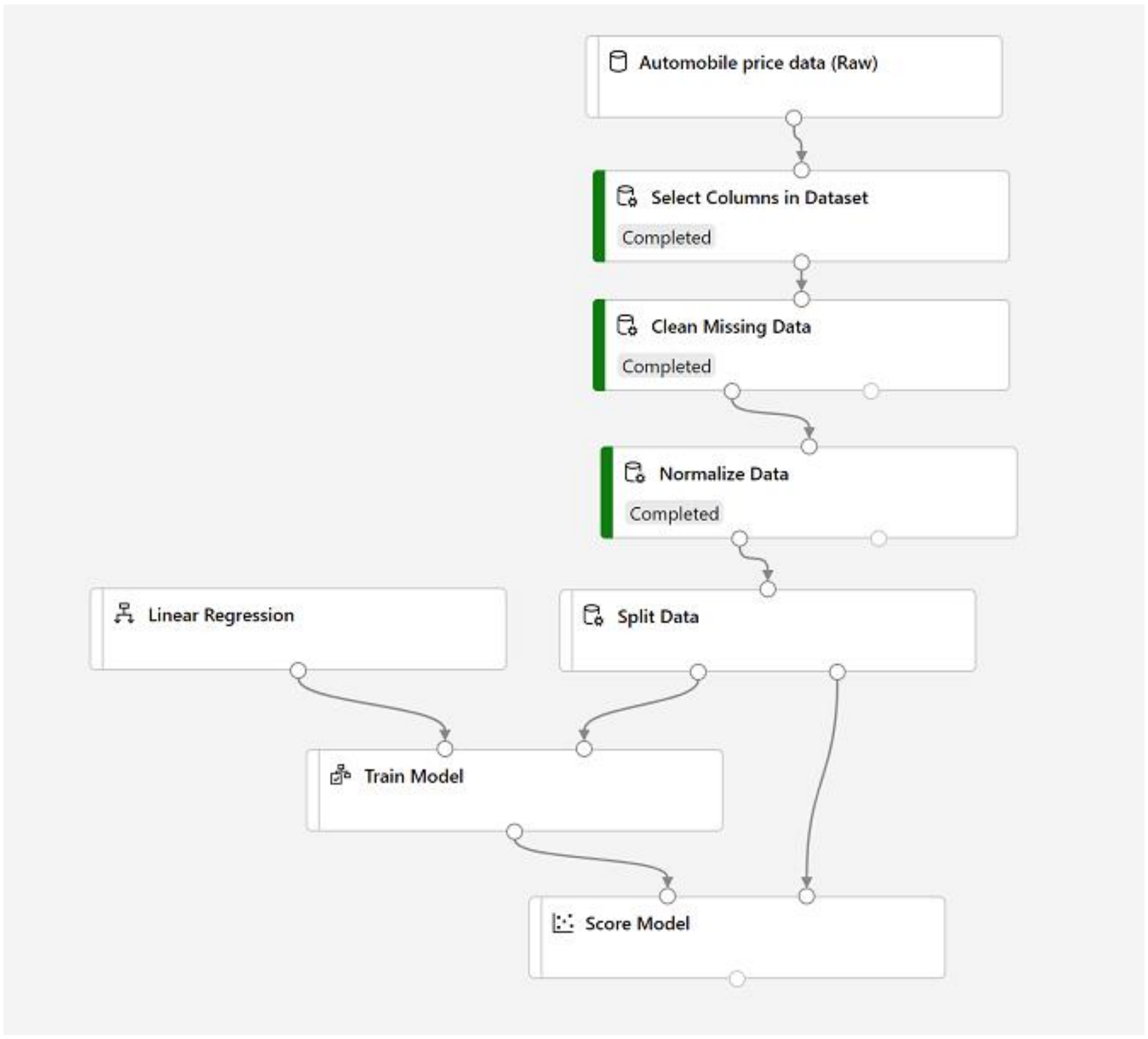
Exercise Part 4: Create and Run a Training Pipeline

After you've used data transformations to prepare the data, you can use it to train a machine learning model.

Add training modules

It's common practice to train the model using a subset of the data, while holding back some data with which to test the trained model. This enables you to compare the labels that the model predicts with the actual known labels in the original dataset.

In this exercise, you're going to extend the **Auto Price Training** pipeline as shown here:



split data, then train with linear regression and score

Follow the steps below, using the image above for reference as you add and configure the required modules.

1. Open the **Auto Price Training** pipeline you created in the previous unit if it's not already open.

2. In the pane on the left, in the **Data Transformations** section, drag a **Split Data** module onto the canvas under the **Normalize Data** module. Then connect the *Transformed Dataset* (left) output of the **Normalize Data** module to the input of the **Split Data** module.

3. Select the **Split Data** module, and configure its settings as follows:

- **Splitting mode**: Split Rows
- **Fraction of rows in the first output dataset**: 0.7
- **Random seed**: 123
- **Stratified split**: False

4. Expand the **Model Training** section in the pane on the left, and drag a **Train Model** module to the canvas, under the **Split Data** module. Then connect the *Result dataset1* (left) output of the **Split Data** module to the *Dataset* (right) input of the **Train Model** module.

5. The model we're training will predict the **price** value, so select the **Train Model** module and modify its settings to set the **Label column** to **price** (matching the case and spelling exactly!)

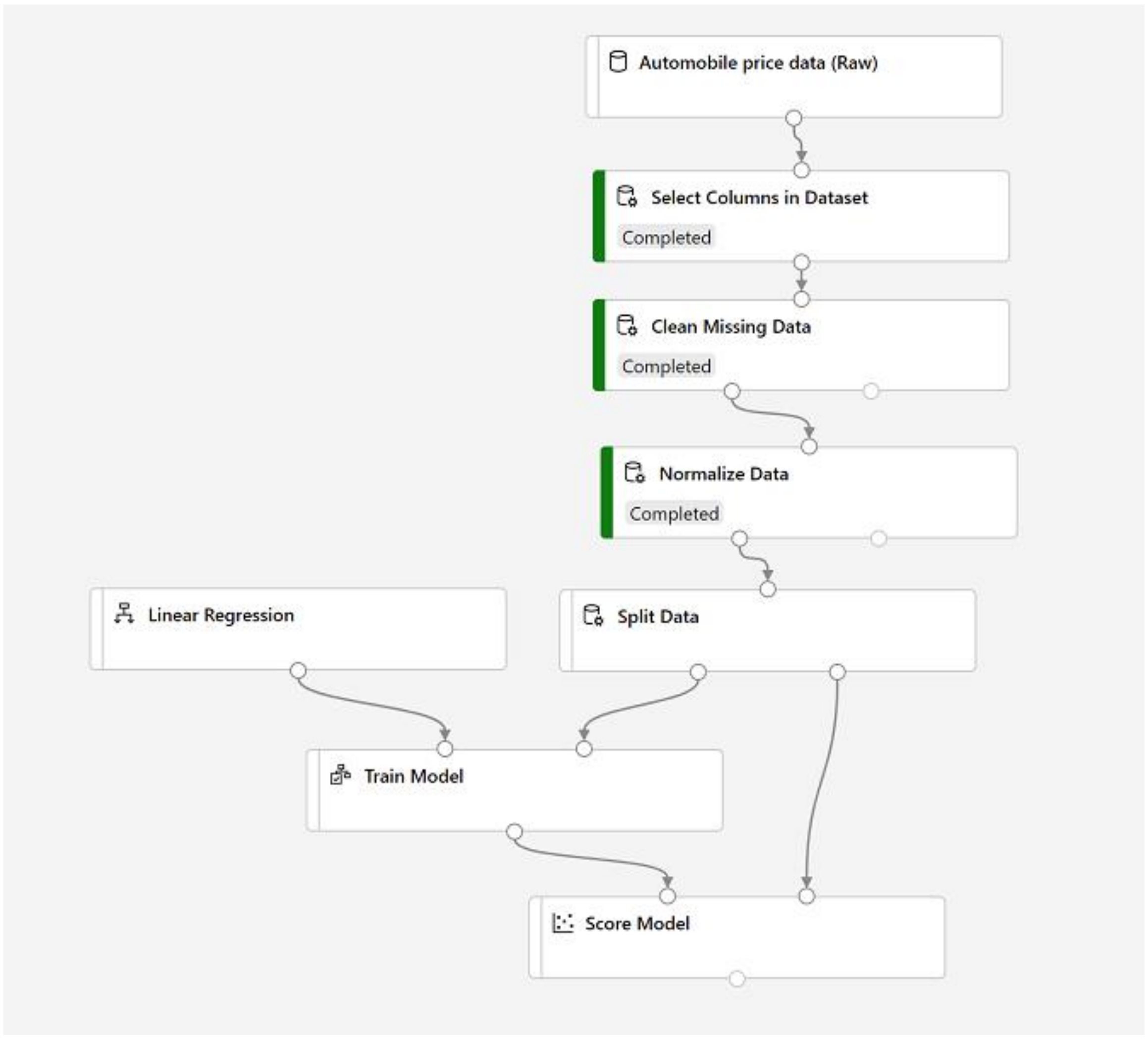
6. The **price** label the model will predict is a numeric value, so we need to train the model using a *regression* algorithm. Expand the **Machine Learning Algorithms** section, and under **Regression**, drag a **Linear Regression** module to the canvas, to the left of the **Split Data** module and above the **Train Model** module. Then connect its output to the **Untrained model** (left) input of the **Train Model** module.

Note

There are multiple algorithms you can use to train a regression model. For help choosing one, take a look at the https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-cheat-sheet for Microsoft Azure Machine Learning designer.

7. To test the trained model, we need to use it to *score* the validation dataset we held back when we split the original data - in other words, predict labels for the features in the validation dataset. Expand the **Model Scoring & Evaluation** section and drag a **Score Model** module to the canvas, below the **Train Model** module. Then connect the output of the **Train Model** module to the **Trained model** (left) input of the **Score Model** module; and drag the **Results dataset2** (right) output of the **Split Data** module to the **Dataset** (right) input of the **Score Model** module.

8. Ensure your pipeline looks like this:



split data, then train with linear regression and score

Run the training pipeline

Now you're ready to run the training pipeline and train the model.

1. Select **Submit**, and run the pipeline using the existing experiment named **mslearn-auto-training**.

2. Wait for the experiment run to complete. This may take 5 minutes or more.

3. When the experiment run has completed, select the **Score Model** module and in the settings pane, on the **Outputs + logs** tab, under **Data outputs** in the **Scored dataset** section, use the **Visualize** icon to view the results.

4. Scroll to the right, and note that next to the **price** column (which contains the known true values of the label) there is a new column named **Scored labels**, which contains the predicted label values.

5. Close the **Score Model result visualization** window.

The model is predicting values for the **price** label, but how reliable are its predictions? To assess that, you need to evaluate the model.

Exercise Part 5: Evaluate a Regression Model

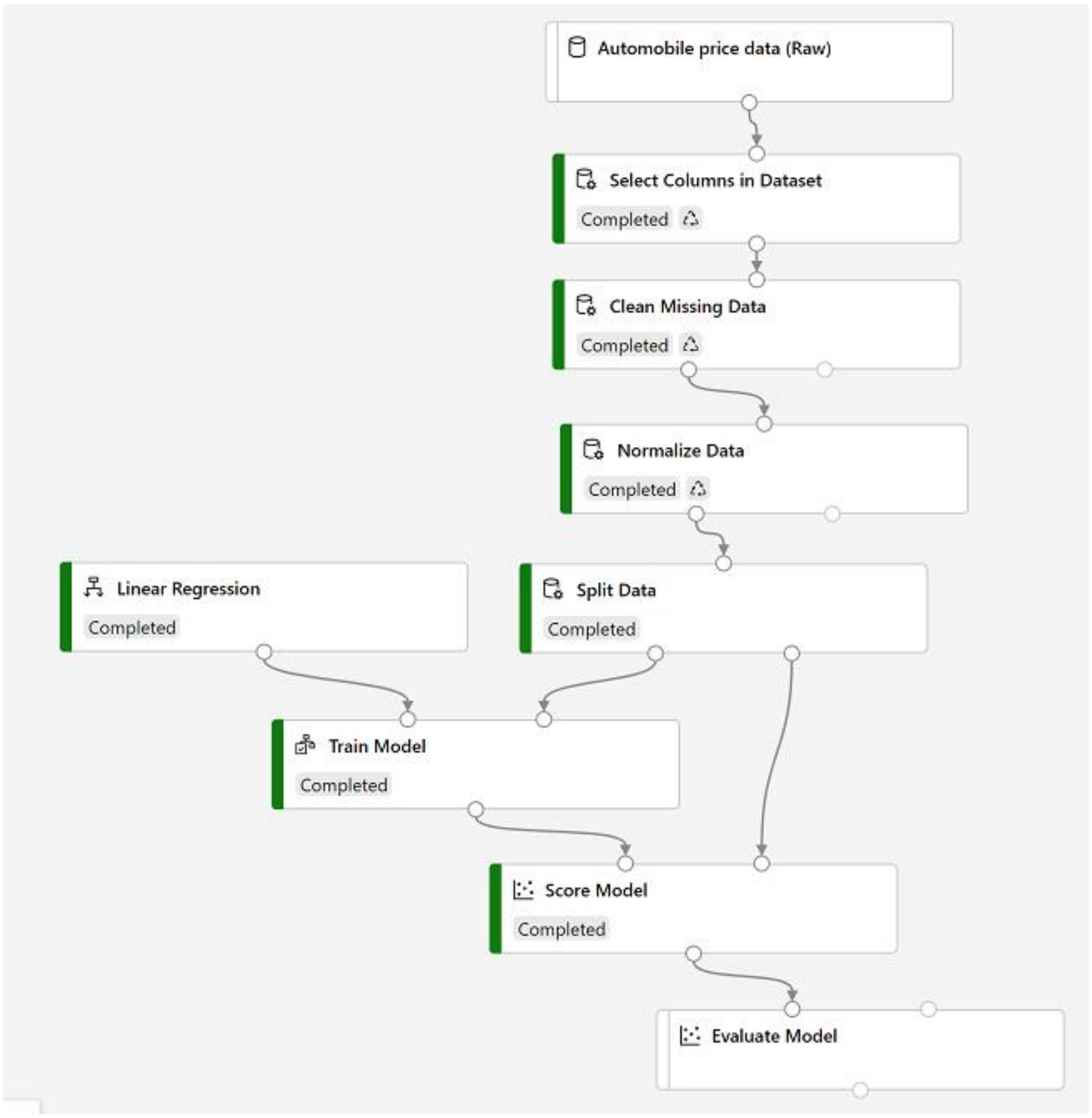
To evaluate a regression model, you could simply compare the predicted labels to the actual labels in the validation dataset to held back during training, but this is an imprecise process and doesn't provide a simple metric that you can use to compare the performance of multiple models.

Add an Evaluate Model module

1. Open the **Auto Price Training** pipeline you created in the previous unit if it's not already open.

2. In the pane on the left, in the **Model Scoring & Evaluation** section, drag an **Evaluate Model** module to the canvas, under the **Score Model** module, and connect the output of the **Score Model** module to the **Scored dataset** (left) input of the **Evaluate Model** module.

3. Ensure your pipeline looks like this:



split data, then train with linear regression and score

4. Select **Submit**, and run the pipeline using the existing experiment named **mslearn-auto-training**.

5. Wait for the experiment run to complete.

6. When the experiment run has completed, select the **Evaluate Model** module and in the settings pane, on the **Outputs + logs** tab, under **Data outputs** in the **Evaluation results** section, use the **Visualize** icon to view the results. These include the following regression performance metrics:

- **Mean Absolute Error (MAE)**: The average difference between predicted values and true values. This value is based on the same units as the label, in this case dollars. The lower this value is, the better the model is predicting.

- **Root Mean Squared Error (RMSE)**: The square root of the mean squared difference between predicted and true values. The result is a metric based on the same unit as the label (dollars). When compared to the MAE (above), a larger difference indicates greater variance in the individual errors (for example, with some errors being very small, while others are large).
- **Relative Squared Error (RSE)**: A relative metric between 0 and 1 based on the square of the differences between predicted and true values. The closer to 0 this metric is, the better the model is performing. Because this metric is relative, it can be used to compare models where the labels are in different units.
- **Relative Absolute Error (RAE)**: A relative metric between 0 and 1 based on the absolute differences between predicted and true values. The closer to 0 this metric is, the better the model is performing. Like RSE, this metric can be used to compare models where the labels are in different units.
- **Coefficient of Determination (R2)**: This metric is more commonly referred to as *R-Squared*, and summarizes how much of the variance between predicted and true values is explained by the model. The closer to 1 this value is, the better the model is performing

7. Close the **Evaluate Model result visualization** window.

You can try a different regression algorithm and compare the results by connecting the same outputs from the **Split Data** module to a second **Train model** module (with a different algorithm) and a second **Score Model** module; and then connecting the outputs of both **Score Model** modules to the same **Evaluate Model** module for a side-by-side comparison.

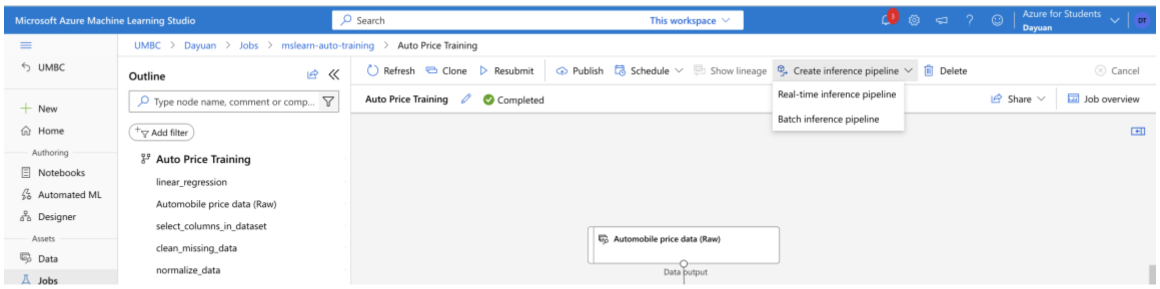
When you've identified a model with evaluation metrics that meet your needs, you can prepare to use that model with new data.

Exercise Part 6: Create an Inference Pipeline

After creating and running a pipeline to train the model, you need a second pipeline that performs the same data transformations for new data, and then uses the trained model to *inference* (in other words, predict) label values based on its features. This will form the basis for a predictive service that you can publish for applications to use.

Create and run an inference pipeline

1. In Microsoft Azure Machine Learning Studio, click the **Designer** page to view all of the pipelines you have created. Then open the **Auto Price Training** pipeline you created previously.

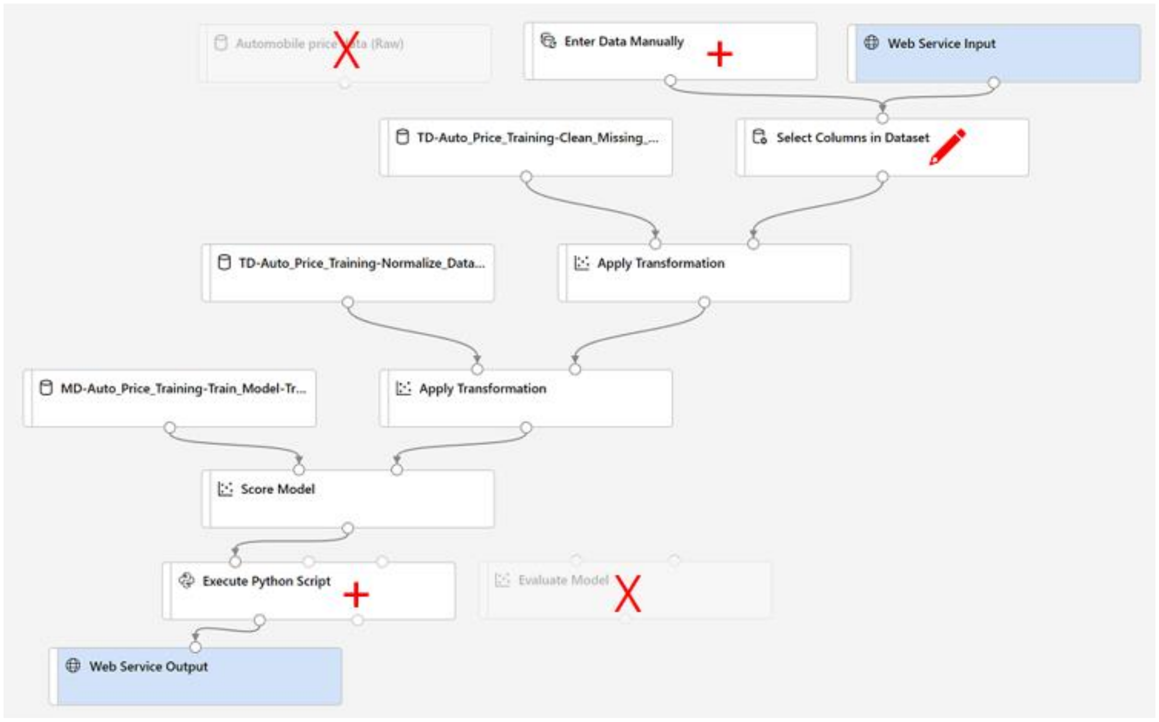


2. (Dayuan: New: it's in ***Jobs***) In the **Create inference pipeline** drop-down list, click **Real-time inference pipeline**. After a few seconds, a new version of your pipeline named **Auto Price Training-real time inference** will be opened.

If the pipeline does not include ***Web Service Input*** *and* ***Web Service Output*** *modules, go back to the* ***Designer*** *page and then re-open the* ***Auto Price Training-real time inference*** *pipeline.*

3. Rename the new pipeline to **Predict Auto Price**, and then review the new pipeline. It contains a web service input for new data to be submitted, and a web service output to return results. Some of the transformations and training steps have been encapsulated in this pipeline so that the statistics from your training data will be used to normalize any new data values, and the trained model will be used to score the new data.

You are going to make the following changes to the inference pipeline:


Automobile price data (Raw)
Enter Data Manually
Web Service Input
TD-Auto_Price_Training-Clean_Missing_...
Select Columns in Dataset
TD-Auto_Price_Training-Normalize_Data...
Apply Transformation
MD-Auto_Price_Training-Train_Model-Tr...
Apply Transformation
Score Model
Execute Python Script
Evaluate Model
Web Service Output

An inference pipeline with changes indicated

- Replace the **Automobile price data (Raw)** dataset with an **Enter Data Manually** module that does not include the label column (**price**)
- Modify the **Select Columns in Dataset** module to remove any reference to the (now absent) **price** column.
- Remove the **Evaluate Model** module.
- Insert an **Execute Python Script** module before the web service output to return only the predicted label.

Follow the remaining steps below, using the image and information above for reference as you modify the pipeline.

4. The inference pipeline assumes that new data will match the schema of the original training data, so the Automobile price data (Raw) dataset from the training pipeline is included. However, this input data includes the price label that the model predicts, which is unintuitive to include in

new car data for which a price prediction has not yet been made. Delete this module and replace it with an Enter Data Manually module from the Data Input and Output section, containing the following CSV data, which includes feature values without labels for three cars.

Important Note

You will need to copy and paste the entire block of text presented in the code block. Make sure you have selected all of the text or data in the code block including endpoints, brackets etc. before copying th and placing it into the specified location or site in the exercise. This will help to avoid errors occurring or l go back and start the exercise again. **Example** /public function processAPI() { if (method_exists($this, $ >endpoint)) { return $this->_response($this->{$this->endpoint}($this->args)); } return $this->_respo Endpoint: $this->endpoint", 404);/ You can also use the following shortcuts to copy and paste the code: inside the code box and select CTL + A followed by CTL + C Alternatively, if you are using a Mac select Com A and Command + C to copy all the code to your clipboard.

CSV Data

```
symboling,normalized-losses,make,fuel-type,aspiration,num-of-doors,body-style,drive-wheels,engine-location,wheel-base,length,width,height,curb-weight,engine-type,num-of-cylinders,engine-size,fuel-system,bore,stroke,compression-ratio,horsepower,peak-rpm,city-mpg,highway-mpg
3,NaN,alfa-romero,gas,std,two,convertible,rwd,front,88.6,168.8,64.1,48.8,2548,dohc,four,130,mpfi,3.47,2.68,9,111,5000,21,27
3,NaN,alfa-romero,gas,std,two,convertible,rwd,front,88.6,168.8,64.1,48.8,2548,dohc,four,130,mpfi,3.47,2.68,9,111,5000,21,27
1,NaN,alfa-romero,gas,std,two,hatchback,rwd,front,94.5,171.2,65.5,52.4,2823,ohcv,six,152,mpfi,2.68,3.47,9,154,5000,19,26
```

5. Connect the new **Enter Data Manually** module to the same **dataset** input of the **Select Columns in Dataset** module as the **Web Service Input**.

6. Now that you've changed the schema of the incoming data to exclude the **price** field, you need to remove any explicit uses of this field in the remaining modules. Select the **Select Columns in Dataset** module and then in the settings pane, edit the columns to remove the **price** field.

7. The inference pipeline includes the **Evaluate Model** module, which is not useful when predicting from new data, so delete this module.

8. The output from the **Score Model** module includes all of the input features as well as the predicted label. To modify the output to include only the prediction:

- Delete the connection between the **Score Model** module and the **Web Service Output**.
- Add an **Execute Python Script** module from the **Python Language** section, replacing all of the the default python script with the following code (which selects only the **Scored Labels** column and renames it to **predicted_price**):

Important Note

You will need to copy and paste the entire block of text presented in the code block. Make sure you have selected all of the text or data in the code block including endpoints, brackets etc. before copying th and placing it into the specified location or site in the exercise. This will help to avoid errors occurring or go back and start the exercise again. **Example** /public function processAPI() { if (method_exists($this, $ >endpoint)) { return $this->_response($this->{$this->endpoint}($this->args)); } return $this->_respo Endpoint: $this->endpoint", 404);/ You can also use the following shortcuts to copy and paste the code: inside the code box and select CTL + A followed by CTL + C Alternatively, if you are using a Mac select Com A and Command + C to copy all the code to your clipboard.

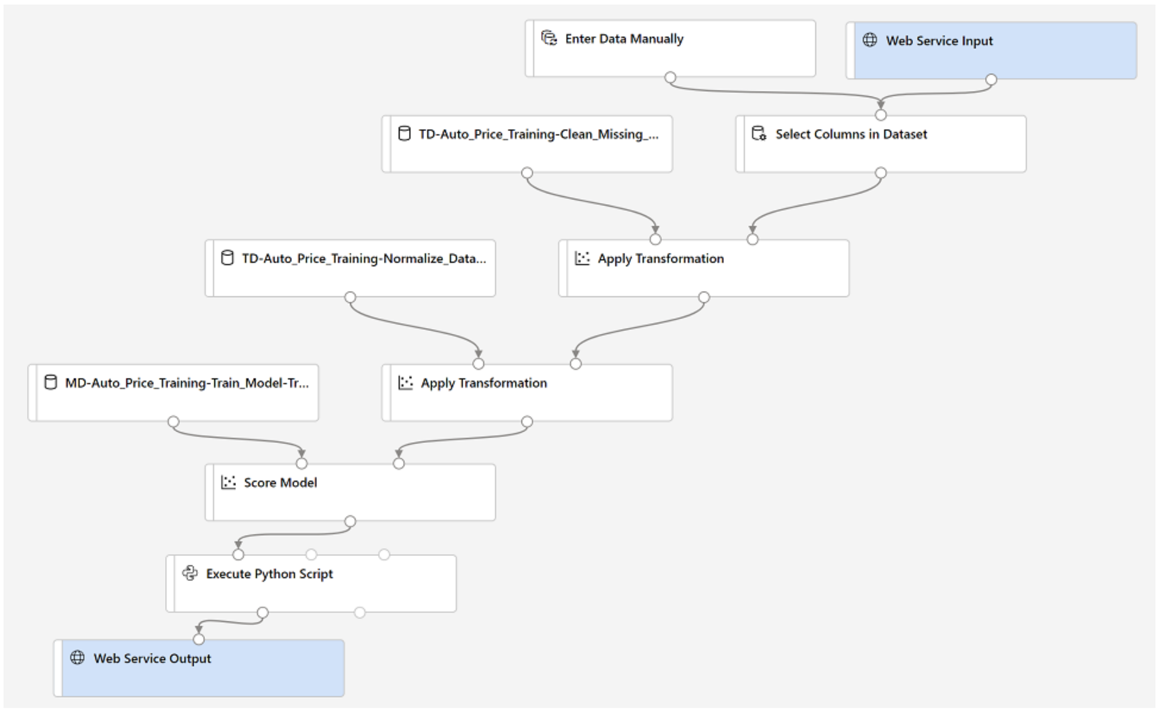
```
import pandas as pd

def azureml_main(dataframe1 = None, dataframe2 = None):

    scored_results = dataframe1[['Scored Labels']]
    scored_results.rename(columns={'Scored Labels':'predicted_price'},
                          inplace=True)
    return scored_results
```

- Connect the output from the **Score Model** module to the **Dataset1** (left-most) input of the **Execute Python Script**, and connect the output of the **Execute Python Script** module to the **Web Service Output**.

9. Verify that your pipeline looks similar to the following:


Enter Data Manually
Web Service Input
TD-Auto_Price_Training-Clean_Missing_...
Select Columns in Dataset
TD-Auto_Price_Training-Normalize_Data...
Apply Transformation
MD-Auto_Price_Training-Train_Model-Tr...
Apply Transformation
Score Model
Execute Python Script
Web Service Output

10. Submit the pipeline as a new experiment named **mslearn-auto-inference** on your compute cluster. This may take a while!

11. When the pipeline has completed, select the **Execute Python Script** module, and in the settings pane, on the **Output + logs** tab, visualize the **Result dataset** to see the predicted prices for the three cars in the input data.

12. Close the visualization window.

Your inference pipeline predicts prices for cars based on their features. Now you're ready to publish the pipeline so that client applications can use it.

Exercise Part 7: Deploy a Predictive Service

After you've created and tested an inference pipeline for real-time inferencing, you can publish it as a service for client applications to use.

Note

In this exercise, you'll deploy the web service to a Microsoft Azure Container Instance (ACI). This type of compute is created dynamically, and is useful for development and testing. For production, you should create an *inference cluster*, which provide an Azure Kubernetes Service (AKS) cluster that provides better scalability and security.

Deploy a service

1. View the **Predict Auto Price** inference pipeline you created in the previous unit.

2. At the top right, select **Deploy**, and deploy a new real-time endpoint, using the following settings:

- **Name**: predict-auto-price
- **Description**: Auto price regression.
- **Compute type**: Azure Container Instance

3. Wait for the web service to be deployed - this can take several minutes. The deployment status is shown at the top left of the designer interface.

Test the service

Now you can test your deployed service from a client application - in this case, you'll use the code in the cell below to simulate a client application.

1. On the **Endpoints** page, open the **predict-auto-price** real-time endpoint.

2. When the **predict-auto-price** endpoint opens, view the **Consume** tab and note the following information there. You need this to connect to your deployed service from a client application.

- The REST endpoint for your service
- The Primary Key for your service

3. Observe that you can use the ⧉ link next to these values to copy them to the clipboard.

4. With the **Consume** page for the **predict-auto-price** service page open in your browser, open a new browser tab and open a second instance of Azure Machine Learning studio. Then in the new tab, view the **Notebooks** page (under **Author**).

5. In the **Notebooks** page, under **My files**, use the 🗋 button to create a new file with the following settings:

- **File location**: Users/*your user name*
- **File name**: Test-Autos
- **File type**: Notebook
- **Overwrite if already exists**: Selected

6. When the new notebook has been created, ensure that the compute instance you created previously is selected in the **Compute** box, and that it has a status of **Running**.

7. Use the ≪ button to collapse the file explorer pane and give you more room to focus on the **Test-Autos.ipynb** notebook tab.

Important note

You will need to copy and paste the entire block of text presented in the code block. Make sure you have selected all of the text or data in the code block including endpoints, brackets etc. before copying this ov placing it into the specified position, such as a note pad or program, in the exercise. This will help to avoi occurring or having to go back and start the exercise again. **Example** /public function processAPI() { if (method_exists($this, $this->endpoint)) { return $this->_response($this->{$this->endpoint}($this-

>args)); } return $this->_response("No Endpoint: $this->endpoint", 404);/ You can also use the followi
shortcuts to copy and paste the code: 1. Click inside the code box and select CTL + A followed by CTL+ C
Alternatively, if you are using a Mac select Command + A and Command + C to copy all the code to your
clipboard8.

8. In the rectangular cell that has been created in the notebook, paste the following code:

```
endpoint = 'YOUR_ENDPOINT' #Replace with your endpoint
key = 'YOUR_KEY' #Replace with your key

import urllib.request
import json
import os

# Prepare the input data
data = {
    "Inputs": {
        "WebServiceInput0":
        [
            {
                    'symboling': 3,
                    'normalized-losses': None,
                    'make': "alfa-romero",
                    'fuel-type': "gas",
                    'aspiration': "std",
                    'num-of-doors': "two",
                    'body-style': "convertible",
                    'drive-wheels': "rwd",
                    'engine-location': "front",
                    'wheel-base': 88.6,
                    'length': 168.8,
                    'width': 64.1,
                    'height': 48.8,
                    'curb-weight': 2548,
                    'engine-type': "dohc",
                    'num-of-cylinders': "four",
                    'engine-size': 130,
                    'fuel-system': "mpfi",
                    'bore': 3.47,
                    'stroke': 2.68,
                    'compression-ratio': 9,
                    'horsepower': 111,
                    'peak-rpm': 5000,
                    'city-mpg': 21,
                    'highway-mpg': 27,
            },
        ],
    },
    "GlobalParameters":  {
    }
}
body = str.encode(json.dumps(data))
headers = {'Content-Type':'application/json', 'Authorization':('Bearer '+ key)}
req = urllib.request.Request(endpoint, body, headers)

try:
    response = urllib.request.urlopen(req)
    result = response.read()
    json_result = json.loads(result)
    y = json_result["Results"]["WebServiceOutput0"][0]["predicted_price"]
    print('Predicted price: {:.2f}'.format(y))

except urllib.error.HTTPError as error:
    print("The request failed with status code: " + str(error.code))

    # Print the headers to help debug the error
    print(error.info())
    print(json.loads(error.read().decode("utf8", 'ignore')))
```

Note

Don't worry too much about the details of the code. It just submits details of a car and uses the **predict-auto-price** service you created to get a predicted price. 9. Switch to the browser tab containing the **Consume** page for the **predict-auto-price** service, and copy the REST endpoint for your service. The switch back to the tab containing the notebook and paste the key into the code, replacing YOUR_ENDPOINT.

10. Switch to the browser tab containing the **Consume** page for the **predict-auto-price** service, and copy the Primary Key for your service. The switch back to the tab containing the notebook and paste the key into the code, replacing YOUR_KEY.

11. Save the notebook. Then use the ▷ button next to the cell to run the code.

12. Verify that predicted price is returned.

Exercise Part 8: Clean-up

Azure Container Instances (ACI) is a service that enables you to deploy containers on Microsoft Azure. The web service you created is hosted in *Azure Container Instance*. If you don't intend to experiment with it further, you should delete the endpoint to avoid accruing unnecessary Azure usage. You should also stop the compute instance until you need it again.

1. In Microsoft https://ml.azure.com, on the **Endpoints** tab, select the **predict-auto-price** endpoint. Then select **Delete** (🗑) and confirm that you want to delete the endpoint.

2. On the **Compute** page, on the **Compute Instances** tab, select your compute instance and then select **Stop**.

If you have finished exploring Azure Machine Learning, you can delete the resource group containing your Azure Machine Learning workspace from your Azure subscription:

1. In the https://portal.azure.com, in the **Resource groups** page, open the resource group you specified when creating your Azure Machine Learning workspace.

2. Click **Delete resource group**, type the resource group name to confirm you want to delete it, and select **Delete**.