Machine Learning

Microsoft

Azure Machine Learning Studio Lab

- Experimentation

Objectives

1. Problem Domain

- Understand the problem involving predicting the price of a car give a list of other attributes about the car

2. Experimentation

- Setup the experiment using Azure ML
- Perform data pre-processing
- Train and evaluate the model

The Problem Domain

In this lab, we are given historical data collected about cars. The goal of your task is to predict the price/value of a car given other attributes associated with the car:

- The attribute columns in the dataset include:
    - model/make
    - fuel type
    - body style
    - performance values such as MPG
    - horsepower
    - engine type
- The goal is to predict the price of the car. In this dataset, the values range from £5,000 to £45,000.

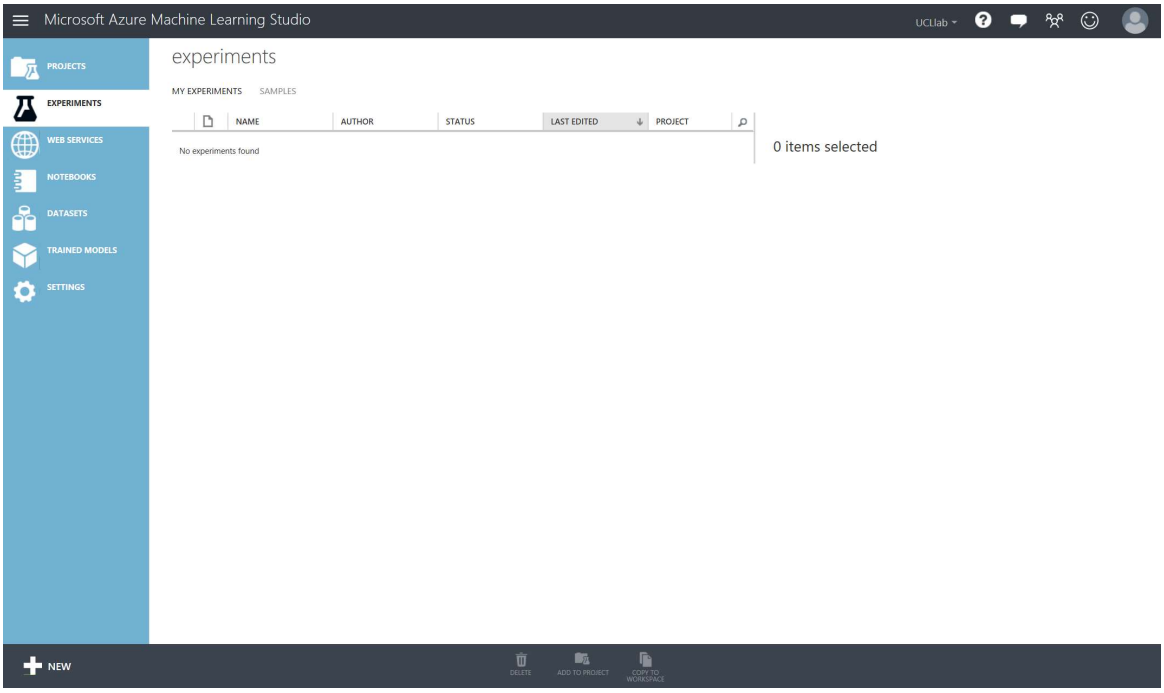
Approach

Store the dataset in Azure Blob and start to pre-process the dataset in order to train a machine learning model.

We will use a regression algorithm to predict the actual price of the car using historical car attribute values with the lowest amount of error.

Enter Workspace

Set up an account at http://studio.azureml.net if you have not. **'Sign In'** to enter the Azure Machine Learning Studio as shown below:

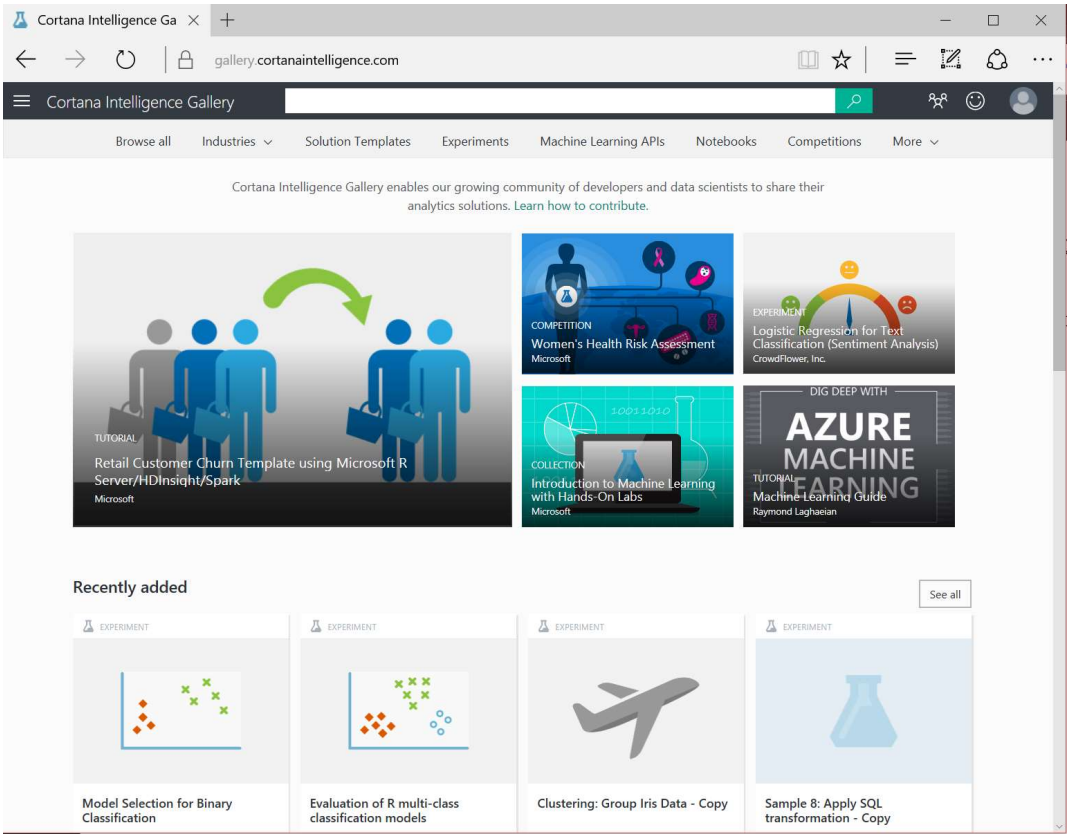


Getting Started

In the studio, we can either upload the file or connect to its source:

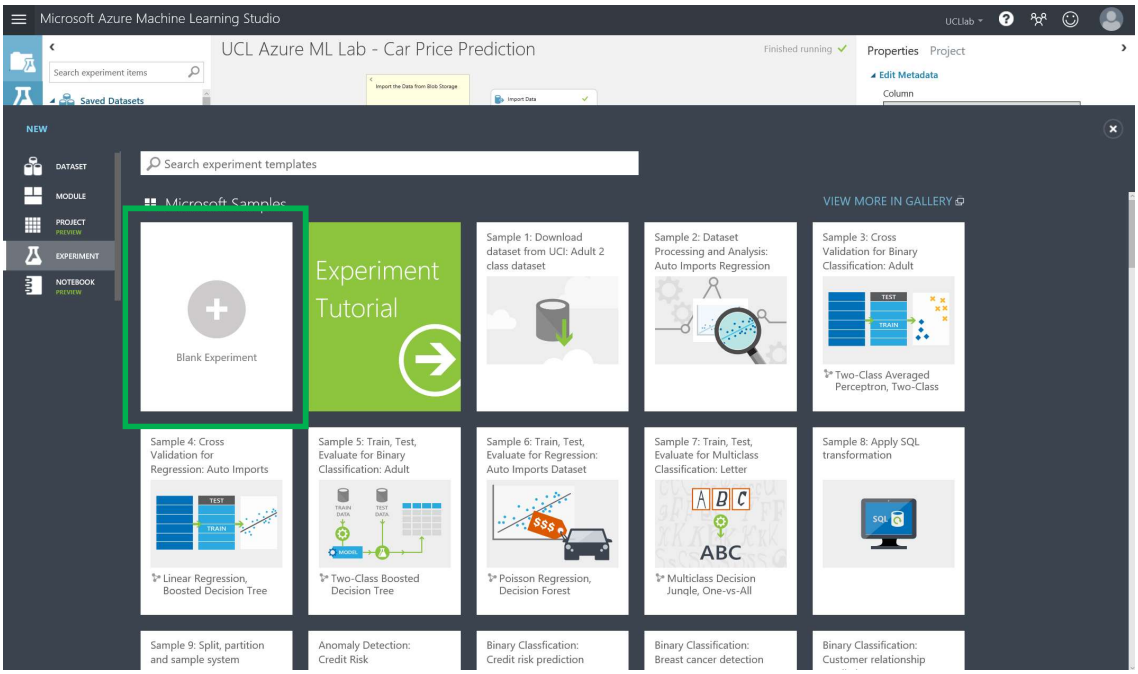
- Azure ML accepts many formats including CSV, TSV, Plain Text and Zip files via upload
- You can use the 'Import Data' Module to access your data from sources such as Azure SQL DB, Blob storage, HDInsight and industry standard OData feeds.

[ADDITIONAL INFORMATION:] It's also possible to adapt an existing experiment to your needs from one of the many samples in the Cortana Intelligence Gallery – you can change it to use your own data and parameters and quickly see how suitable it is for your needs:

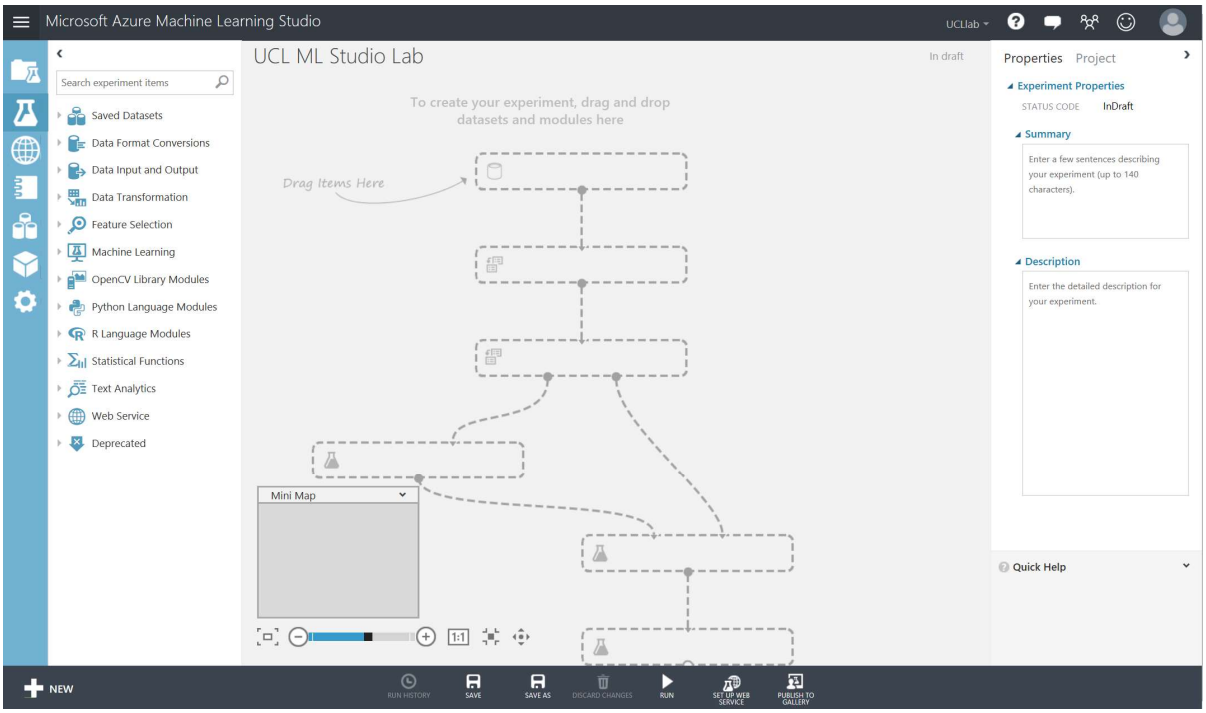


In this lab, we will create a new experiment using the 'Import Data' wizard to read in data from an Azure Blob Storage account.

- Let's start by creating a new experiment: Bottom left corner choose 'New' -> 'Experiment' -> 'Blank Experiment'



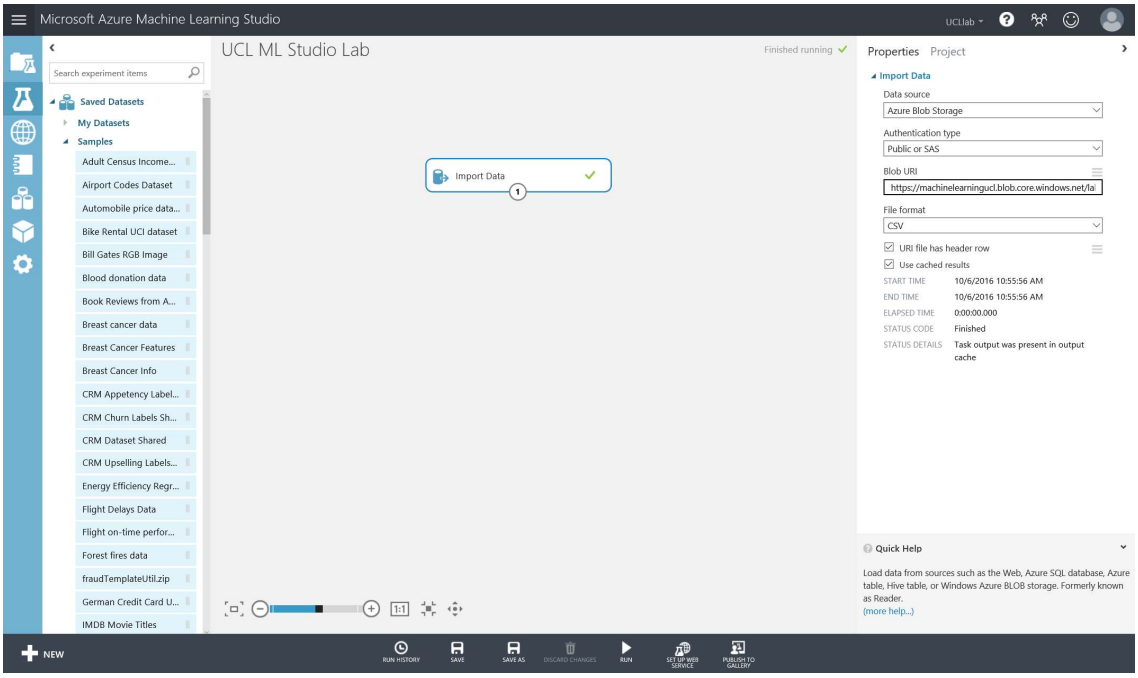
- Change the title of the experiment to a meaningful description, Azure ML Lab



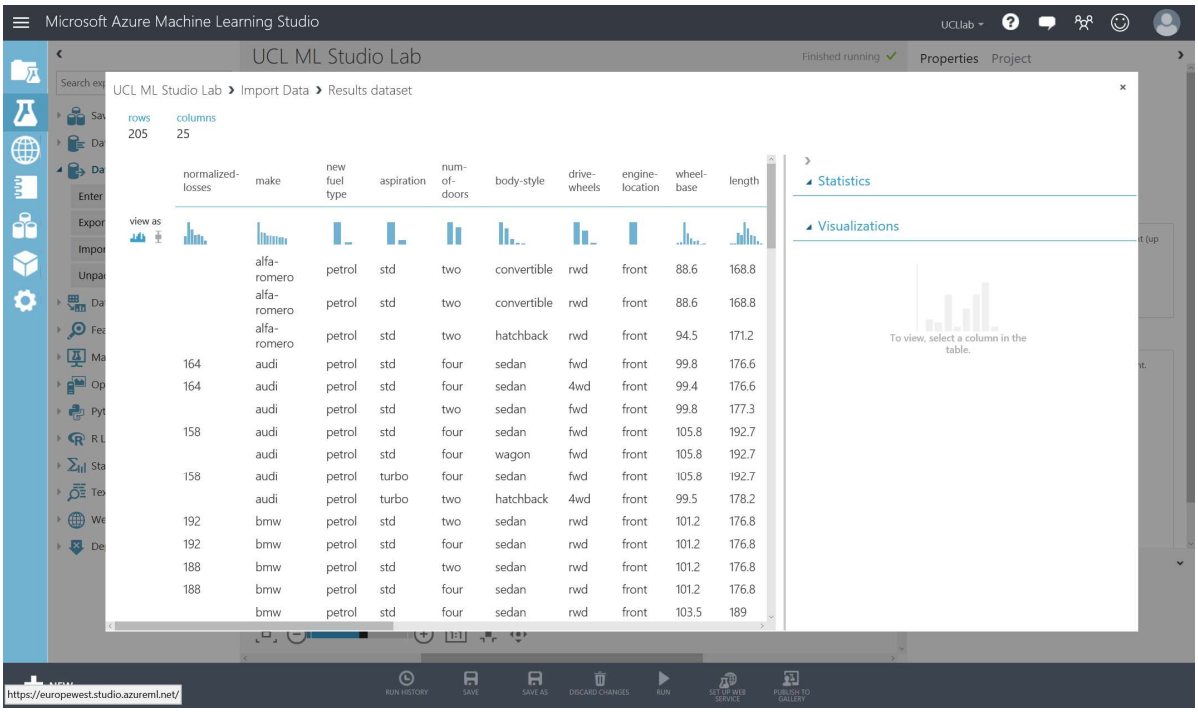
From the modules on the left, choose 'Data Input and Output' and then click and drag 'Import Data' onto the grey workspace. Over on the properties pane, enter the details below. This will allow access to a Car Price Prediction Dataset from an Azure Blob Storage account:

- Data Source: Azure Blob Storage
- Authentication Type: Public or SAS
- Blob URI: https://raw.githubusercontent.com/mithun-prasad/azure-ml/master/Data/carPriceData.csv
- File Format: CSV
- File Has Header Row: CHECKED
- Use Cached Results: CHECKED

After the details are entered, go to the functions toolbar at the bottom of the page and click 'Run'. Once completed processing the 'Import Data' module will have a green tick by it, as shown below



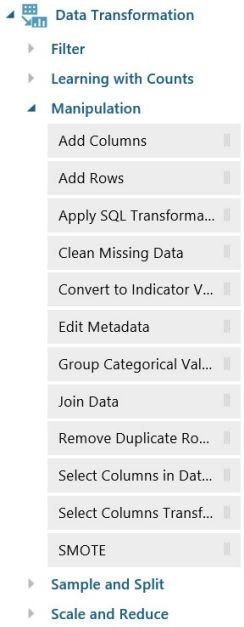
To view the data, hover over the output port of the Import Data Module. Right click on 'Results Dataset' and choose 'Visualise' to see a snapshot of the dataset and summary.



Things to note are:

- The number of rows and columns in the dataset (top left)
- The names of the columns in the dataset
- If you click on any of the columns you will see on the right of the pop up some statistics such as data type and range of values or amount of missing values
- Also, each column will show a histogram of the distribution of the data

Data Pre-Processing

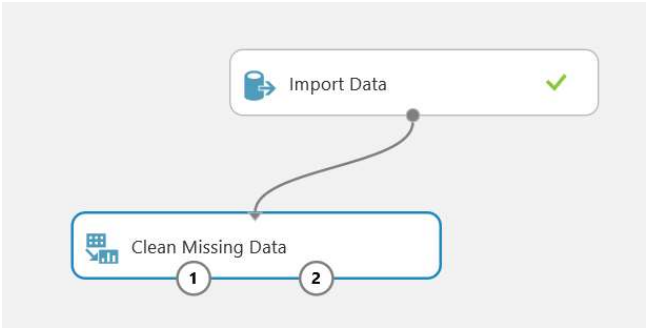


In this part of the lab, we will prepare the dataset into a form that is suitable for building a model.

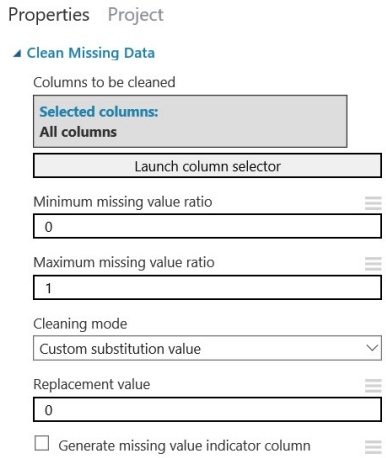
The major data preparation tasks include data cleaning, integration, transformation, reduction, and discretization or quantization. These operations are available in the Data Transformation group in the left panel.

We can start investigating the missing values in the dataset. For example, have a look at the 'normalised losses' column – there are 41 missing values.

Click and drag the 'Clean Missing Data' module in the Data Transformation modules on the left of the screen into the experiment space. Connect the output of the 'Import Data' module to the input port of the 'Clean Missing Data' module



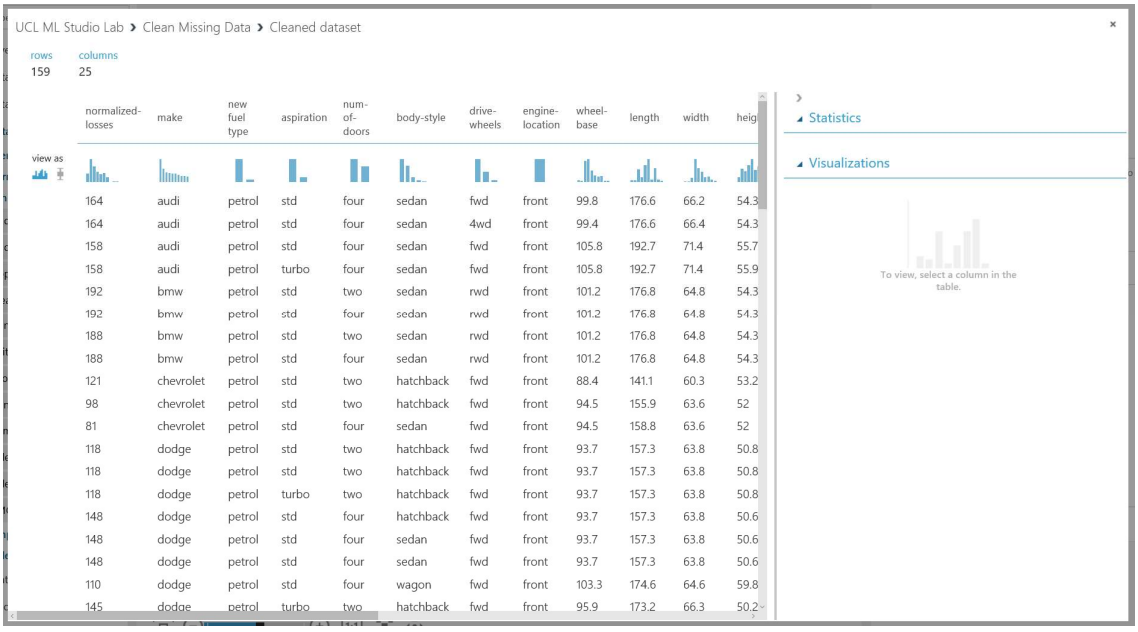
In the properties section on the right, you will see the properties below such as the columns that are selected to operate on as well as what type of cleaning mode will be used



Change the cleaning mode property to 'Remove entire row'. This transformation technique scans the entire dataset for missing values. If any missing value is found, the entire row is removed. As a result, the output of this transformation will be a slightly smaller dataset. However, the dataset records will be complete.

Once properties are set, run the experiment from the bottom toolbar and wait for all modules to have a green tick by them. Once complete choose the left output port of the 'Clean Missing Data' module and right click -> visualise.

What is the number of rows in the transformed data?



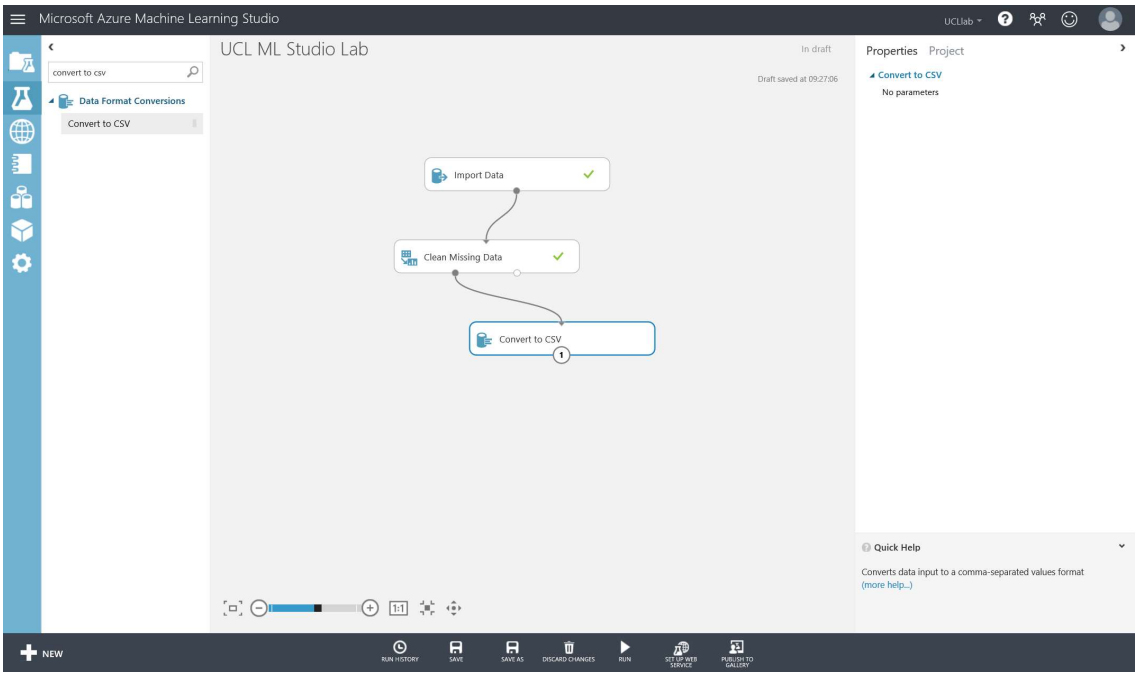
Data Pre-Processing with Python

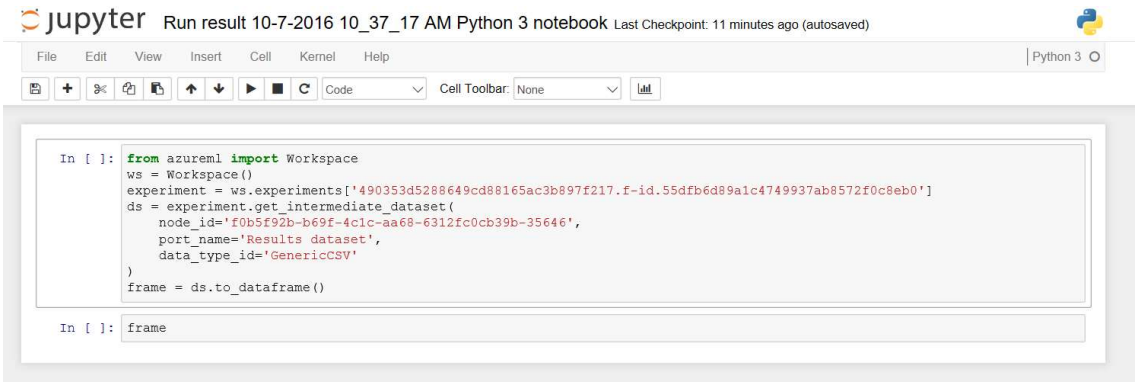
This section introduces Jupyter Notebooks in Azure ML and provides us the flexibility to manipulate the dataset using Python.

The objective of this section is to use python to change a couple of columns of the dataset imported in the previous section using simple python statements. This task is to introduce Jupyter notebooks and not challenge your python skills.

Below are the steps to access the dataset using Jupyter notebooks:

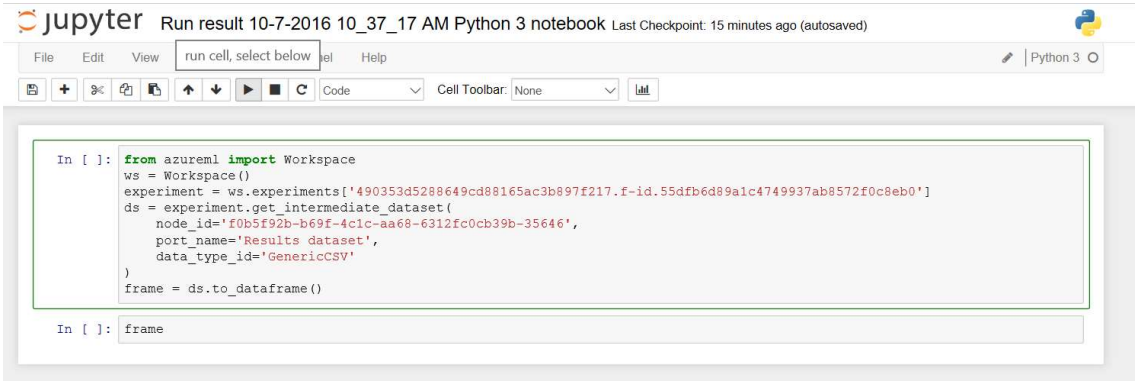
1. It is important to note that in order to access the Jupyter Notebooks functionality from inside the Azure ML Studio, you must first convert the dataset into a CSV. Connect the left output port of the 'Clean Missing Data' module to a 'Convert to CSV' module. Use the search box in the top left of the screen to find this module and connect it up, so it looks like the screenshot below.
2. Once connected, run the experiment again.
3. Right click on the output port of the 'Convert to CSV' file and select 'Open in a new notebook' and 'Python 3'.



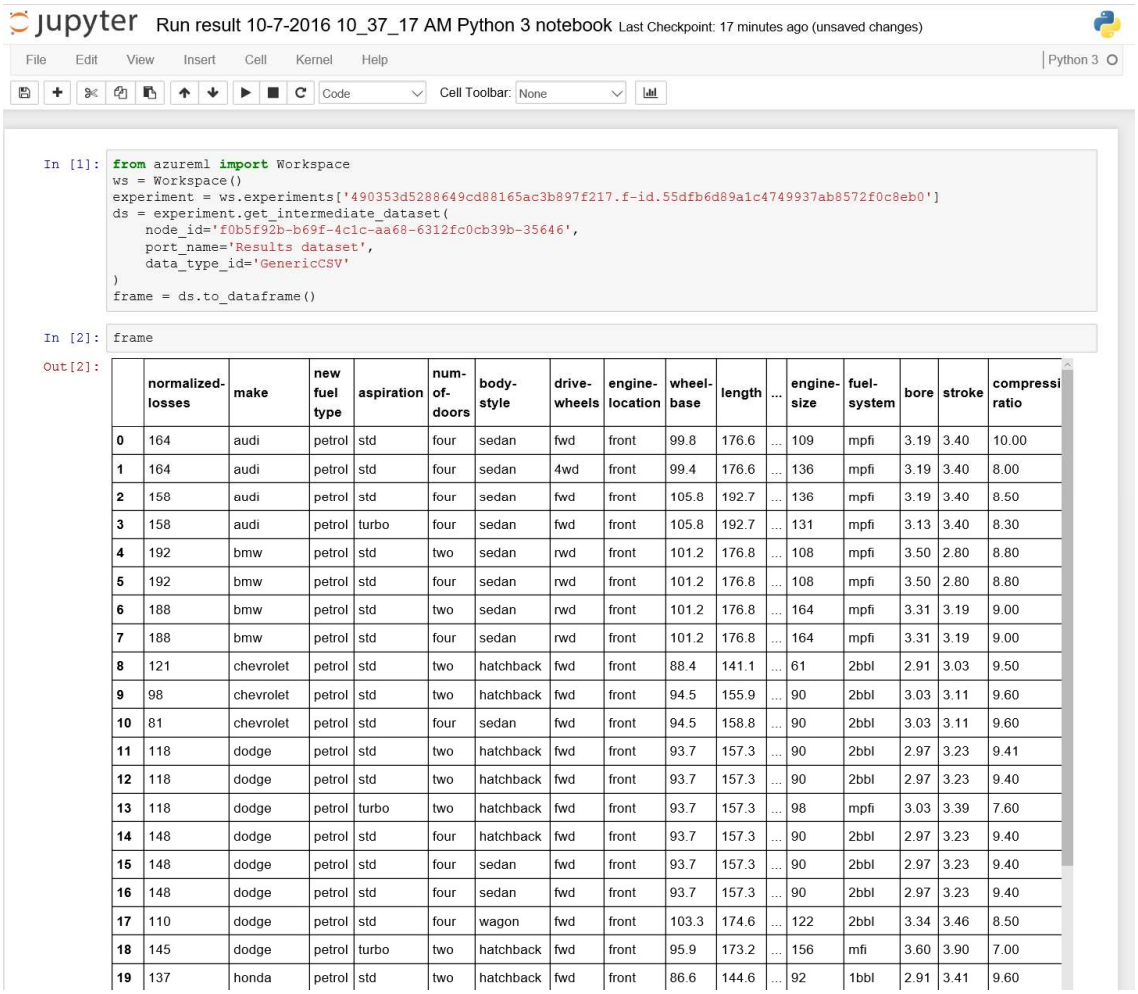


The notebook uses 'azureml' package in the code snippet above to import the dataset from the experiment. Notice the node ID (convert to csv) that the dataset is coming from.

Run the first block of interactive code from the toolbar above and you will find the dataset is pulled into a variable called 'frame' as shown below.

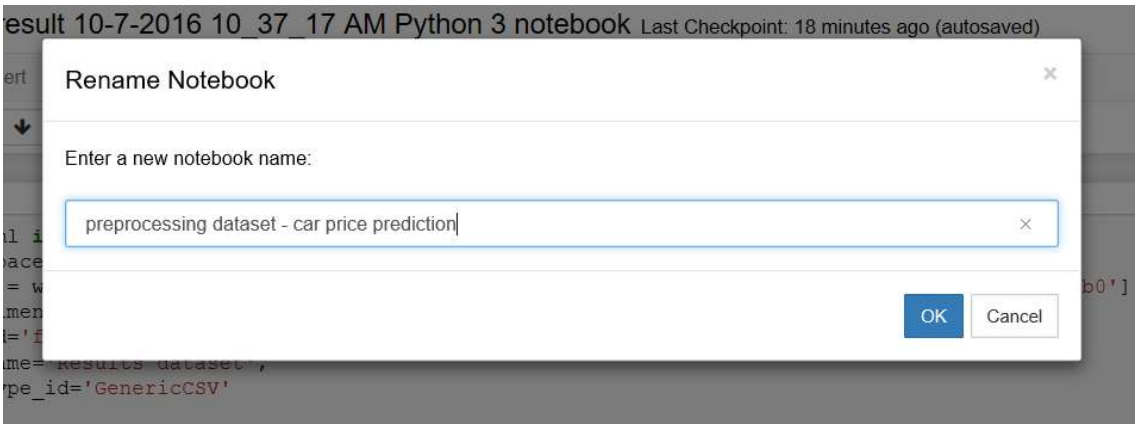


On completion, run the second interactive code block to view the dataset.



	normalized-losses	make	new fuel type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	...	engine-size	fuel-system	bore	stroke	compressi ratio
0	164	audi	petrol	std	four	sedan	fwd	front	99.8	176.6	...	109	mpfi	3.19	3.40	10.00
1	164	audi	petrol	std	four	sedan	4wd	front	99.4	176.6	...	136	mpfi	3.19	3.40	8.00
2	158	audi	petrol	std	four	sedan	fwd	front	105.8	192.7	...	136	mpfi	3.19	3.40	8.50
3	158	audi	petrol	turbo	four	sedan	fwd	front	105.8	192.7	...	131	mpfi	3.13	3.40	8.30
4	192	bmw	petrol	std	two	sedan	rwd	front	101.2	176.8	...	108	mpfi	3.50	2.80	8.80
5	192	bmw	petrol	std	four	sedan	rwd	front	101.2	176.8	...	108	mpfi	3.50	2.80	8.80
6	188	bmw	petrol	std	two	sedan	rwd	front	101.2	176.8	...	164	mpfi	3.31	3.19	9.00
7	188	bmw	petrol	std	four	sedan	rwd	front	101.2	176.8	...	164	mpfi	3.31	3.19	9.00
8	121	chevrolet	petrol	std	two	hatchback	fwd	front	88.4	141.1	...	61	2bbl	2.91	3.03	9.50
9	98	chevrolet	petrol	std	two	hatchback	fwd	front	94.5	155.9	...	90	2bbl	3.03	3.11	9.60
10	81	chevrolet	petrol	std	four	sedan	fwd	front	94.5	158.8	...	90	2bbl	3.03	3.11	9.60
11	118	dodge	petrol	std	two	hatchback	fwd	front	93.7	157.3	...	90	2bbl	2.97	3.23	9.41
12	118	dodge	petrol	std	two	hatchback	fwd	front	93.7	157.3	...	90	2bbl	2.97	3.23	9.40
13	118	dodge	petrol	turbo	two	hatchback	fwd	front	93.7	157.3	...	98	mpfi	3.03	3.39	7.60
14	148	dodge	petrol	std	four	hatchback	fwd	front	93.7	157.3	...	90	2bbl	2.97	3.23	9.40
15	148	dodge	petrol	std	four	sedan	fwd	front	93.7	157.3	...	90	2bbl	2.97	3.23	9.40
16	148	dodge	petrol	std	four	sedan	fwd	front	93.7	157.3	...	90	2bbl	2.97	3.23	9.40
17	110	dodge	petrol	std	four	wagon	fwd	front	103.3	174.6	...	122	2bbl	3.34	3.46	8.50
18	145	dodge	petrol	turbo	two	hatchback	fwd	front	95.9	173.2	...	156	mfi	3.60	3.90	7.00
19	137	honda	petrol	std	two	hatchback	fwd	front	86.6	144.6	...	92	1bbl	2.91	3.41	9.60

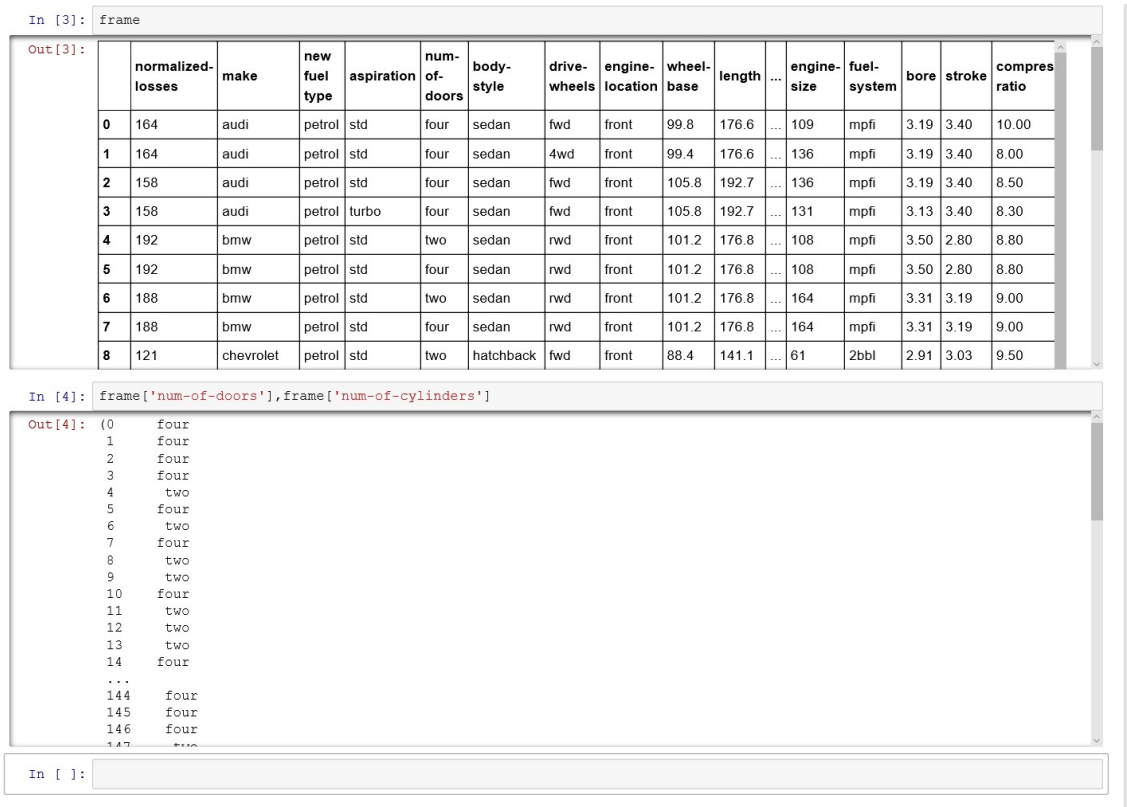
It is now a good time to rename the notebook to a meaningful name. Choose File -> Rename … and choose a filename such as: "preprocessing dataset – car price prediction" and click OK.



In this dataset, a couple of columns ('num-of-doors' and 'num-of-cylinders') represent numeric values as strings. Change their values to integer types and feed back into the experiment the process to do that.

Let's view the two columns mentioned above by running the below code:

```
frame['num-of-doors'], frame['num-of-cylinders']
```



In [3]: frame

Out[3]:

	normalized-losses	make	new fuel type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	...	engine-size	fuel-system	bore	stroke	compres ratio
0	164	audi	petrol	std	four	sedan	fwd	front	99.8	176.6	...	109	mpfi	3.19	3.40	10.00
1	164	audi	petrol	std	four	sedan	4wd	front	99.4	176.6	...	136	mpfi	3.19	3.40	8.00
2	158	audi	petrol	std	four	sedan	fwd	front	105.8	192.7	...	136	mpfi	3.19	3.40	8.50
3	158	audi	petrol	turbo	four	sedan	fwd	front	105.8	192.7	...	131	mpfi	3.13	3.40	8.30
4	192	bmw	petrol	std	two	sedan	rwd	front	101.2	176.8	...	108	mpfi	3.50	2.80	8.80
5	192	bmw	petrol	std	four	sedan	rwd	front	101.2	176.8	...	108	mpfi	3.50	2.80	8.80
6	188	bmw	petrol	std	two	sedan	rwd	front	101.2	176.8	...	164	mpfi	3.31	3.19	9.00
7	188	bmw	petrol	std	four	sedan	rwd	front	101.2	176.8	...	164	mpfi	3.31	3.19	9.00
8	121	chevrolet	petrol	std	two	hatchback	fwd	front	88.4	141.1	...	61	2bbl	2.91	3.03	9.50

In [4]: frame['num-of-doors'],frame['num-of-cylinders']

Out[4]:
```
(0     four
 1     four
 2     four
 3     four
 4      two
 5     four
 6      two
 7     four
 8      two
 9      two
 10    four
 11     two
 12     two
 13     two
 14    four
 ...
 144    four
 145    four
 146    four
```

In [ ]:

We can swap the string values in the data frame with numeric values. For the num-of-doors column, run the below code:

```
newdf=frame.replace(['two','four'],['2','4'])
```

```
newdf
```

```
In [5]: newdf=frame.replace(['two','four'],['2','4'])
        newdf
```

Out[5]:

	normalized-losses	make	new fuel type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	...	engine-size	fuel-system	bore	stroke	compres ratio
0	164	audi	petrol	std	4	sedan	fwd	front	99.8	176.6	...	109	mpfi	3.19	3.40	10.00
1	164	audi	petrol	std	4	sedan	4wd	front	99.4	176.6	...	136	mpfi	3.19	3.40	8.00
2	158	audi	petrol	std	4	sedan	fwd	front	105.8	192.7	...	136	mpfi	3.19	3.40	8.50
3	158	audi	petrol	turbo	4	sedan	fwd	front	105.8	192.7	...	131	mpfi	3.13	3.40	8.30
4	192	bmw	petrol	std	2	sedan	rwd	front	101.2	176.8	...	108	mpfi	3.50	2.80	8.80
5	192	bmw	petrol	std	4	sedan	rwd	front	101.2	176.8	...	108	mpfi	3.50	2.80	8.80
6	188	bmw	petrol	std	2	sedan	rwd	front	101.2	176.8	...	164	mpfi	3.31	3.19	9.00
7	188	bmw	petrol	std	4	sedan	rwd	front	101.2	176.8	...	164	mpfi	3.31	3.19	9.00
8	121	chevrolet	petrol	std	2	hatchback	fwd	front	88.4	141.1	...	61	2bbl	2.91	3.03	9.50

```
In [ ]:
```

In addition, the num-of-cylinder column also has other values to be changed. To change them, run the below code:

```
newdf =
frame.replace(['four','six','five','eight','two','three','twelve'],[
'4','6','5','8','2','3','12'])

newdf['num-of-cylinders'],newdf['num-of-doors']
```

Now that the data types and values in the dataset are changes, we want to be able to use this dataset back in the code. There are a couple of ways to do this:

1. If it is short snippets of code (such as above) we can take this code and copy it into an 'Execute Python' Module back in the experiment space
2. For a more complex function, you can tidy up the notebook to contain the function call and return a data frame. Save the notebook, by using File -> download as -> Python (.py) file and upload to execute python module in the .zip input port.

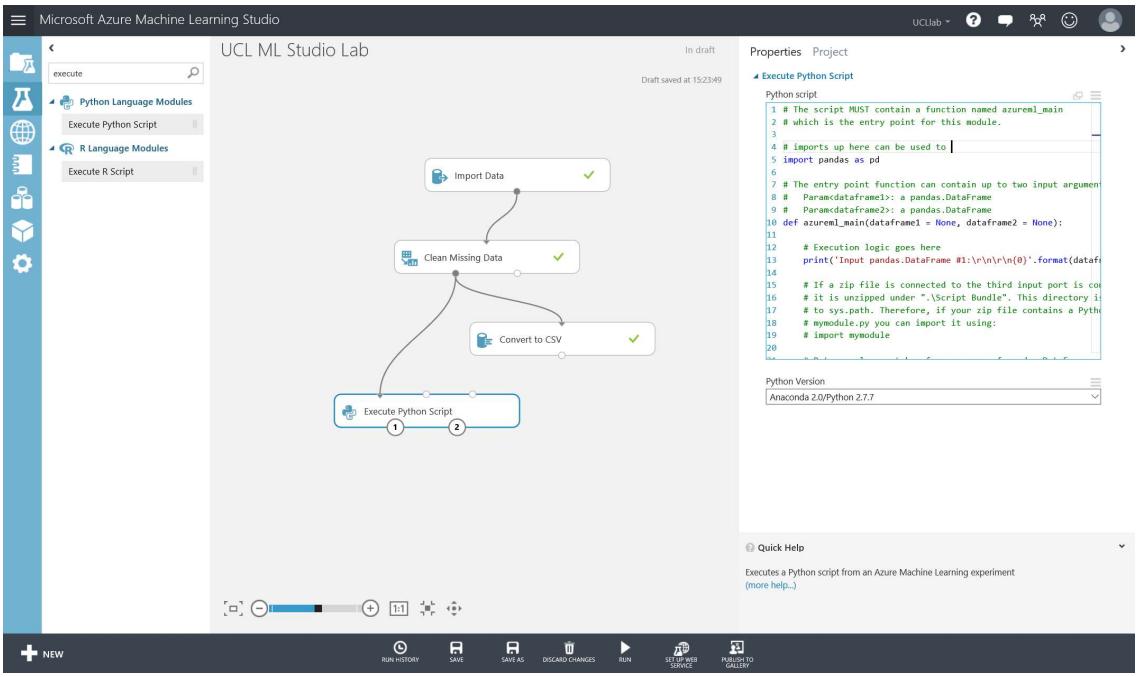
In this lab, we will explore 1.

Now let's edit the dataset using the code we just tested in the Python Notebooks. Search for the module 'Execute Python Script' and drag onto the design surface.

Connect the first port of the Python module to the left output port of the 'Clean Missing Data' module as shown below. The Execute Python Script module has 3 input ports and 2 output ports. It can take in two datasets and .zip files that could contain python packages to import.

This module also provides flexibility to output the whole data frame to continue your experiment. Additionally, 'Python Device' is also optional that allows the user to visualize plots created in the python code.

To find out more about this modules capability check out the documentation here: https://azure.microsoft.com/en-gb/documentation/articles/machine-learning-execute-python-scripts/



Read and copy the code below into the python script box:

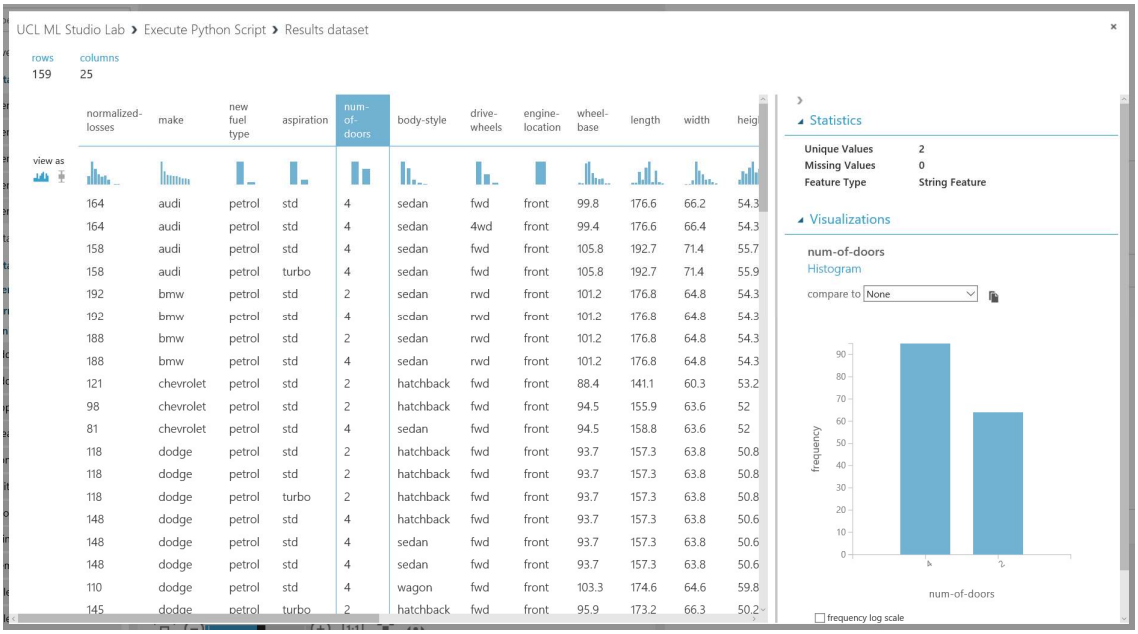
```
# imports up here can be used to
import pandas as pd

# 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('Input pandas.DataFrame
#1:\r\n\r\n{0}'.format(dataframe1))
    newdf =
dataframe1.replace(['four','six','five','eight','two','three','twelv
e'],['4','6','5','8','2','3','12'])
```

```
# Return value must be of a sequence of pandas.DataFrame
return newdf,
```

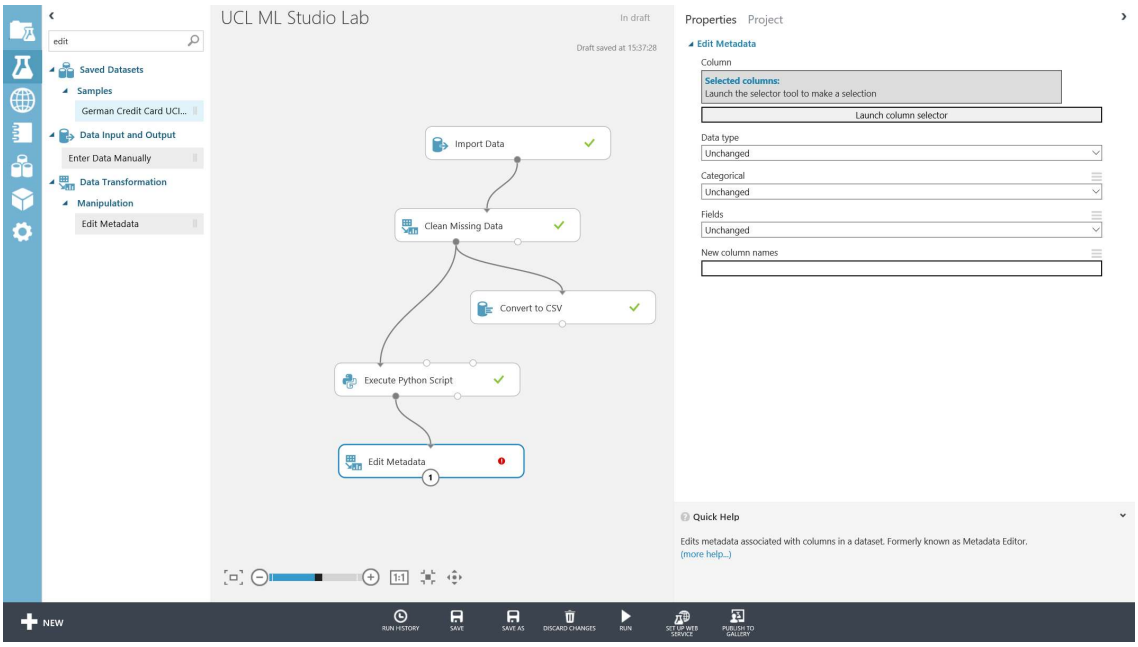
Run the experiment and once completed, right click and visualise the left output port of the 'Execute Python Script' module. You should see new values for 'num-of-doors' and 'num-of-cylinder'.



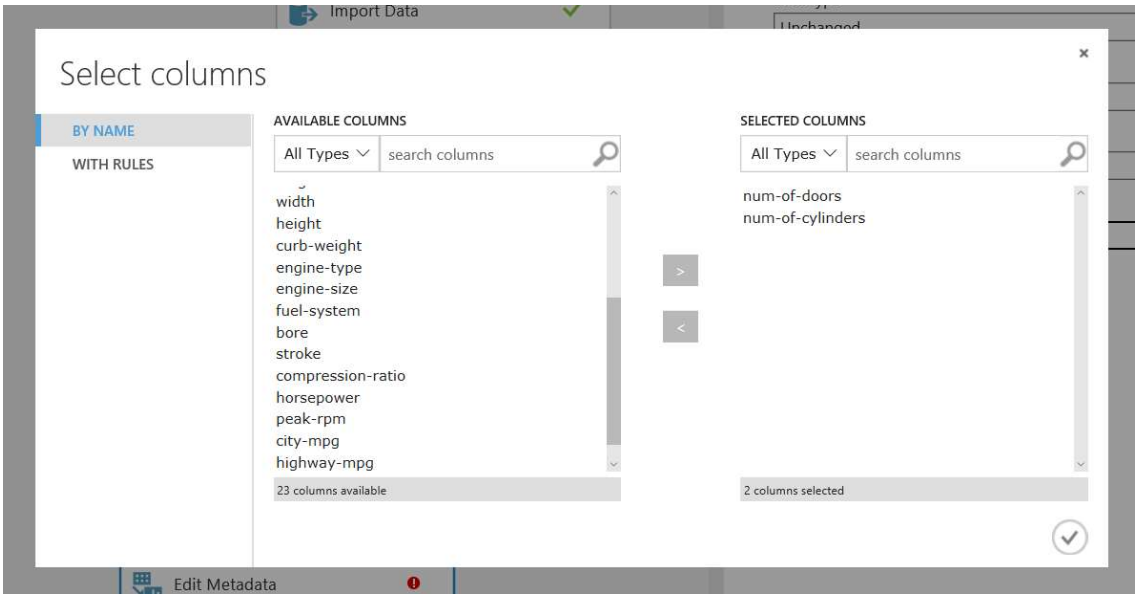
However, there is one more step left with column manipulation. You will notice that on the right of the visualise data pane, the feature type for num-of-doors is still string feature. This needs to be changed.

Metadata

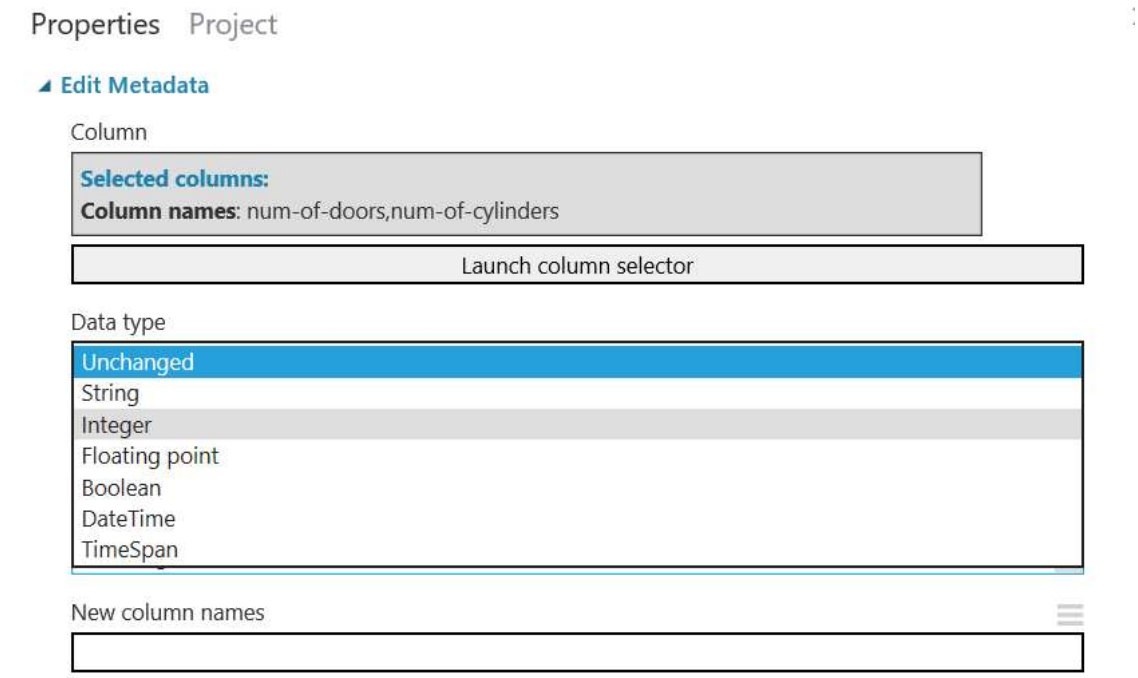
1. Find the 'Edit Metadata' module under Data Transformation, Manipulation.
2. Drag it to the experiment area.
3. Connect the left output port (dataset) of the 'Execute Python Script' module to the input port of the 'Edit Metadata' module.



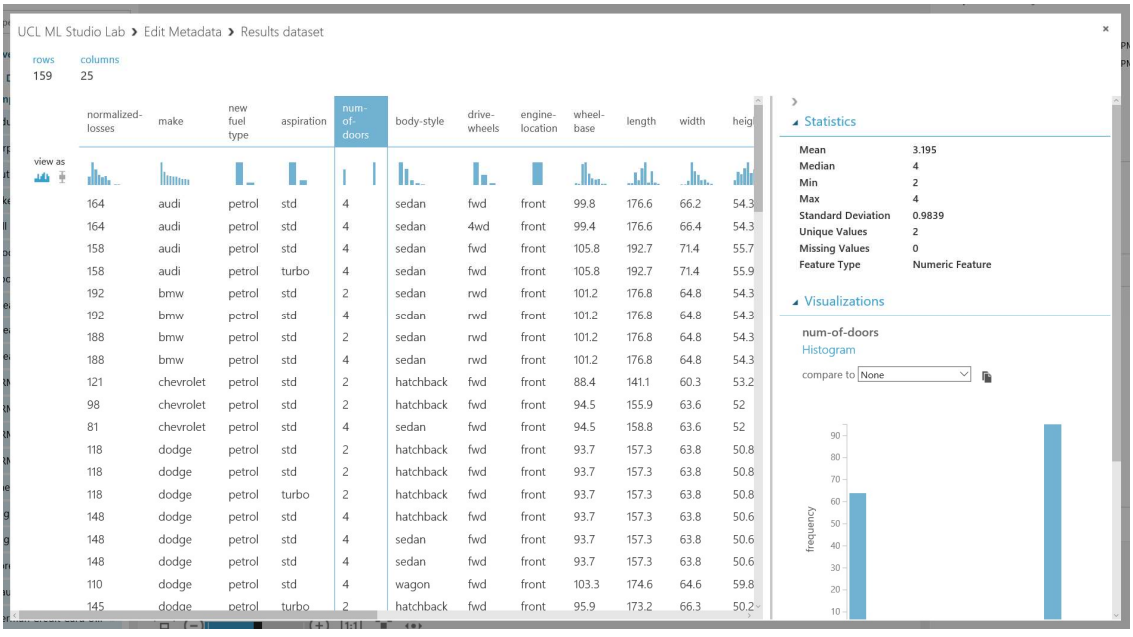
On the properties pane, launch column selector and choose 'num-of-doors' and 'num-of-cylinders' columns as below and click the tick button to complete.



Change the data type to integer using the drop down available. Run the experiment.



After the experiment is run, visualise the data from the Edit Metadata module and confirm that the columns selected have now been changed to numeric data type.



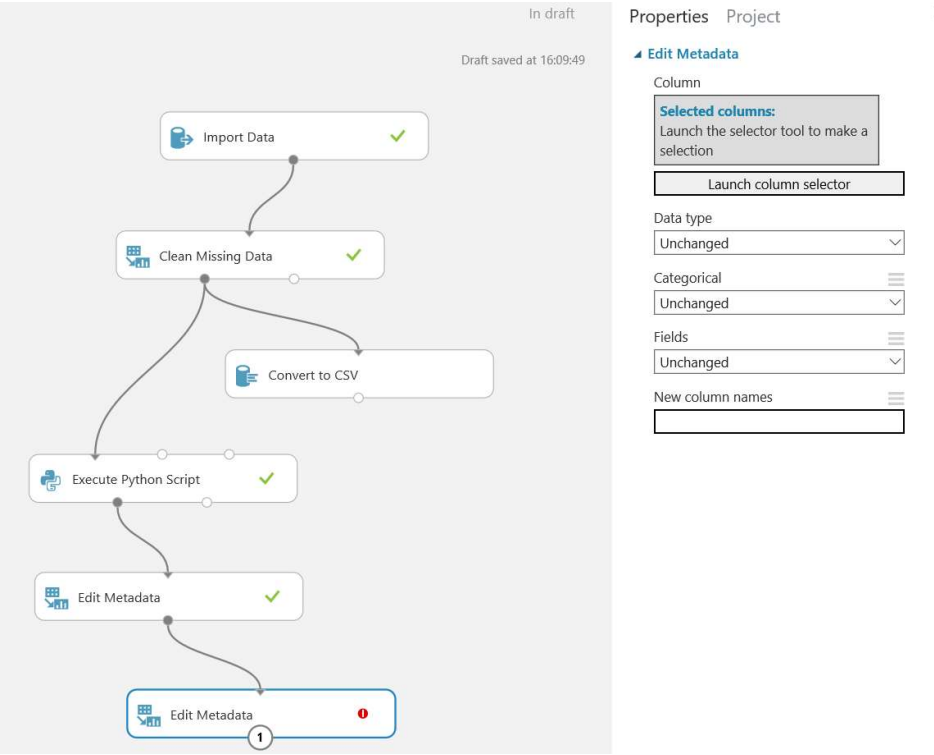
Training the Model

The pre-processed data is now ready for training. In this section, we will build a regression model to predict the price of a car.

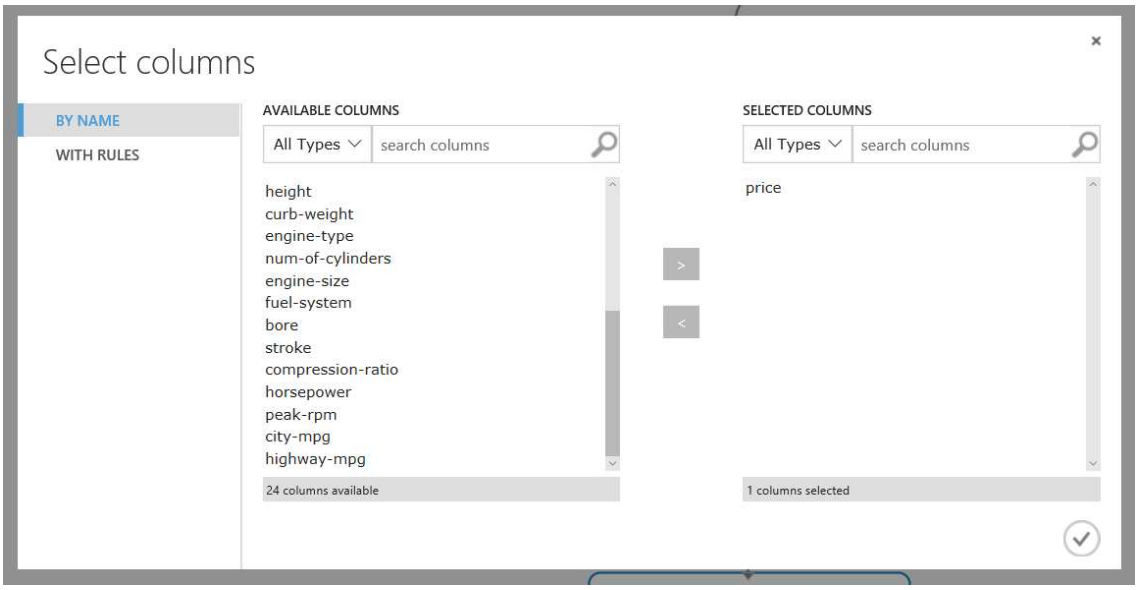
The first step in Supervised Learning is to identify column that is a label and features.

- Labels: Our label is price – as we want to predict the value of a car
- Features: The features are rest of the columns

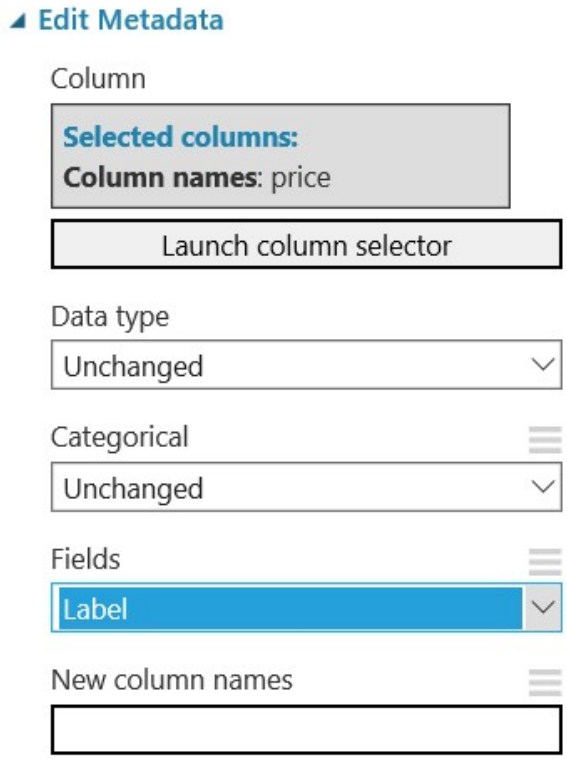
Drag another 'Edit Metadata' module to the experiment space and launch the column selector from the properties pane.



Select the 'price' column in the dataset and click the tick:

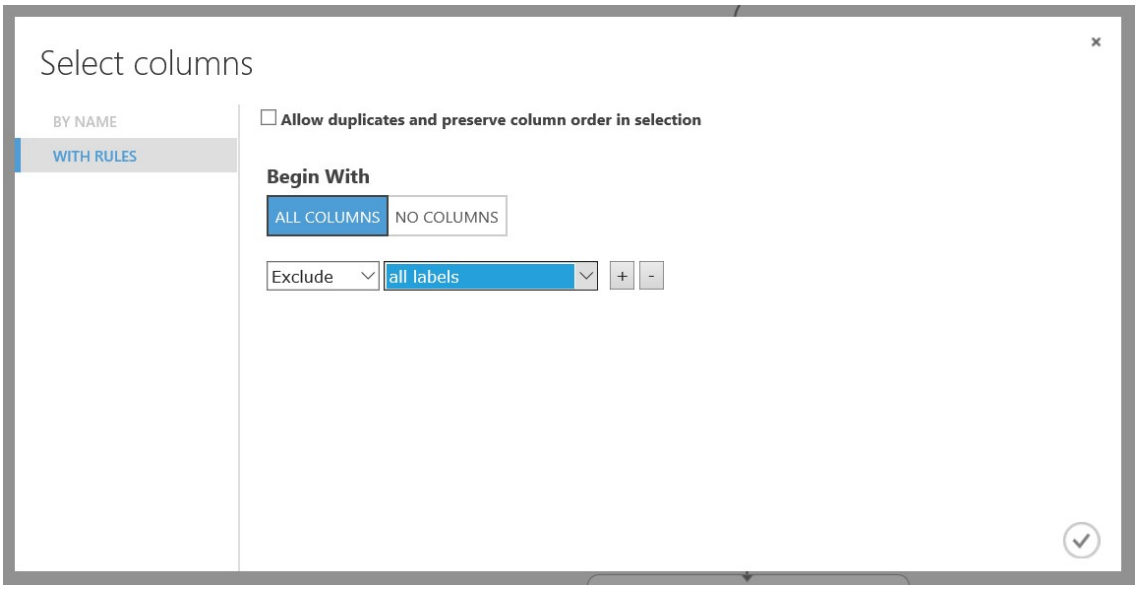


Open the Field property and select 'Label' from the drop-down menu

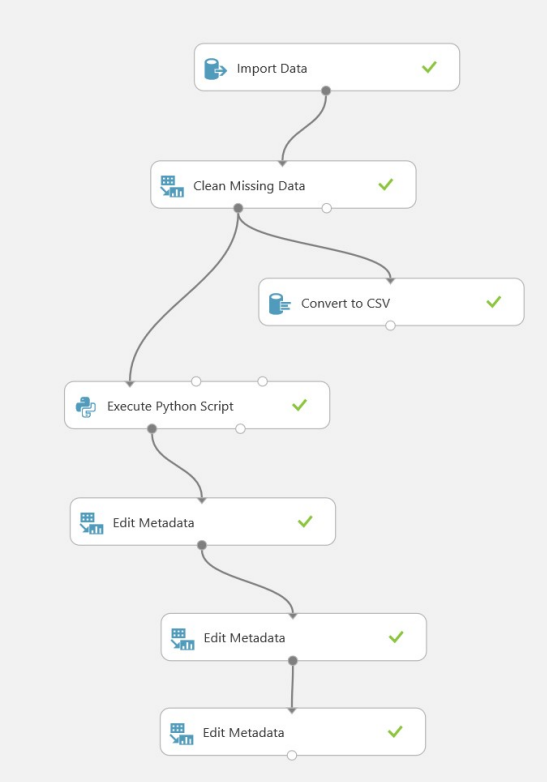


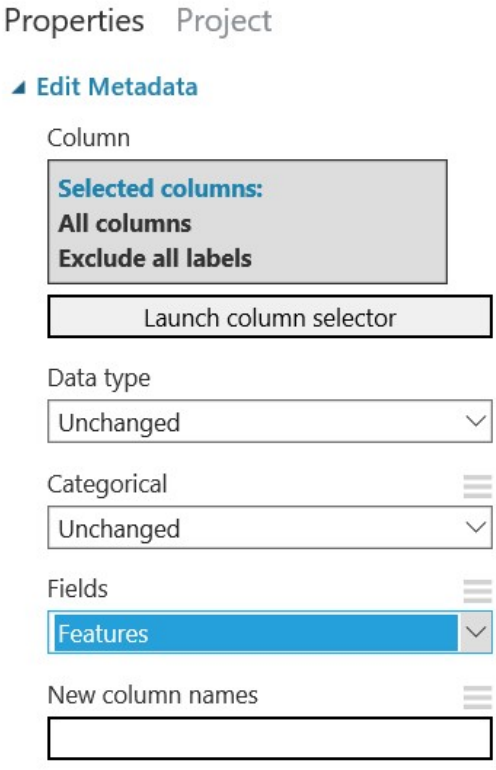
Repeat the same process for features. Select another 'Edit Metadata' module and drag onto the experiment surface and connect to the bottom of the previous module. In the properties column open the 'Launch Column Selector'.

In the column selector, select 'With Rules' on the left. Pick 'Begin With' 'All Columns' and 'Exclude' 'All Labels'. See the screenshot below for clarification:



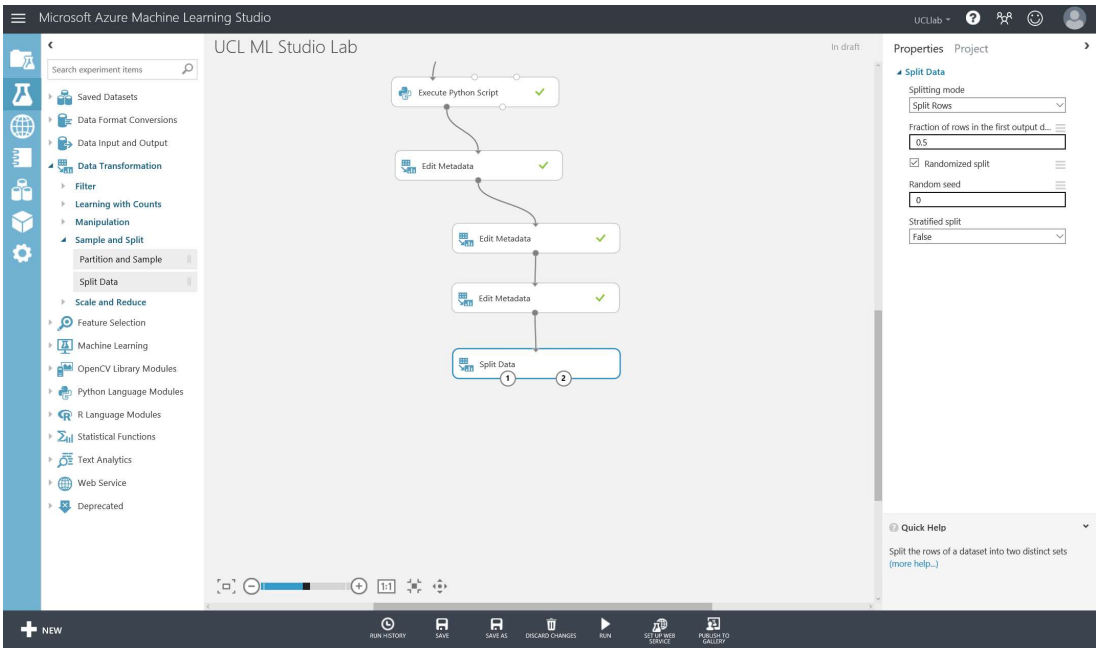
Select the tick button and then back in the properties pane set Field to Features as shown below. And run the experiment:



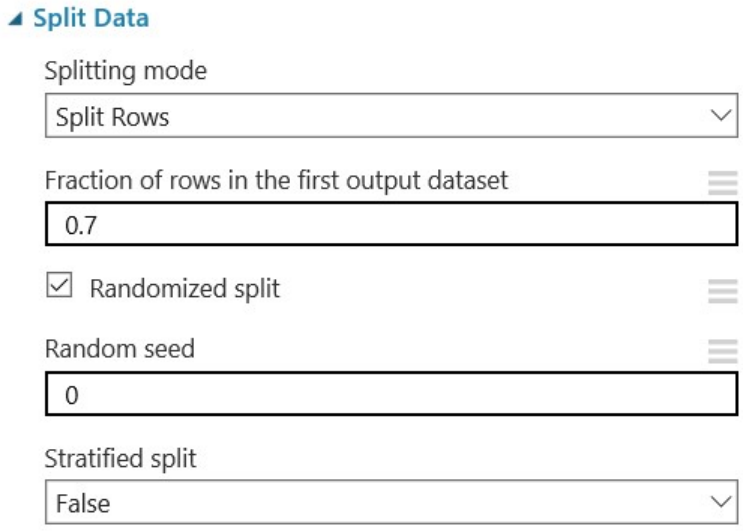


Splitting the dataset

In this section, we will split the dataset to perform training and testing. Find the 'Split Data' module under the Data Transformation -> Sample and Split headings in the module list. Drag it to the experiment space and connect the output port of the last module in the experiment.

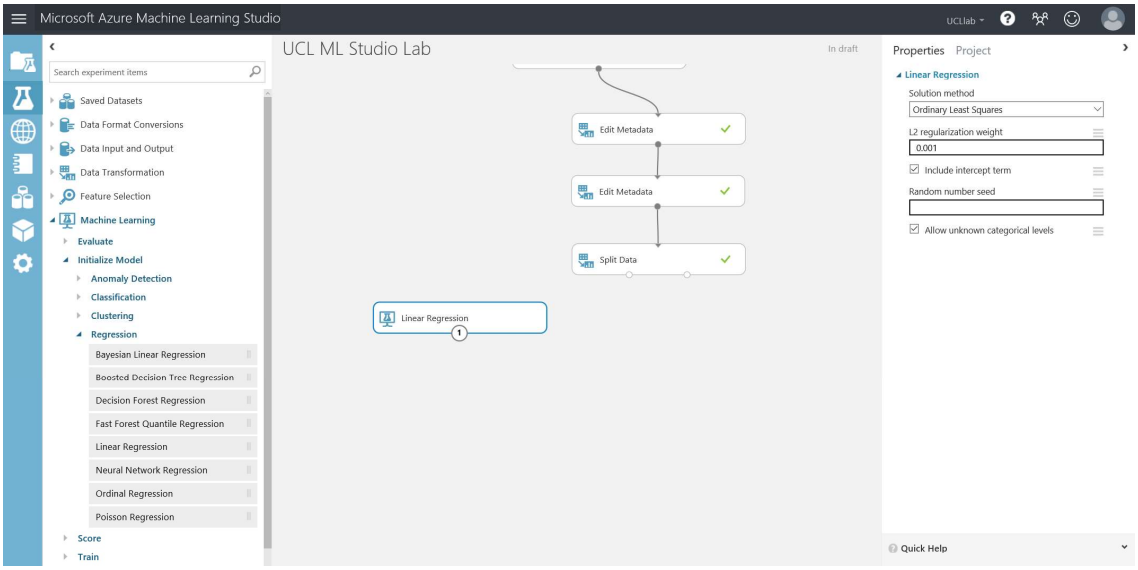


Open the properties pane and pick the 'splitting method' as 'split rows'. We will split the dataset into 70% training and 30% testing. In the 'Fraction of rows in the first output dataset', put 0.7 for the training dataset. This will automatically mean that 30% of the data will be put into the second output port of the module for the test dataset.



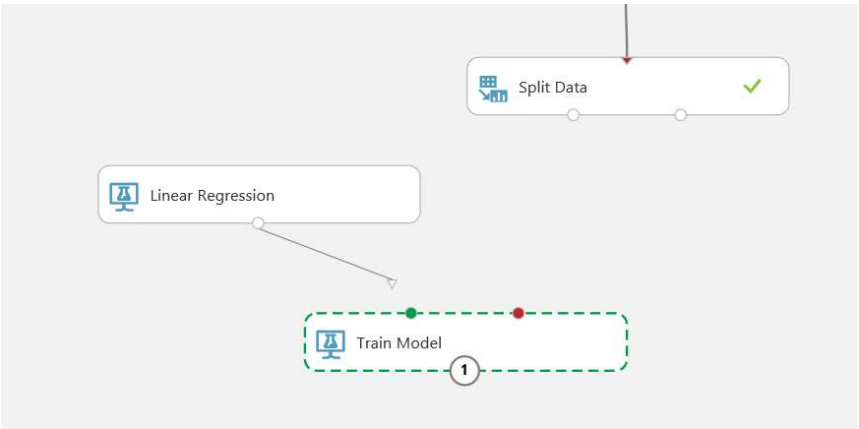
Select an algorithm to train using your datasets. In the modules on the left find:

Machine Learning -> Initialise Model -> Regression section and find 'Linear Regression'. Drag this module to the experiment space.

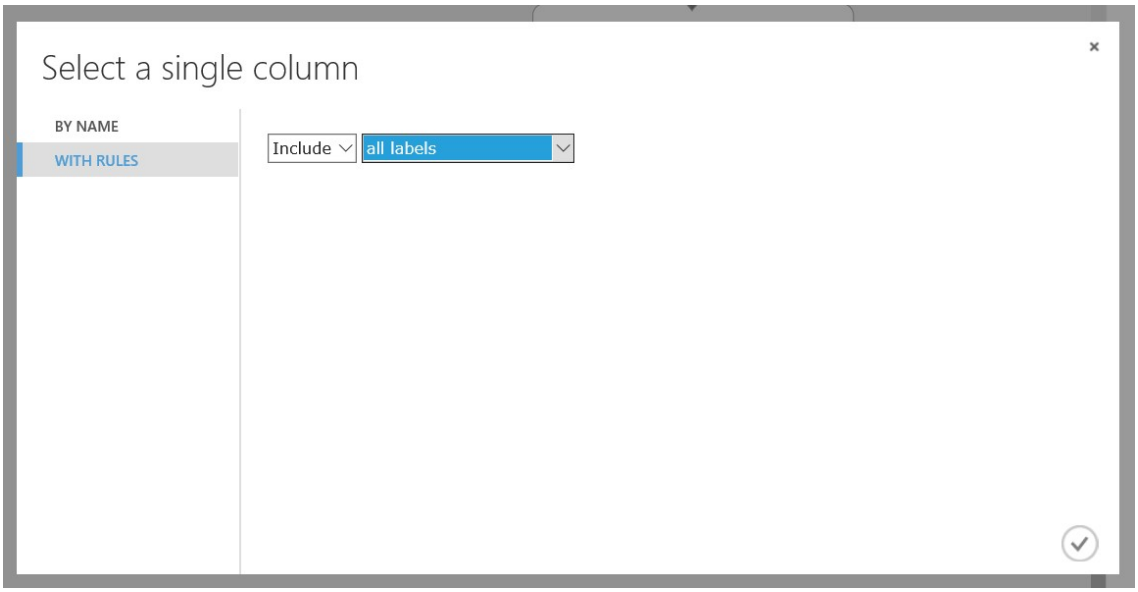


To combine the algorithm and our dataset, introduce the 'Train Model' module in the experiment space. This takes an untrained algorithm and dataset as inputs and produces a trained model out of the module output port.

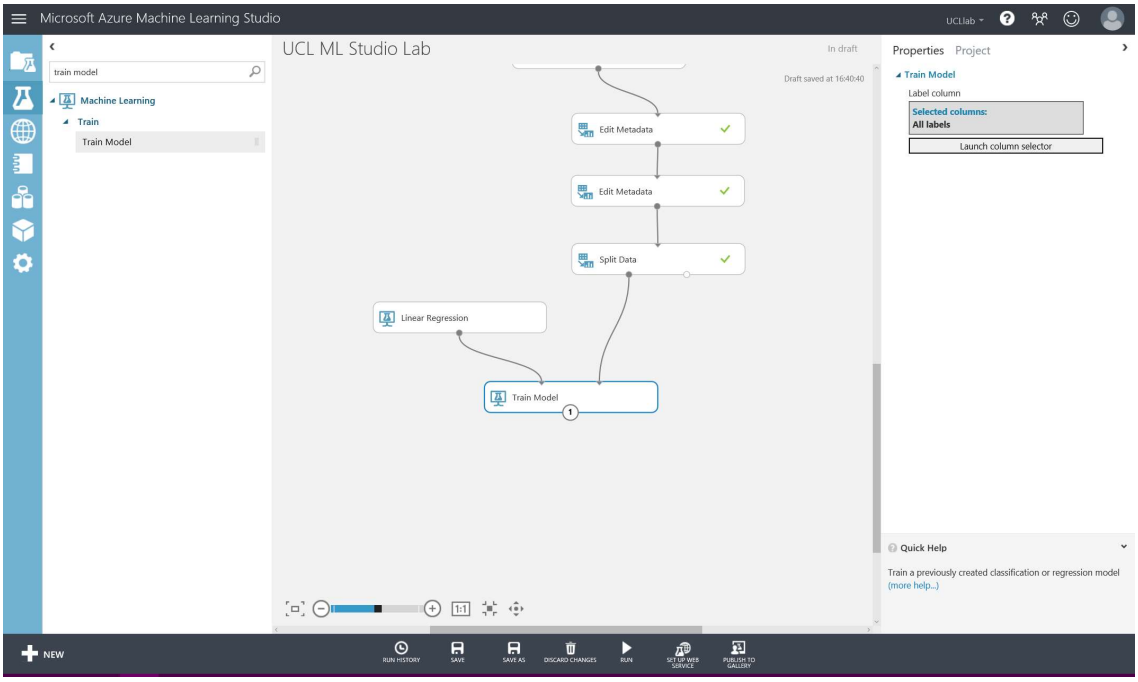
Connect the linear regression algorithm to the left input port of the Train Model module, then connect your 70% training dataset (left output port of Split Data) to the right input port of the Train Model module.



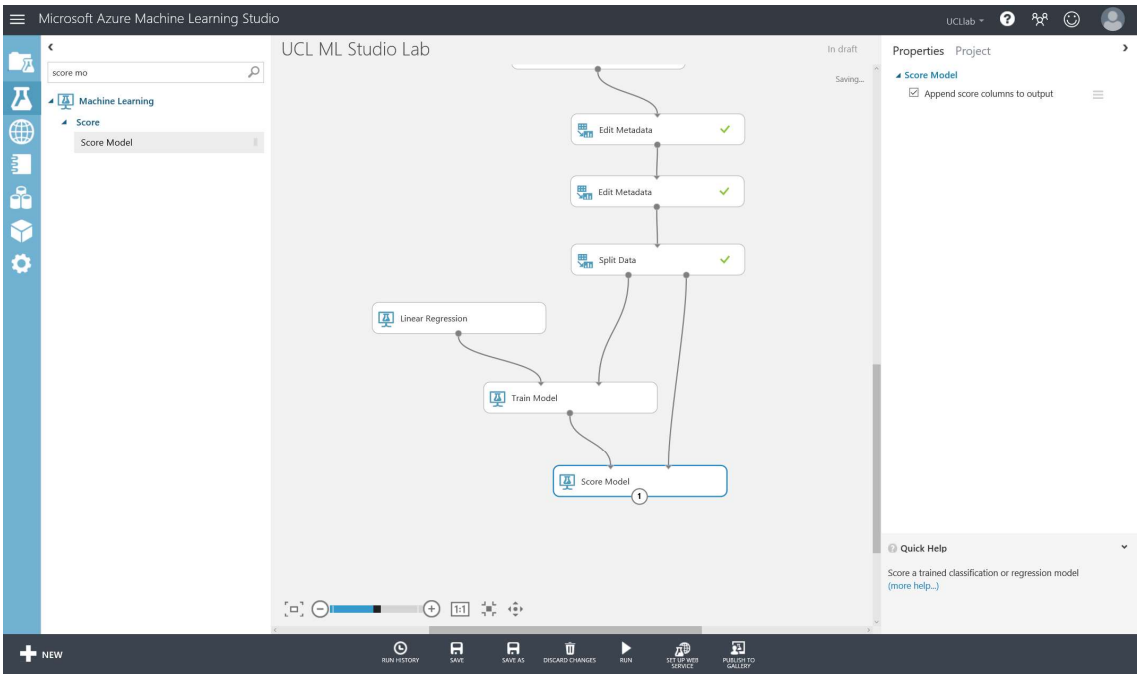
Once connected, select the Train Model module and in the properties pane, Launch the Column Selector. Choose the 'With Rules' section on the left and select option 'Include' with 'All Labels'. This allows the train model module to focus on learning the price of the car (label) given the attributes and patterns in the dataset (features).



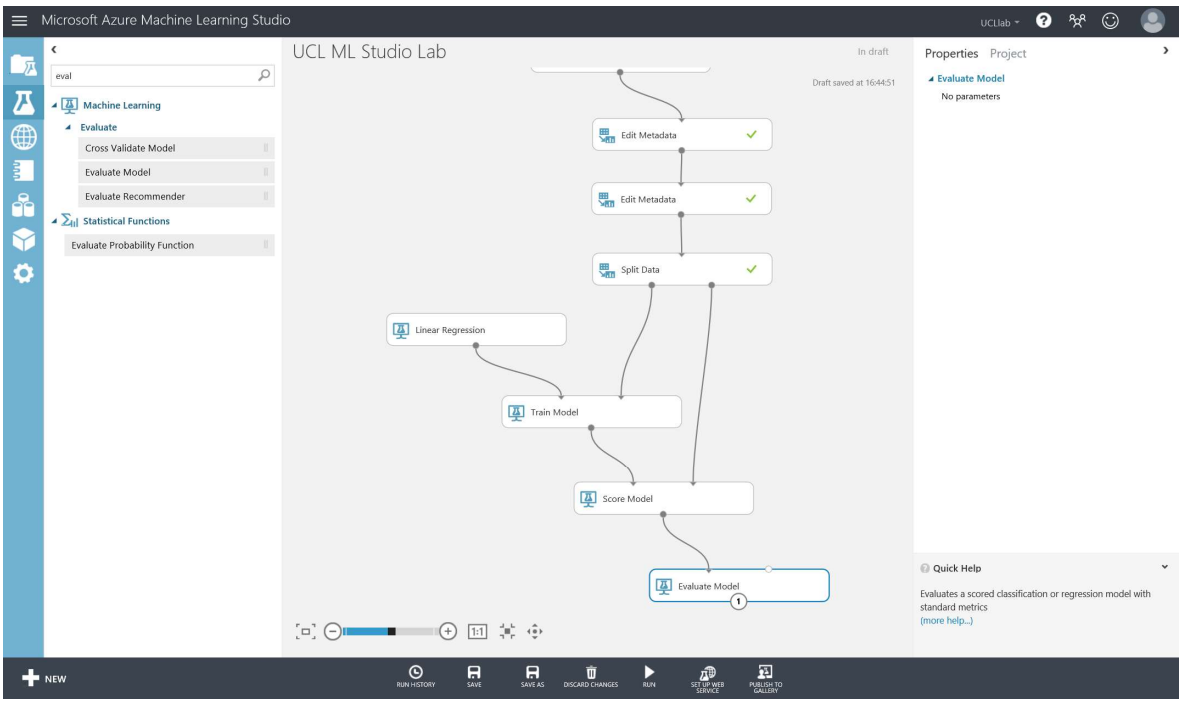
Select the tick box.



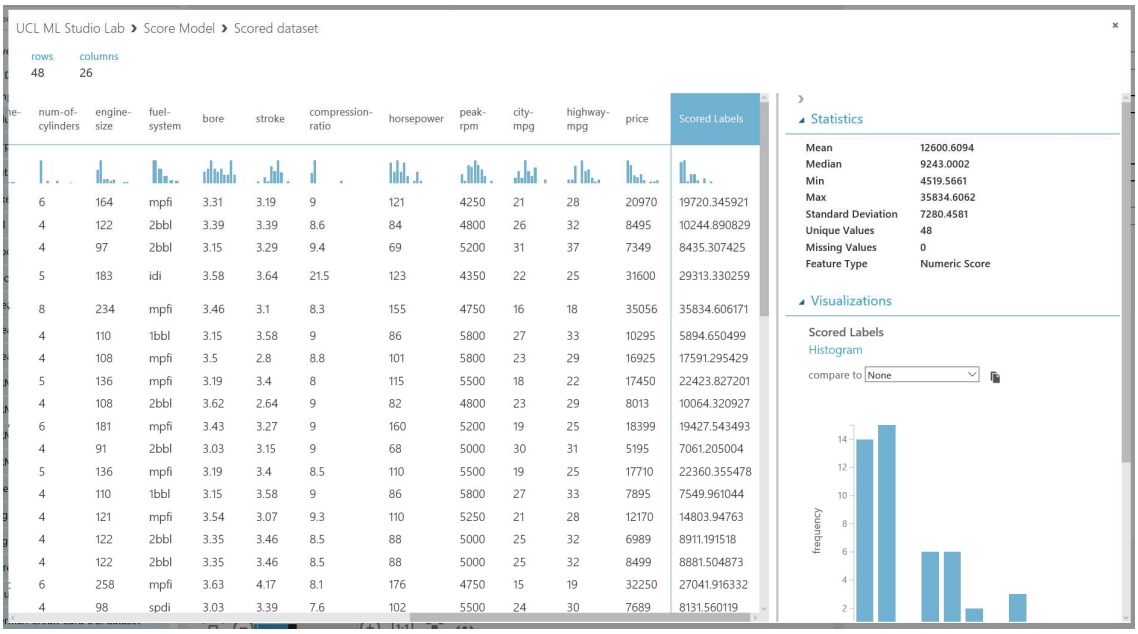
After training, we need to score the model on how well it can predict the price of a car. Connect the trained model to a 'Score Model' module and the testing dataset (the right output port of the Split Data module).



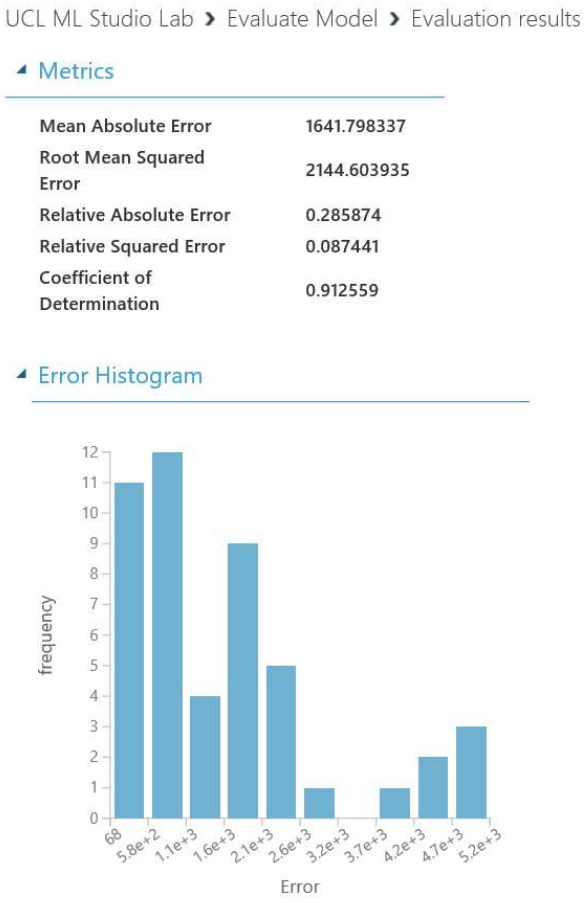
Connect 'Evaluate Model' module to the output of the 'Score Model' module. Notice the 'Evaluate Model' module has two input ports. At this point, connect the score model module to the left port of the evaluate model module as shown below and Run the experiment.



To view the result, select the output of the Score Model module and right click -> visualise data. Scroll to the end of the dataset and find the Scored Labels column. This is the predicted column. Compare this with the actual labels next to it, 'price'.



The following metrics are reported for evaluating regression models. All metrics are reported but the models are ranked by the metric you select for evaluation.



Mean absolute error (MAE) measures how close the predictions are to the actual outcomes; thus, a lower score is better.

Root mean squared error (RMSE) creates a single value that summarizes the error in the model. By squaring the difference, the metric disregards the difference between over-prediction and under-prediction.

Relative absolute error (RAE) is the relative absolute difference between expected and actual values; relative because the mean difference is divided by the arithmetic mean.

Relative squared error (RSE) similarly normalizes the total squared error of the predicted values by dividing by the total squared error of the actual values.

Coefficient of determination, often referred to as R2, represents the predictive power of the model as a value between 0 and 1. Zero means the model is random (explains nothing); 1 means there is a perfect fit. However, caution should be used in interpreting R2 values, as low values can be entirely normal and high values can be suspect.

Conclusion

This lab introduced you to the basic steps in building a Machine Learning model in the Azure Machine Learning Studio. In the next lab, we will explore how we can compare algorithms inside the studio.