Microsoft

Machine Learning

Azure Machine Learning Studio Lab

Contents

What is Azure Machine Learning?

This lab will introduce you to machine learning capabilities available in Microsoft Azure, specifically Microsoft Azure Machine Learning (Azure ML) and the Studio options.

Azure ML is a fully managed machine learning platform that allows you to perform predictive analytics. Development of models (experiments) is achieved using the Azure ML Studio, a web based design environment that empowers both data scientists and domain specialists to build end-to-end solutions and significantly reduce the complexity of building and publishing predictive models. It has a simple drag-and-drop authoring interface and a catalogue of modules that provide functionality for an end-to-end workflow. More experienced users can also embed their own Python or R scripts inline and explore the data interactively with Jupyter Notebooks.

One of the most important features of Azure ML is its publishing service where by a finished (trained) experiment can be exposed as a web API that can be consumed by any other applications, such as a website or mobile application etc. Each experiment can also be configured to be re-trained with new data using a separate API to maintain it.

In this short lab, we'll explore how Azure ML works by looking at a simple scenario, involving predicting the price of a car give a list of other attributes about the car. This will be a regression based experiment and we will see how built in modules alongside Python scripts and Jupyter Notebooks can be used to aid in the exploration and evaluation of the data.

In future labs, we'll see how to publish the model from within the studio to create an API for your model. Then review the sample code for a published experiment and test the output from an Excel add-in/application.

The Problem Domain

During this lab, we will create and evaluate two models, that given past data collected about cars and their values, will try to predict the price/value of a car given other attributes associated with the car:

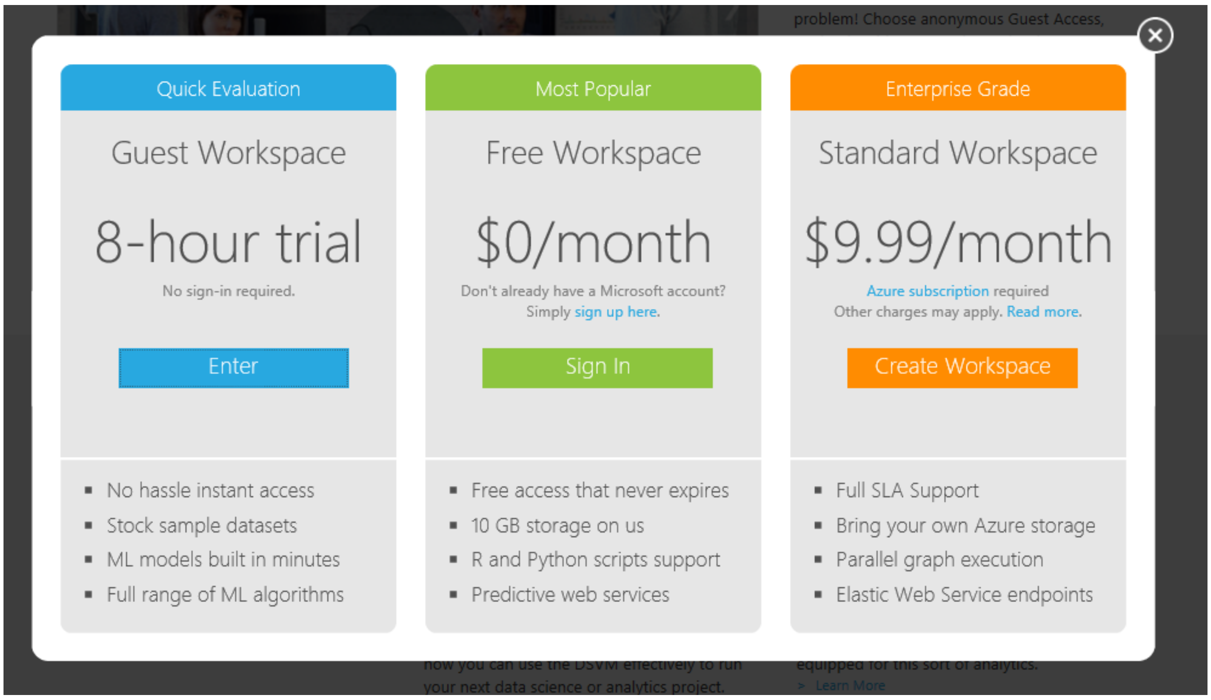
- The attribute columns in the dataset include values such as the model/make, fuel type and body style as well as performance values such as MPG, horsepower and engine type
- The value we are trying to predict is the price of the car. In this dataset, the values range from £5,000 to £45,000.

We will retrieve data from an Azure Blob Storage account and start to pre-process the dataset ready to train a machine learning model. The model we'll create is a form of supervised learning so we will use historical car attributes and values to predict the price of future cars we might receive. This model will perform a regression algorithm to try and predict the actual price of the car with the lowest amount of error, for example £16,595. This information is in the sample data in the 'price' column.

Enter Workspace

In the previous lab, you should have setup your **Azure Subscription**. Now visit the link: https://studio.azureml.net/ and choose **'Sign Up'**

Select the Free Workspace Option and sign in using your Microsoft Azure credentials

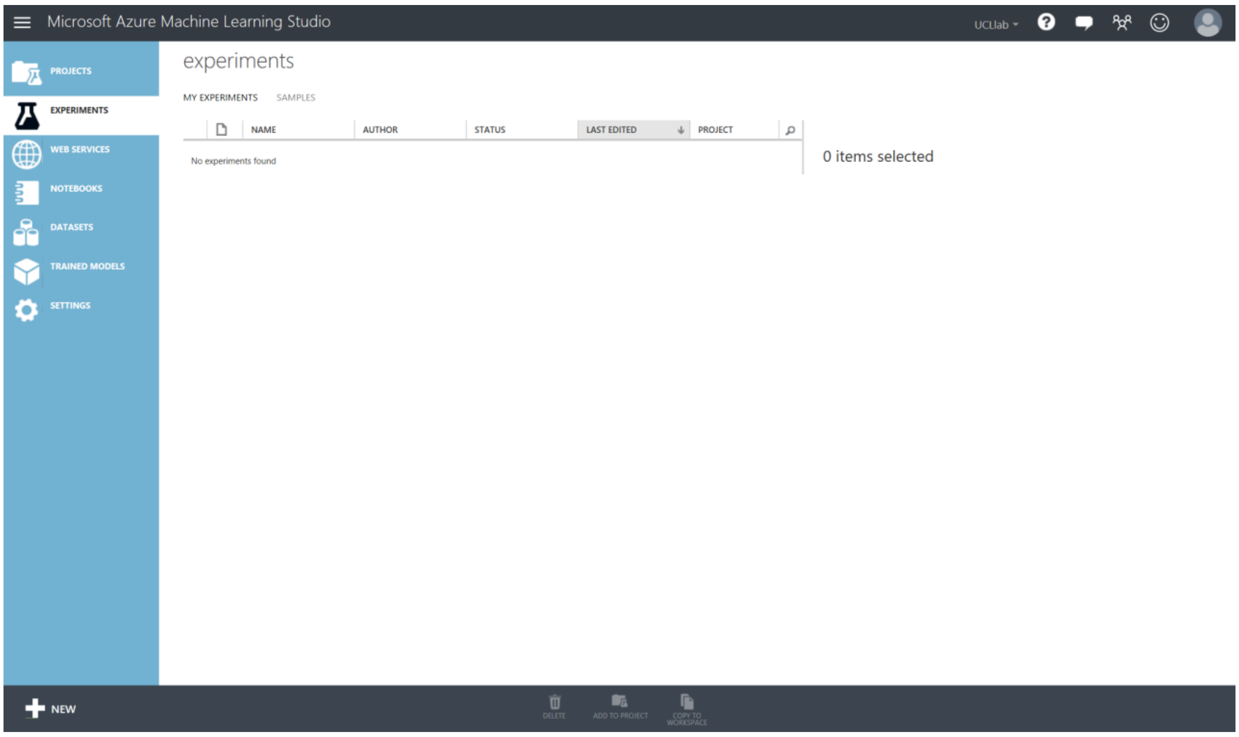


Quick Evaluation	Most Popular	Enterprise Grade
Guest Workspace	Free Workspace	Standard Workspace
8-hour trial	$0/month	$9.99/month
No sign-in required.	Don't already have a Microsoft account? Simply sign up here.	Azure subscription required. Other charges may apply. Read more.
Enter	Sign In	Create Workspace
▪ No hassle instant access	▪ Free access that never expires	▪ Full SLA Support
▪ Stock sample datasets	▪ 10 GB storage on us	▪ Bring your own Azure storage
▪ ML models built in minutes	▪ R and Python scripts support	▪ Parallel graph execution
▪ Full range of ML algorithms	▪ Predictive web services	▪ Elastic Web Service endpoints

This will take you to the Azure Machine Learning Studio, by default you will end in the Experiments tab on the left but there are other tabs available and described below:

- **Projects:** Like a file-explorer, a place to keep all associated resources together for a similar project. You may look at creating a project called 'AzureMLStudioLab' to keep all the experiments, notebooks and datasets you create today in one place that's well organised.
- **Experiments:** As you start creating machine learning experiments/models you will build up a list of different experiments that can be add, edited and deleted by an owner of the workspace

- Web Services: Once you have created a model and are ready to make it a production-ready API, this is the section you will find all your web services listed under
- Notebooks: You may have already explored notebooks. Although the Studio provides an easy to use, yet powerful, drag-drop style of creating experiments, you sometimes need a good old "REPL" to have a tight loop where you enter some script code and get a response. This functionality is integrated into ML Studio through Jupyter Notebooks
- Datasets: A place to store datasets/folders of scripts or functions (CSV, TSV,.zip) that you upload from your local machine
- Trained Models: Each time you create a model and start scoring against it to create a web service, this will create a 'Trained Model' module that will be reusable. This will be your chosen algorithm, trained on your chosen data in order to answer a machine learning question.
- Settings: The settings tab allows you to add descriptions to your workspace as well as add new users so you can work in a more collaborative enviroment when designing your machine learning algorithms



No experiments found

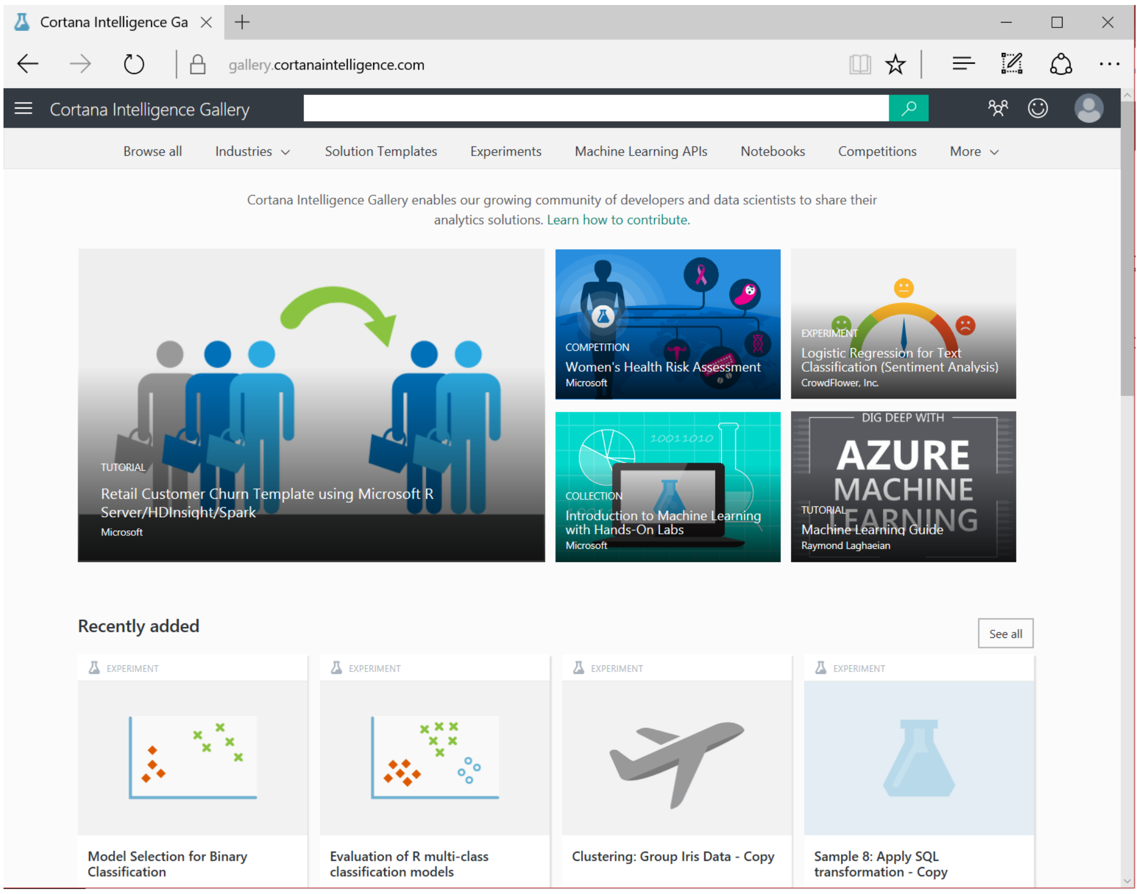
0 items selected

Getting Started

Now we can start building our experiment. Typically, we would get hold of a data set and either connect to its source or upload a file:

- Azure ML accepts many formats including CSV, TSV, Plain Text and Zip files via upload
- Alternatively, 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.

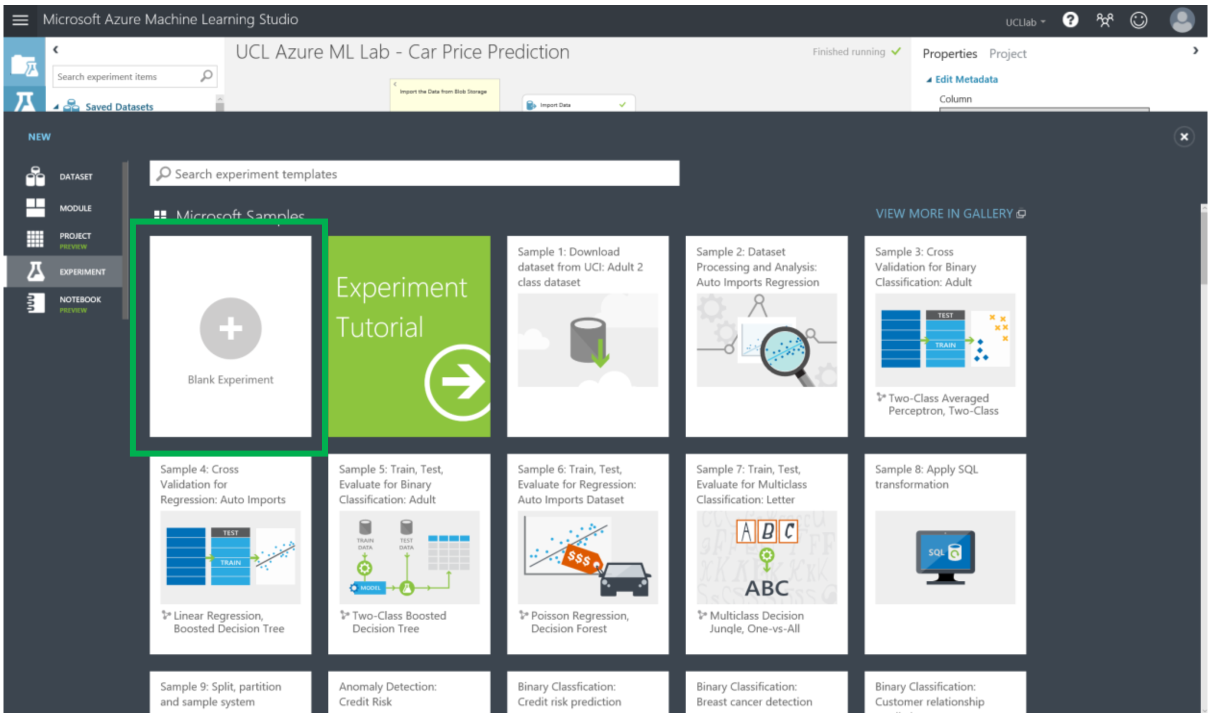
[EXTRA INFORMATION:] It's also possible to adapt an existing experiment to your needs from one of the many samples in the [Cortana Intelligence Gallery](https://gallery.cortanaintelligence.com) – you can change it to use your own data and parameters and quickly see how suitable it is for your needs:


Cortana Intelligence Ga
gallery.cortanaintelligence.com
Cortana Intelligence Gallery
Browse all
Industries
Solution Templates
Experiments
Machine Learning APIs
Notebooks
Competitions
More
TUTORIAL
Retail Customer Churn Template using Microsoft R Server/HDInsight/Spark
Microsoft
COMPETITION
Women's Health Risk Assessment
Microsoft
EXPERIMENT
Logistic Regression for Text Classification (Sentiment Analysis)
CrowdFlower, Inc.
COLLECTION
Introduction to Machine Learning with Hands-On Labs
Microsoft
TUTORIAL
Machine Learning Guide
Raymond Laghaeian
Recently added
See all
EXPERIMENT
Model Selection for Binary Classification
EXPERIMENT
Evaluation of R multi-class classification models
EXPERIMENT
Clustering: Group Iris Data - Copy
EXPERIMENT
Sample 8: Apply SQL transformation - Copy

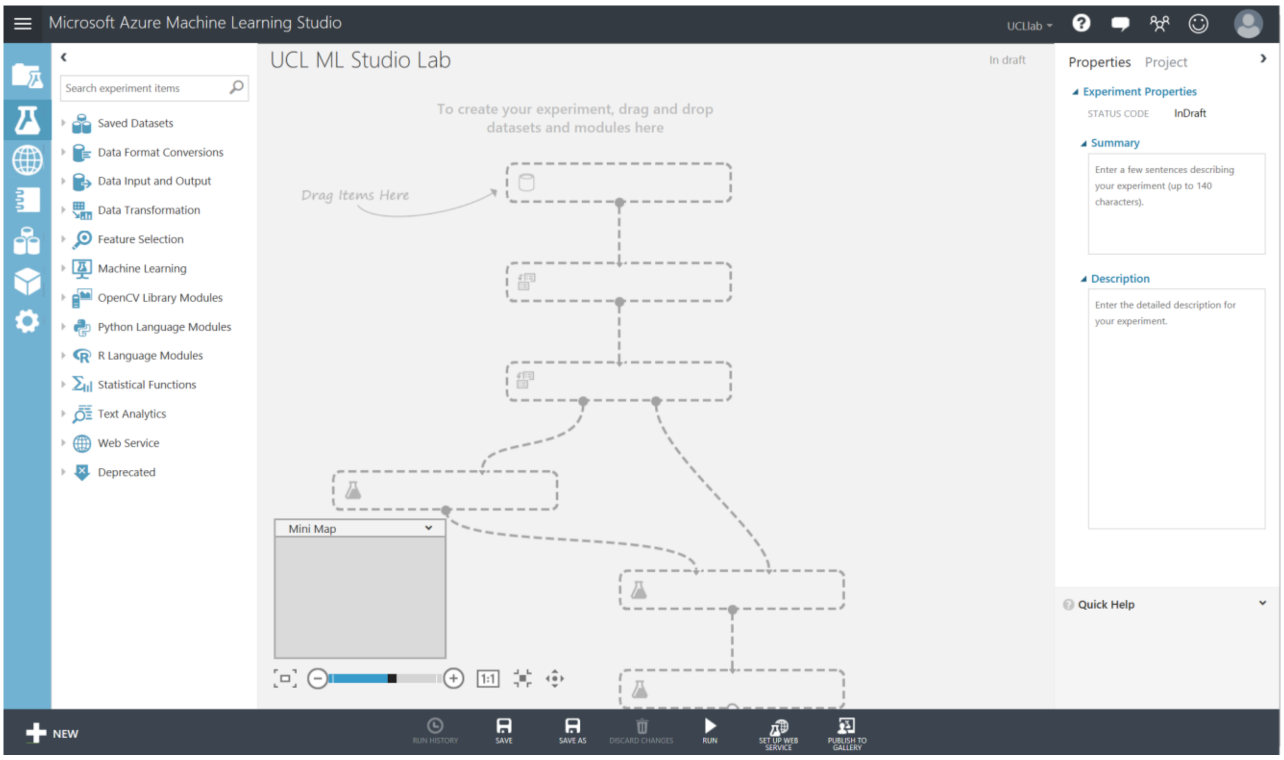
Cortana Intelligence Gallery enables our growing community of developers and data scientists to share their analytics solutions. Learn how to contribute.

For this lab, we will be creating a new experiment and using the 'Import Data' module 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'


Microsoft Azure Machine Learning Studio
UCL Azure ML Lab - Car Price Prediction
NEW
DATASET
MODULE
PROJECT
EXPERIMENT
NOTEBOOK
Search experiment templates
Microsoft Samples
VIEW MORE IN GALLERY
Blank Experiment
Experiment Tutorial
Sample 1: Download dataset from UCI: Adult 2 class dataset
Sample 2: Dataset Processing and Analysis: Auto Imports Regression
Sample 3: Cross Validation for Binary Classification: Adult
Sample 4: Cross Validation for Regression: Auto Imports
Sample 5: Train, Test, Evaluate for Binary Classification: Adult
Sample 6: Train, Test, Evaluate for Regression: Auto Imports Dataset
Sample 7: Train, Test, Evaluate for Multiclass Classification: Letter
Sample 8: Apply SQL transformation
Sample 9: Split, partition and sample system
Anomaly Detection: Credit Risk
Binary Classfication: Credit risk prediction
Binary Classification: Breast cancer detection
Binary Classification: Customer relationship

- Next change the title of the experiment to something descriptive, for example: UCL ML Studio Lab

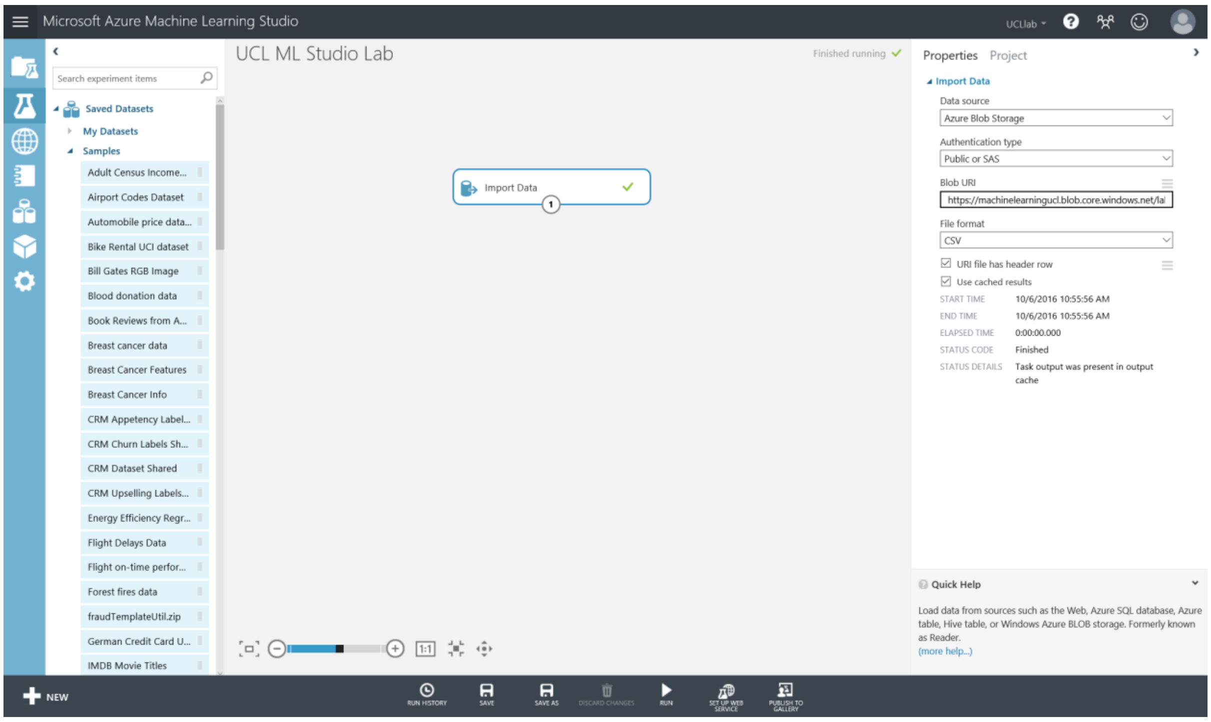


On the left in blue with white icons are the different objects we use in ML studio such as Projects, Experiments, Web services, Jupyter Notebooks, Data Sets, Trained Models and Settings. Next on the left is a nested list of all the modules we can drag on to the design surface in the middle of the screen. On the right is the properties pane, this will change dynamically for each module or properties for the whole experiment if no module is selected. There are also functions we can perform in the bottom dark grey toolbar such as Save, Run and Setup Web Service.

From the list of nested 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://machinelearningucl.blob.core.windows.net/lab/carPriceData.csv
- File Format: CSV
- File Has Header Row: CHECKED
- Use Cached Results: CHECKED

Once these 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, like below



Properties Project

Import Data

Data source: Azure Blob Storage

Authentication type: Public or SAS

Blob URI: https://machinelearningucl.blob.core.windows.net/lal

File format: CSV

☑ URI file has header row

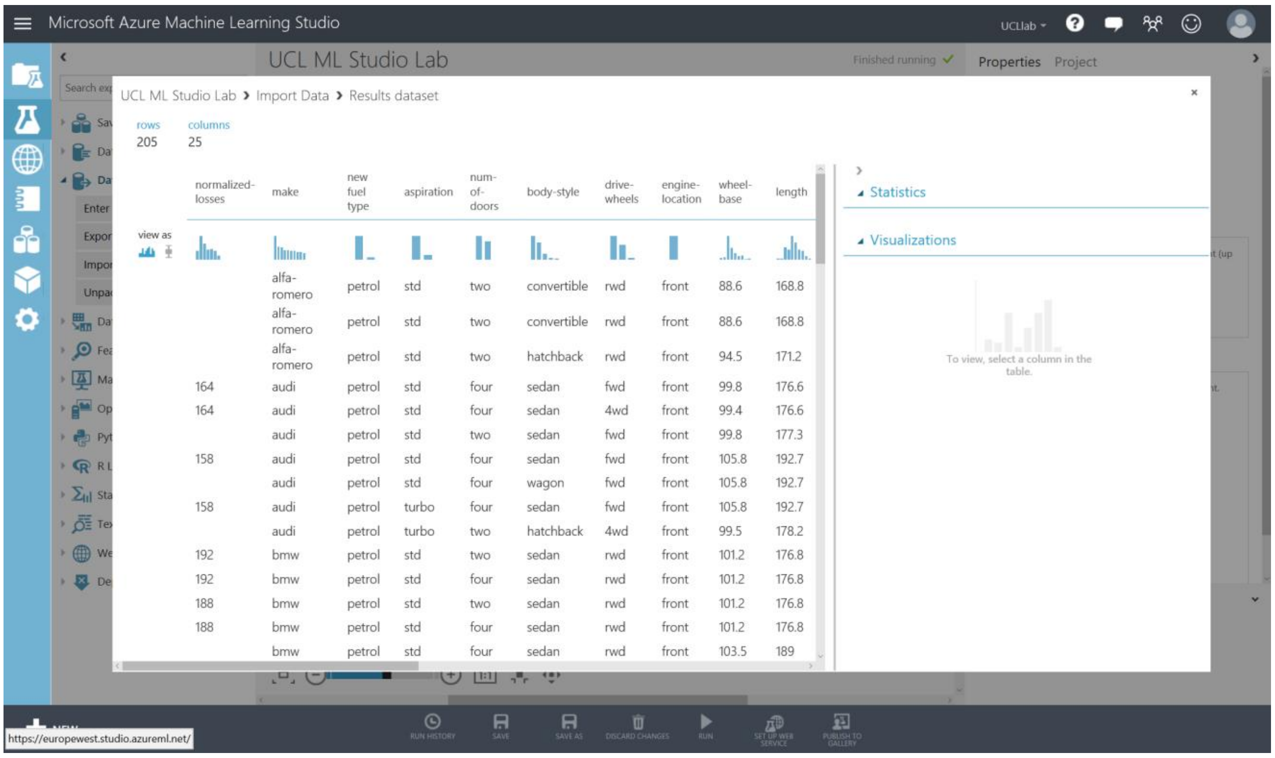
☑ Use cached results

START TIME	10/6/2016 10:55:56 AM
END TIME	10/6/2016 10:55:56 AM
ELAPSED TIME	0:00:00.000
STATUS CODE	Finished
STATUS DETAILS	Task output was present in output cache

Quick Help

Load data from sources such as the Web, Azure SQL database, Azure table, Hive table, or Windows Azure BLOB storage. Formerly known as Reader.
(more help...)

Now let's look at the data. Hover over the output port of the Import Data Module. As you hover over it, it will say 'Results Dataset', right click and choose 'Visualise' to see a snapshot of the dataset and some summary information



UCL ML Studio Lab › Import Data › Results dataset

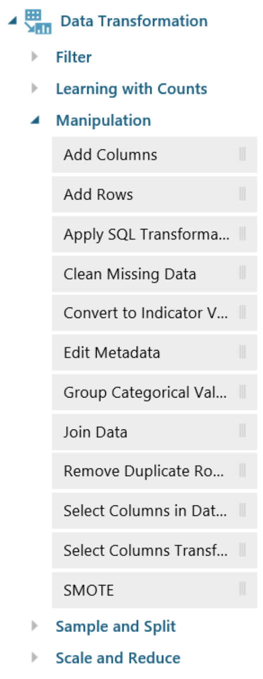
rows 205
columns 25

normalized-losses	make	new fuel type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length
	alfa-romero	petrol	std	two	convertible	rwd	front	88.6	168.8
	alfa-romero	petrol	std	two	convertible	rwd	front	88.6	168.8
	alfa-romero	petrol	std	two	hatchback	rwd	front	94.5	171.2
164	audi	petrol	std	four	sedan	fwd	front	99.8	176.6
164	audi	petrol	std	four	sedan	4wd	front	99.4	176.6
	audi	petrol	std	two	sedan	fwd	front	99.8	177.3
158	audi	petrol	std	four	sedan	fwd	front	105.8	192.7
	audi	petrol	std	four	wagon	fwd	front	105.8	192.7
158	audi	petrol	turbo	four	sedan	fwd	front	105.8	192.7
	audi	petrol	turbo	two	hatchback	4wd	front	99.5	178.2
192	bmw	petrol	std	two	sedan	rwd	front	101.2	176.8
192	bmw	petrol	std	four	sedan	rwd	front	101.2	176.8
188	bmw	petrol	std	two	sedan	rwd	front	101.2	176.8
188	bmw	petrol	std	four	sedan	rwd	front	101.2	176.8
	bmw	petrol	std	four	sedan	rwd	front	103.5	189

The 'Visualise Data' option is extremely useful as you work through building your experiment. I encourage you to visualise the dataset after each module change to see how the transformation or function affected your original data. 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
- Note as you scroll down in this window you cannot scroll through the whole dataset, this is only showing the head of the dataset

Data Pre-Processing


Data Transformation
Filter
Learning with Counts
Manipulation
Add Columns
Add Rows
Apply SQL Transforma...
Clean Missing Data
Convert to Indicator V...
Edit Metadata
Group Categorical Val...
Join Data
Remove Duplicate Ro...
Select Columns in Dat...
Select Columns Transf...
SMOTE
Sample and Split
Scale and Reduce

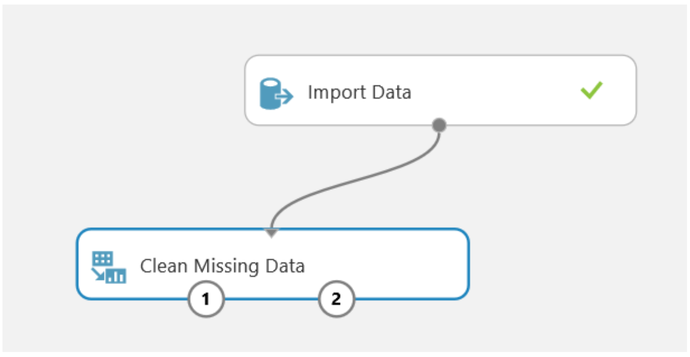
In this section, we will get the dataset into a form that is suitable to train a machine learning model.

The major data preparation tasks include data cleaning, integration, transformation, reduction, and discretization or quantization.

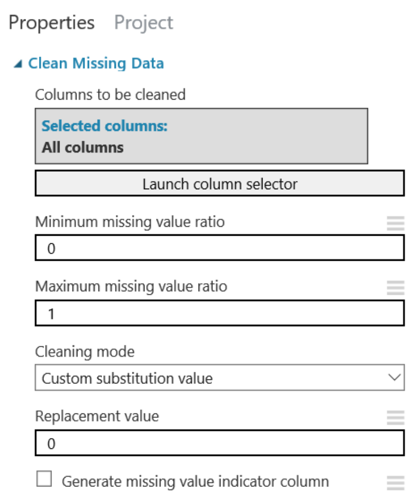
In Azure ML studio, you can find modules to perform all these operations and other data pre-processing tasks in the Data Transformation group in the left panel, we will look at a couple of built in modules, listed to the left. We will also see who we can integrate Jupyter Notebooks and Python code to help use cleanse and understand our data.

Let's start by cleaning some of 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. Then 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



Clean Missing Data

Columns to be cleaned

Selected columns:
All columns

Minimum missing value ratio: 0

Maximum missing value ratio: 1

Cleaning mode: Custom substitution value

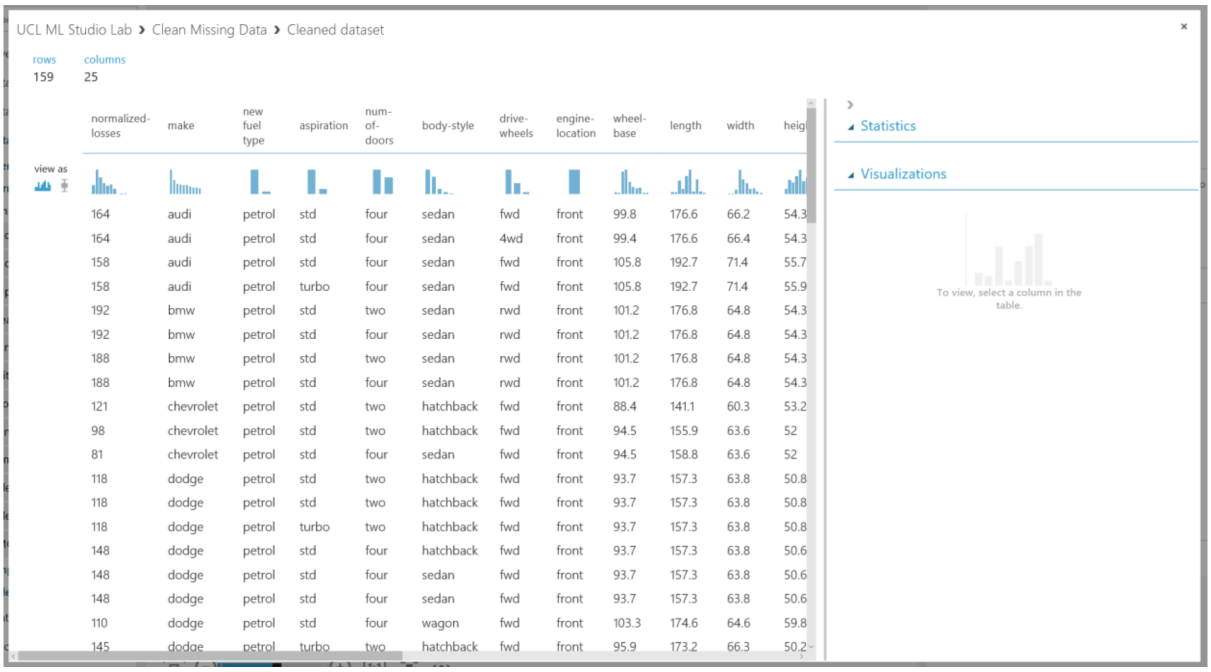
Replacement value: 0

☐ Generate missing value indicator column

Change the cleaning mode property to 'Remove entire row'. This will perform a transformation where all columns in the dataset are scanned for missing values, if found, it will remove the row of data. The output of this transformation will be a slightly smaller dataset; however, the dataset records will be full, descriptive data for the machine learning algorithm to learn from.

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.

Note the number of rows in the data has decreased to 159 and the normalised losses columns now doesn't have sparse gaps.



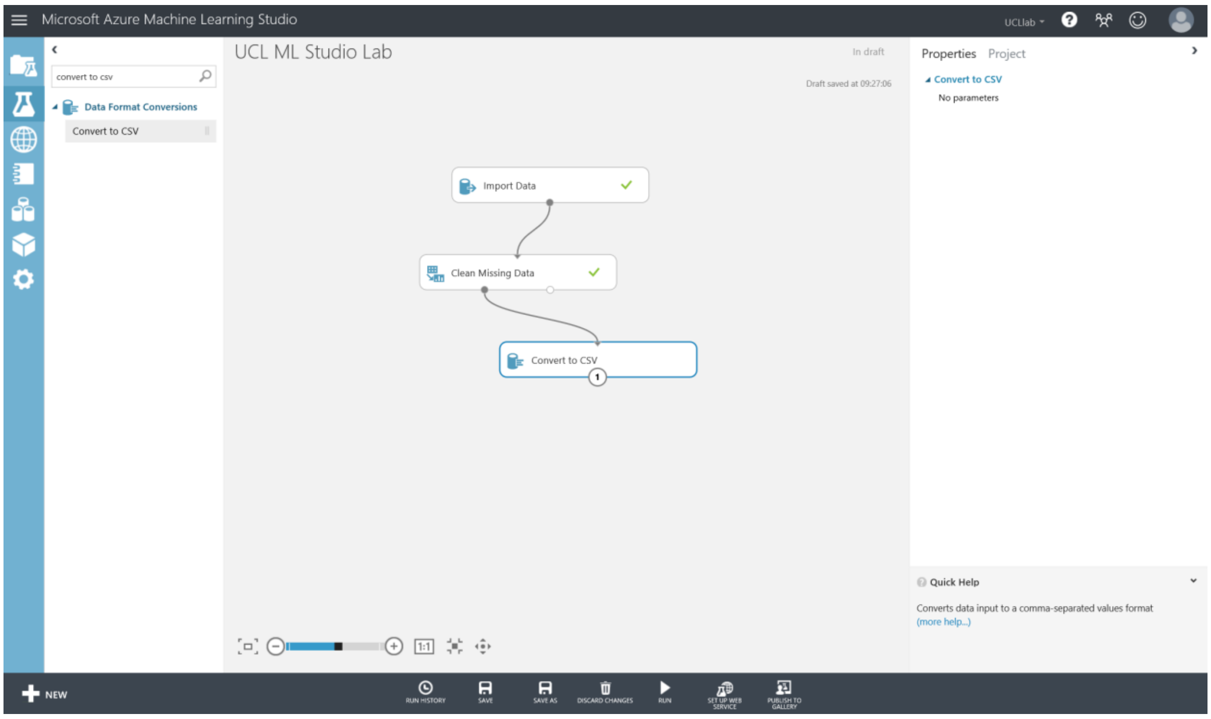
UCL ML Studio Lab › Clean Missing Data › Cleaned dataset

rows 159 columns 25

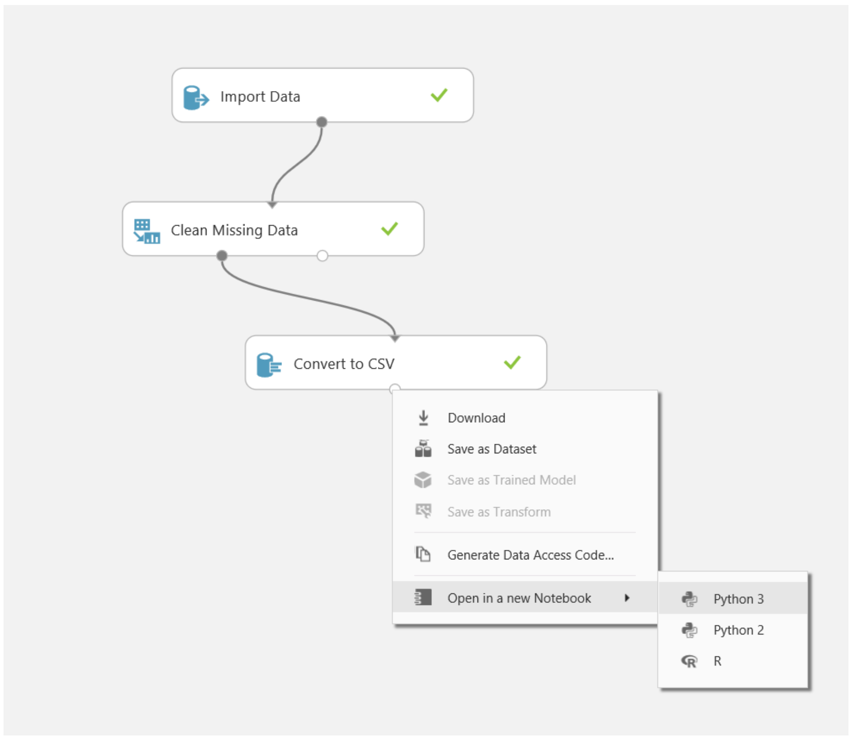
normalized-losses	make	new fuel type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	heig
164	audi	petrol	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3
164	audi	petrol	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3
158	audi	petrol	std	four	sedan	fwd	front	105.8	192.7	71.4	55.7
158	audi	petrol	turbo	four	sedan	fwd	front	105.8	192.7	71.4	55.9
192	bmw	petrol	std	two	sedan	rwd	front	101.2	176.8	64.8	54.3
192	bmw	petrol	std	four	sedan	rwd	front	101.2	176.8	64.8	54.3
188	bmw	petrol	std	two	sedan	rwd	front	101.2	176.8	64.8	54.3
188	bmw	petrol	std	four	sedan	rwd	front	101.2	176.8	64.8	54.3
121	chevrolet	petrol	std	two	hatchback	fwd	front	88.4	141.1	60.3	53.2
98	chevrolet	petrol	std	two	hatchback	fwd	front	94.5	155.9	63.6	52
81	chevrolet	petrol	std	four	sedan	fwd	front	94.5	158.8	63.6	52
118	dodge	petrol	std	two	hatchback	fwd	front	93.7	157.3	63.8	50.8
118	dodge	petrol	std	two	hatchback	fwd	front	93.7	157.3	63.8	50.8
118	dodge	petrol	turbo	two	hatchback	fwd	front	93.7	157.3	63.8	50.8
148	dodge	petrol	std	four	hatchback	fwd	front	93.7	157.3	63.8	50.6
148	dodge	petrol	std	four	sedan	fwd	front	93.7	157.3	63.8	50.6
148	dodge	petrol	std	four	sedan	fwd	front	93.7	157.3	63.8	50.6
110	dodge	petrol	std	four	wagon	fwd	front	103.3	174.6	64.6	59.8
145	dodge	petrol	turbo	two	hatchback	fwd	front	95.9	173.2	66.3	50.2

Next we will see how to interact with Jupyter Notebooks in Azure ML from the studio space. This allows us more flexibility to explore the dataset using Python.

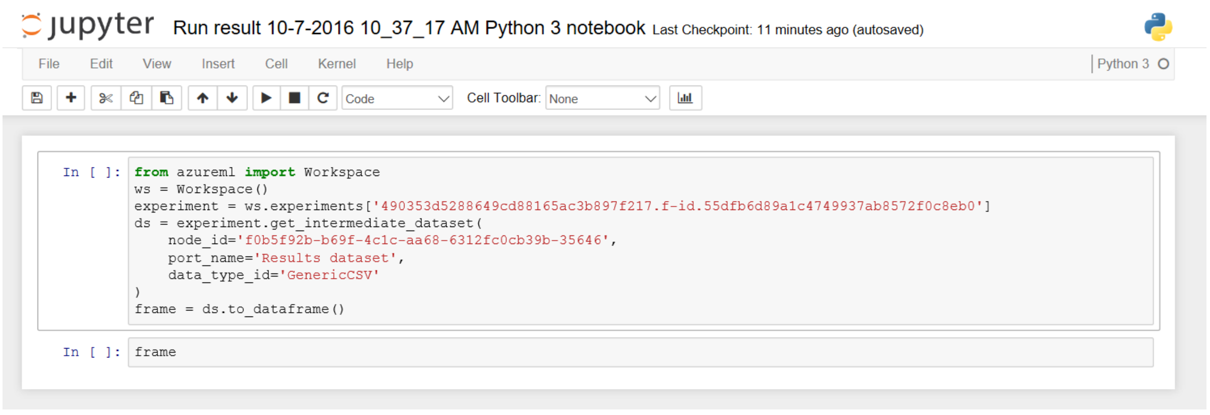
Connect the left output port of the 'Clean Missing Data' module to a 'Convert to CSV' module. Try and use the search box in the top left of the screen to find this module and connect it up, so it looks like the screenshot below. Once connected, run the experiment again.



To access the Jupyter Notebooks functionality from inside the Azure ML Studio, you must first convert the dataset into a CSV, as we have done above. Now, if you right click on the output port of the 'Convert to CSV' file and select 'Open in a new notebook' and 'Python 3'.



This will then take you to a new tab in the browser, which you might recognise from previous lab sessions, Jupyter Notebooks.

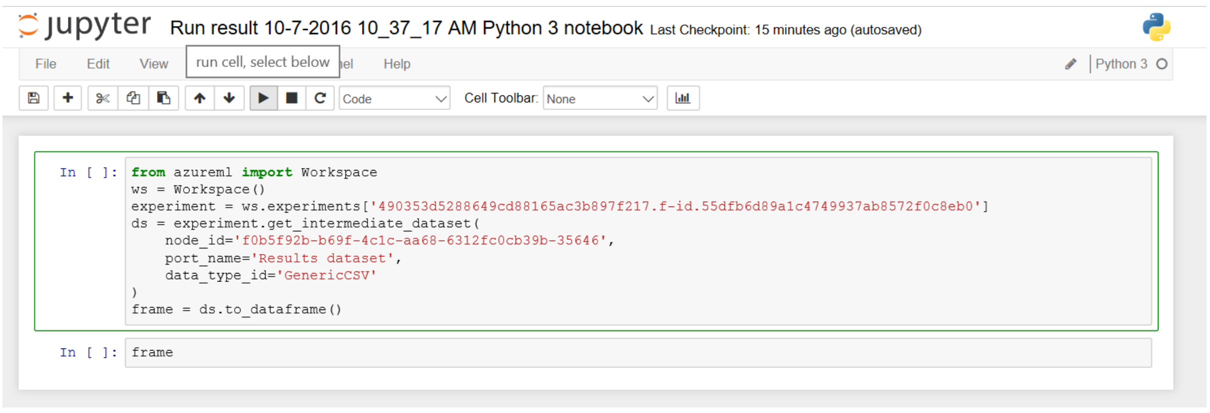

jupyter
Run result 10-7-2016 10_37_17 AM Python 3 notebook Last Checkpoint: 11 minutes ago (autosaved)
File Edit View Insert Cell Kernel Help
Python 3
Code
Cell Toolbar: None

```
In [ ]: from azureml import Workspace
        ws = Workspace()
        experiment = ws.experiments['490353d5288649cd88165ac3b897f217.f-id.55dfb6d89a1c4749937ab8572f0c8eb0']
        ds = experiment.get_intermediate_dataset(
            node_id='f0b5f92b-b69f-4c1c-aa68-6312fc0cb39b-35646',
            port_name='Results dataset',
            data_type_id='GenericCSV'
        )
        frame = ds.to_dataframe()

In [ ]: frame
```

One thing to note, is the use of the 'azureml' package in the code snippet above that imports the dataset from the workspace/experiment we were working in, and the node ID (convert to csv) that the dataset is coming from. Now you can interact with the dataset from the car prediction experiment using Python.

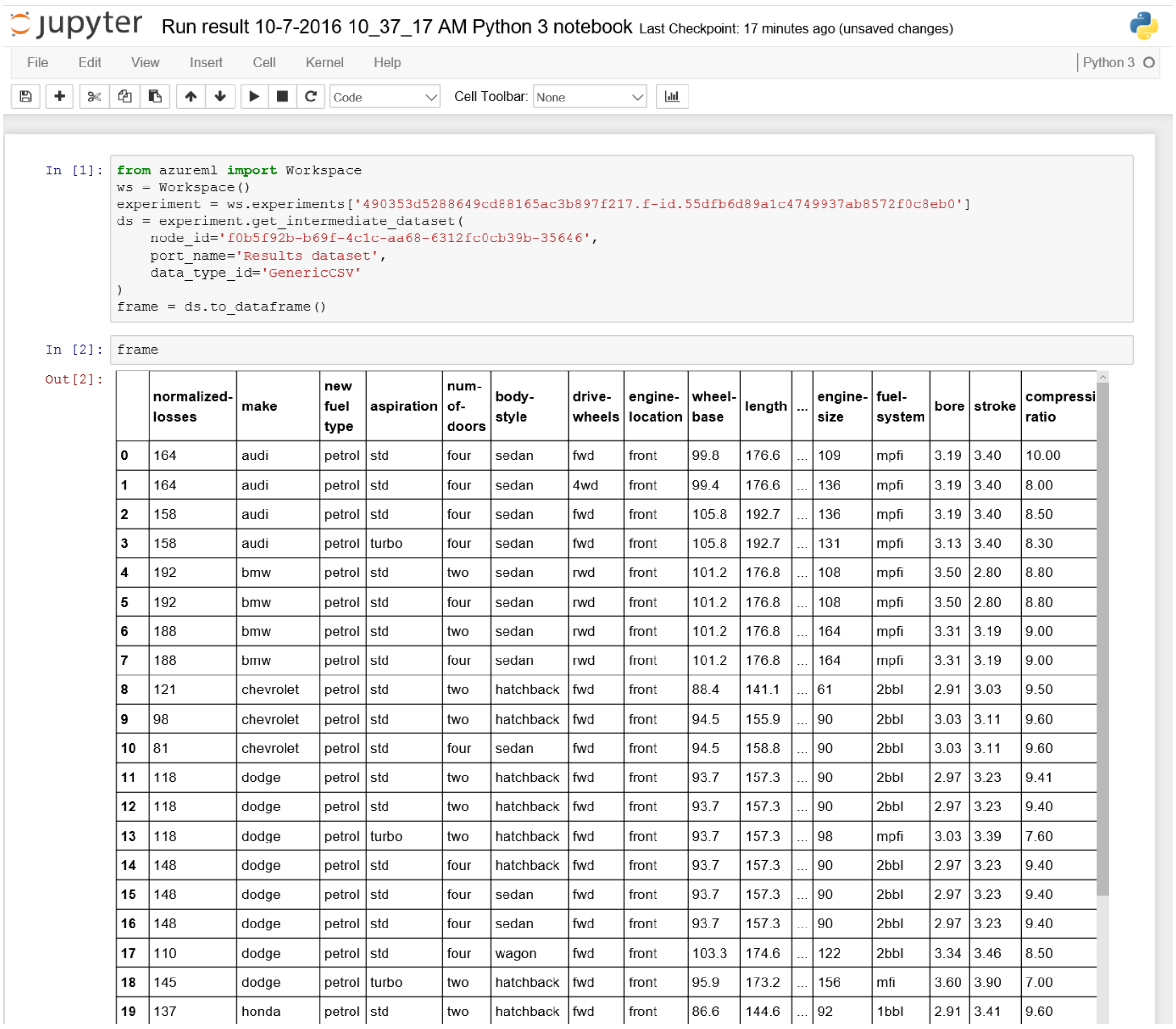
In the next step, we will change a couple of columns of data in the data frame using simple python statements. This task is to get you used to the notebooks space and not to challenge your python skills.

Choose the first block of interactive code and hit Run from the toolbar above. This will pull in the dataset from the experiment into a variable call 'frame'.


jupyter
Run result 10-7-2016 10_37_17 AM Python 3 notebook Last Checkpoint: 15 minutes ago (autosaved)
File Edit View run cell, select below Help
Python 3
Code
Cell Toolbar: None

```
In [ ]: from azureml import Workspace
        ws = Workspace()
        experiment = ws.experiments['490353d5288649cd88165ac3b897f217.f-id.55dfb6d89a1c4749937ab8572f0c8eb0']
        ds = experiment.get_intermediate_dataset(
            node_id='f0b5f92b-b69f-4c1c-aa68-6312fc0cb39b-35646',
            port_name='Results dataset',
            data_type_id='GenericCSV'
        )
        frame = ds.to_dataframe()

In [ ]: frame
```

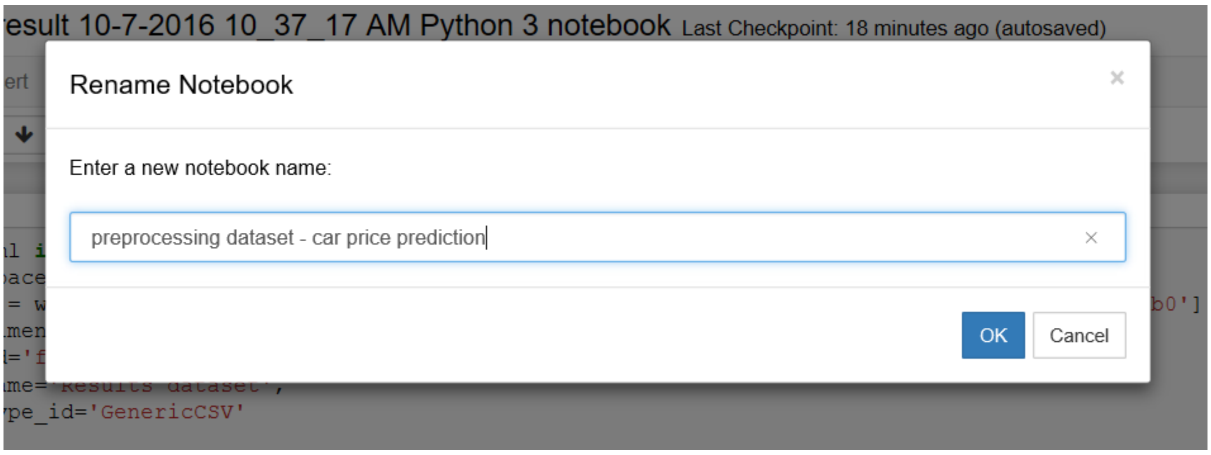
Once complete then run the second interactive code block to see the dataset displayed.


jupyter Run result 10-7-2016 10_37_17 AM Python 3 notebook Last Checkpoint: 17 minutes ago (unsaved changes)
File
Edit
View
Insert
Cell
Kernel
Help
Python 3
Code
Cell Toolbar: None

```
In [1]: from azureml import Workspace
        ws = Workspace()
        experiment = ws.experiments['490353d5288649cd88165ac3b897f217.f-id.55dfb6d89a1c4749937ab8572f0c8eb0']
        ds = experiment.get_intermediate_dataset(
            node_id='f0b5f92b-b69f-4c1c-aa68-6312fc0cb39b-35646',
            port_name='Results dataset',
            data_type_id='GenericCSV'
        )
        frame = ds.to_dataframe()

In [2]: frame

Out[2]:
```

	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

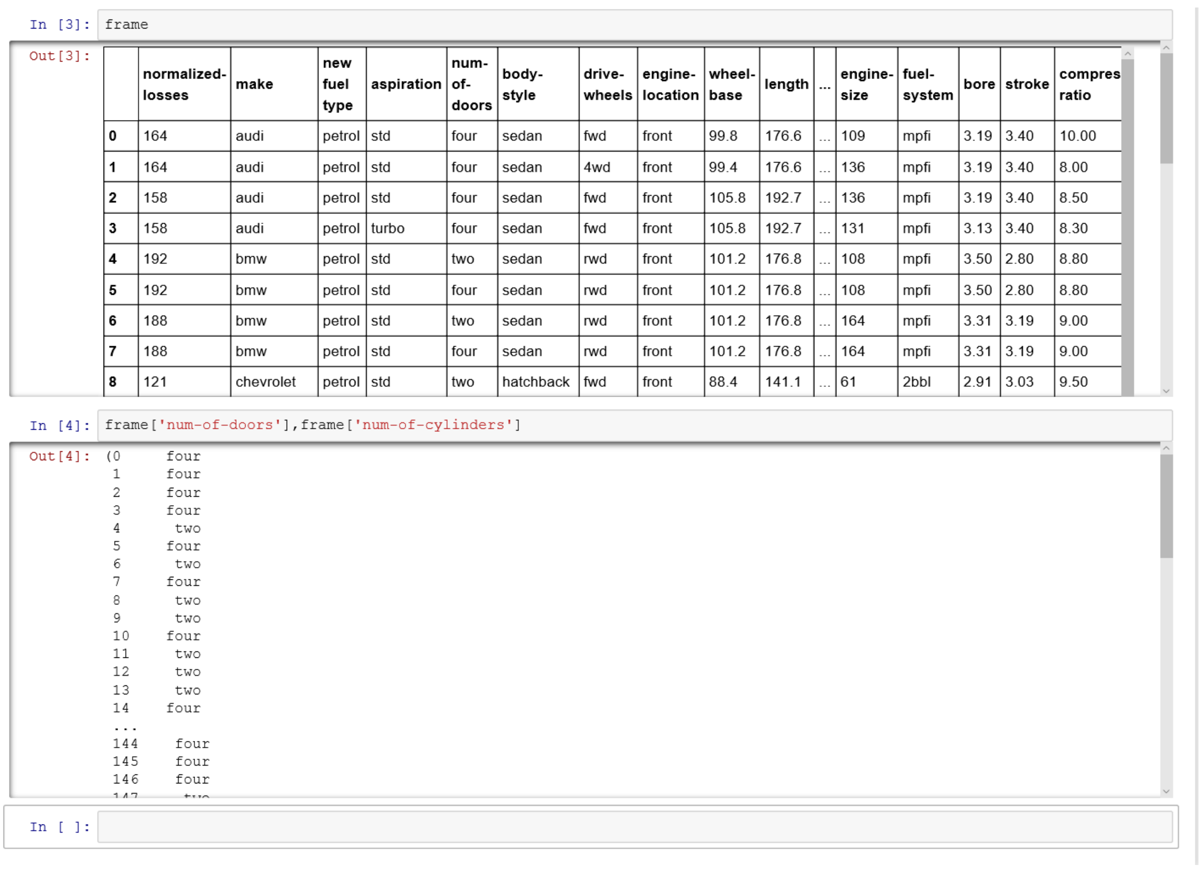
Before we interact further with the dataset, lets rename the notebook to something we recognise. Choose File -> Rename … and choose a filename such as: “preprocessing dataset – car price prediction” and click OK.


esult 10-7-2016 10_37_17 AM Python 3 notebook Last Checkpoint: 18 minutes ago (autosaved)
Rename Notebook
Enter a new notebook name:
preprocessing dataset - car price prediction
OK
Cancel

In this dataset there are a couple of columns, such as 'num-of-doors' and 'num-of-cylinders' that are string values such as "two" or "four", lets change their values to integer types and feed back into the experiment the process to do that.

Lets have a look at the two columns mentioned above by creating a new code cell and running:

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



```
In [3]: frame
```

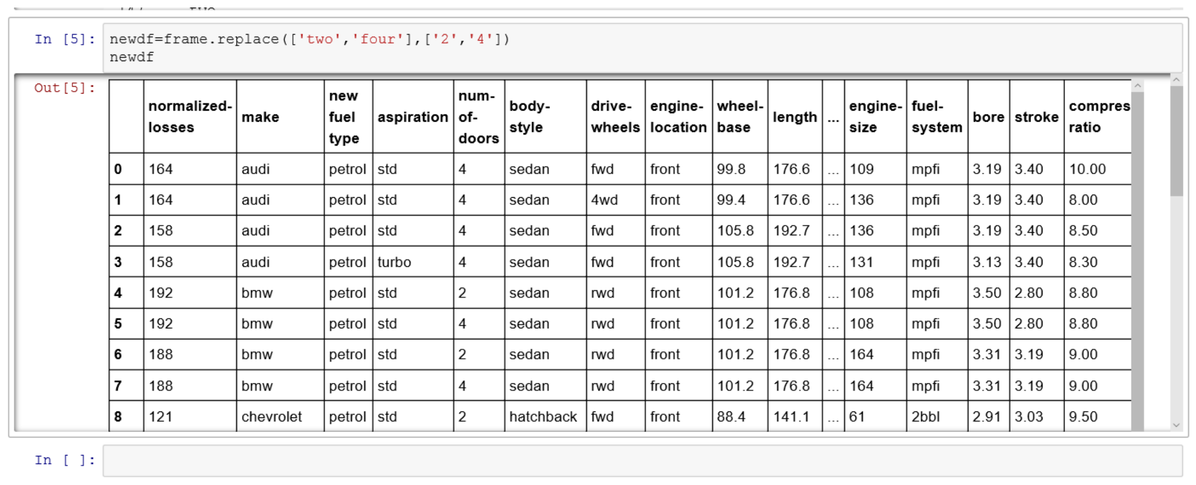
Out[3]:

	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 [ ]:
```

Now let's take those values in the data frame and swap them for numeric values. We will start with the number of doors example. Create a new code cell and run:

```
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 [ ]:
```

This shows the num-of-doors column now in integer values however the num-of-cylinder column has other values. To change them and check the columns are correct run the piece of python code below:

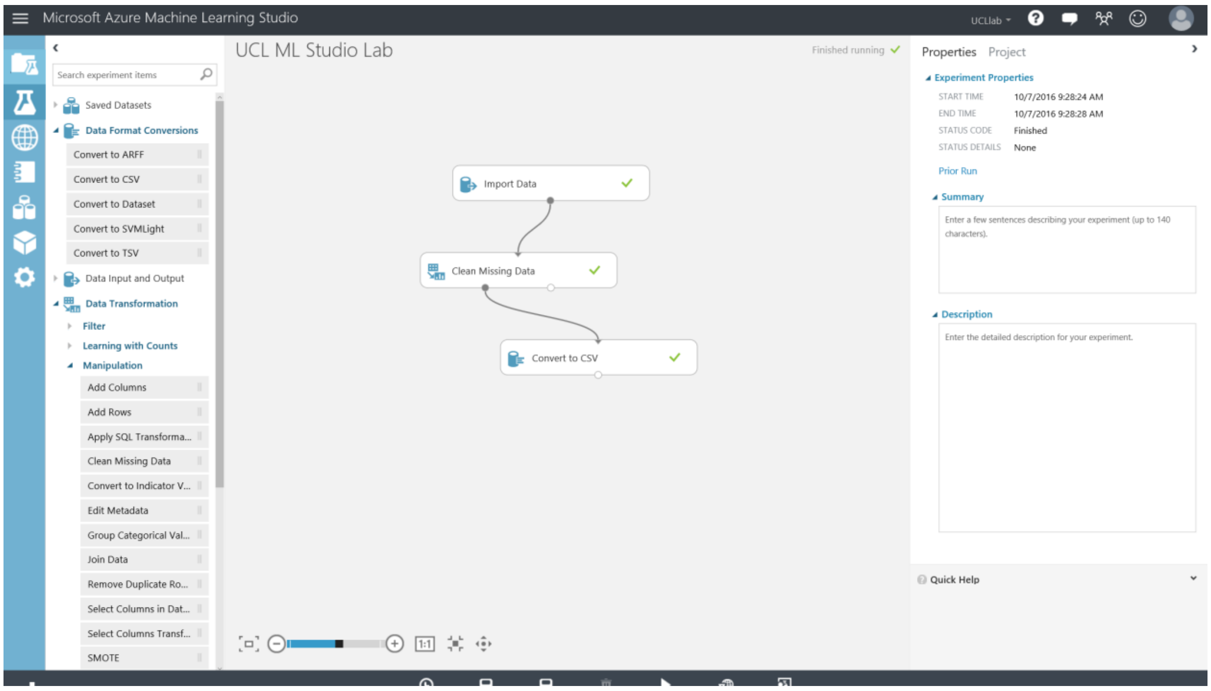
```
newdf =
frame.replace(['four','six','five','eight','two','three','twelve'],[
'4','6','5','8','2','3','12'])

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

Now we have changed the data types and values in the dataset we want to be able to use this edited 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. If this is a more complex and involved function. You can tidy up the notebook to contain the function call and return a data frame. The save the notebook, by using File -> download as -> Python (.py) file and upload to execute python module in the .zip input port.

In the lab we will look into option 1, however an extend section at the bottom of this lab is provided to explore the opportunities with option 2.

Make sure you save the notebook and head back to the experiment tab in your browser:

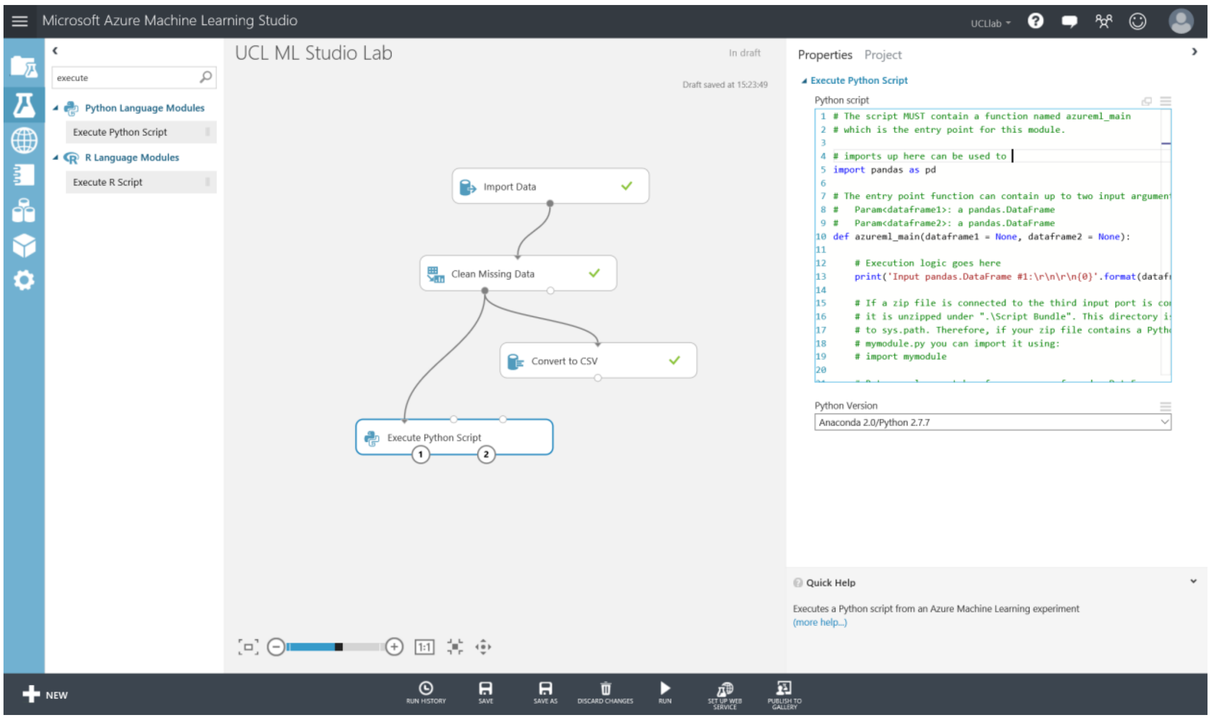


START TIME 10/7/2016 9:28:24 AM
END TIME 10/7/2016 9:28:28 AM
STATUS CODE Finished
STATUS DETAILS None

Enter a few sentences describing your experiment (up to 140 characters).

Enter the detailed description for your experiment.

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.



```
# 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

# The entry point function can contain up to two input argumen
#   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(dataf

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

```

Python Version: Anaconda 2.0/Python 2.7.7

Connect the Python module to the left output port of the 'Clean Missing Data' module, the last iteration of the dataset within the experiment space. Connect it to the first input port of the python module.

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 you would like to import or python functions you want to call in your script. This module also in output form allows to output the whole data frame to continue your experiment as well as a 'Python Device' which lets you see and plots and graphs you may have created in your python code.

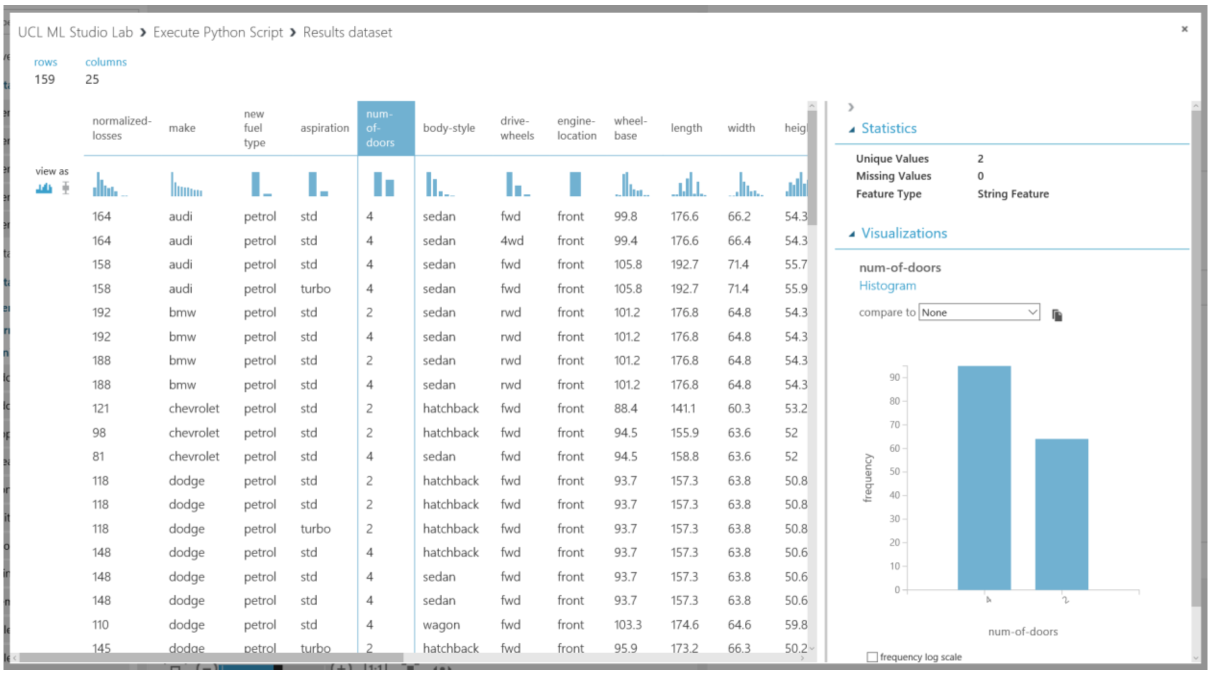
To find out more about this modules capability check out the documentation here:
https://azure.microsoft.com/en-gb/documentation/articles/machine-learning-execute-python-scripts/

This module gives you sample code structure where you can add your scripts inside the main function and return a data frame.

Read and copy the code below into the properties, 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,
```

Once this code is copied in, run the experiment from the bottom toolbar as we have done before. Once completed, right click and visualise the left output port of the 'Execute Python Script' module. You should see that 'num-of-doors' and 'num-of-cylinder' columns have been changed to digits instead of text.



UCL ML Studio Lab › Execute Python Script › Results dataset

rows	columns
159	25

normalized-losses	make	new fuel type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	heig
164	audi	petrol	std	4	sedan	fwd	front	99.8	176.6	66.2	54.3
164	audi	petrol	std	4	sedan	4wd	front	99.4	176.6	66.4	54.3
158	audi	petrol	std	4	sedan	fwd	front	105.8	192.7	71.4	55.7
158	audi	petrol	turbo	4	sedan	fwd	front	105.8	192.7	71.4	55.9
192	bmw	petrol	std	2	sedan	rwd	front	101.2	176.8	64.8	54.3
192	bmw	petrol	std	4	sedan	rwd	front	101.2	176.8	64.8	54.3
188	bmw	petrol	std	2	sedan	rwd	front	101.2	176.8	64.8	54.3
188	bmw	petrol	std	4	sedan	rwd	front	101.2	176.8	64.8	54.3
121	chevrolet	petrol	std	2	hatchback	fwd	front	88.4	141.1	60.3	53.2
98	chevrolet	petrol	std	2	hatchback	fwd	front	94.5	155.9	63.6	52
81	chevrolet	petrol	std	4	sedan	fwd	front	94.5	158.8	63.6	52
118	dodge	petrol	std	2	hatchback	fwd	front	93.7	157.3	63.8	50.8
118	dodge	petrol	std	2	hatchback	fwd	front	93.7	157.3	63.8	50.8
118	dodge	petrol	turbo	2	hatchback	fwd	front	93.7	157.3	63.8	50.8
148	dodge	petrol	std	4	hatchback	fwd	front	93.7	157.3	63.8	50.6
148	dodge	petrol	std	4	sedan	fwd	front	93.7	157.3	63.8	50.6
148	dodge	petrol	std	4	sedan	fwd	front	93.7	157.3	63.8	50.6
110	dodge	petrol	std	4	wagon	fwd	front	103.3	174.6	64.6	59.8
145	dodge	petrol	turbo	2	hatchback	fwd	front	95.9	173.2	66.3	50.2

Statistics

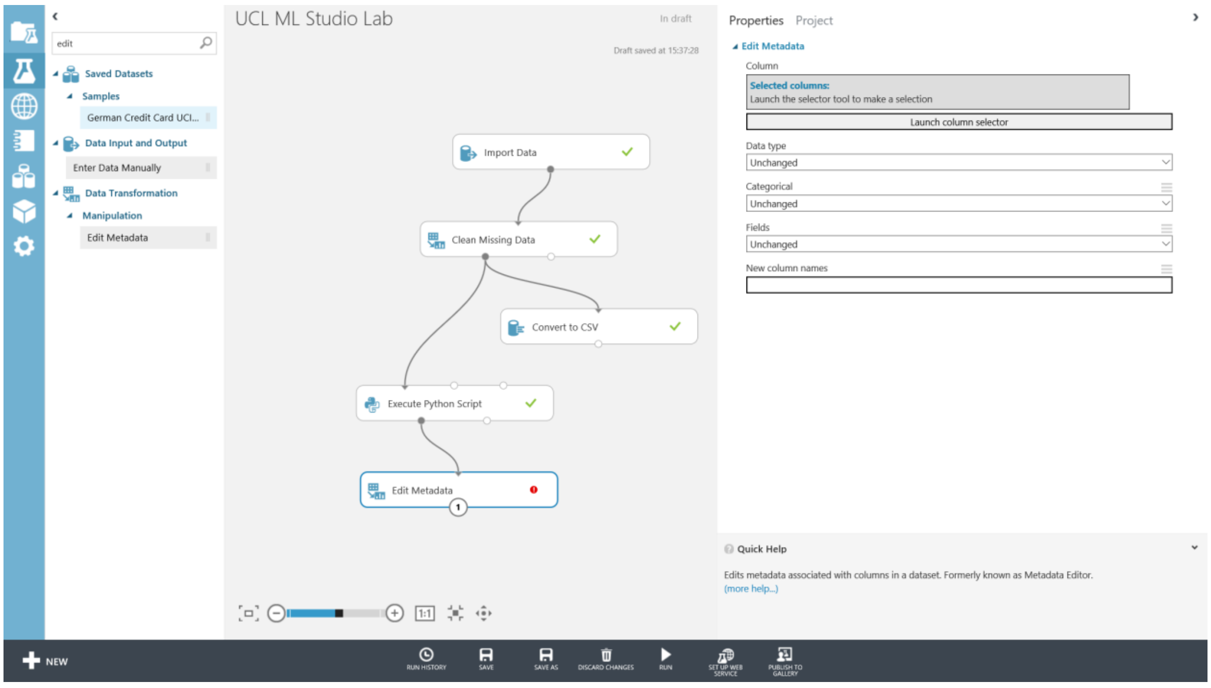
Unique Values	2
Missing Values	0
Feature Type	String Feature

Visualizations

num-of-doors
Histogram
compare to None

However over on the right of the visualise data pane, the feature type is still a string feature. We need to make sure the studio experiment understands the true data type of this column.

To do that we can use a 'Edit Metadata' module, find that module under Data Transformation, Manipulation and click and drag it onto the experiment surface. Connect the left output port (dataset) of the 'Execute Python Script' module to the input port of the 'Edit Metadata' module



Properties Project

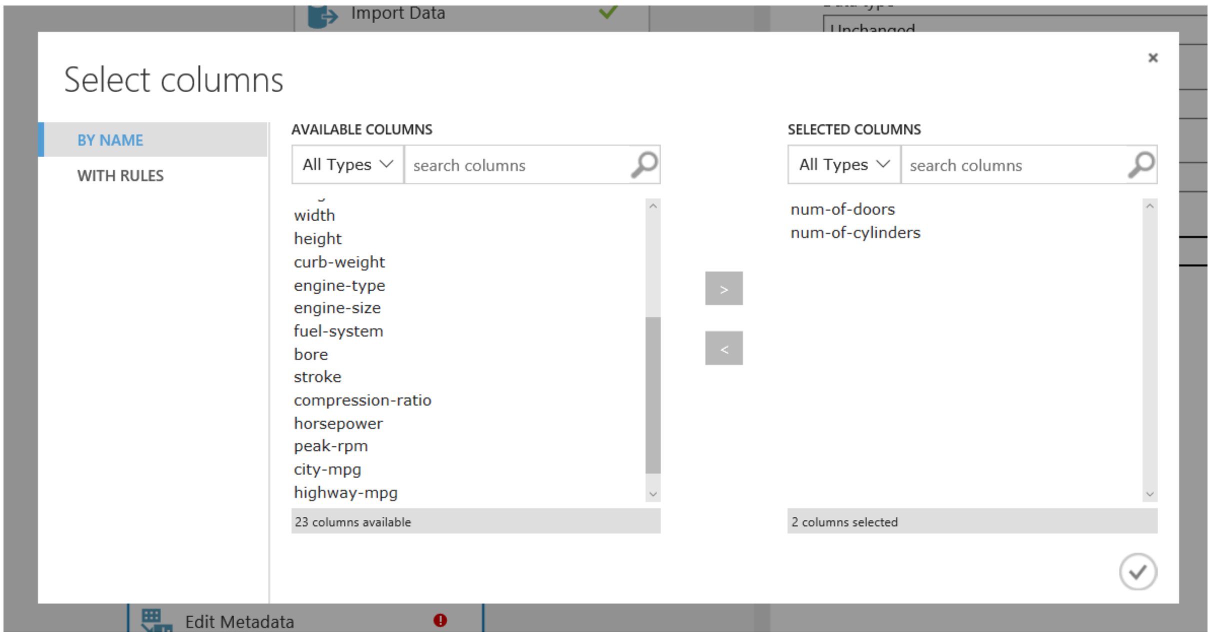
Edit Metadata

Column
Selected columns:
Launch the selector tool to make a selection
Launch column selector

Data type: Unchanged
Categorical: Unchanged
Fields: Unchanged
New column names:

Quick Help
Edits metadata associated with columns in a dataset. Formerly known as Metadata Editor.
(more help...)

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



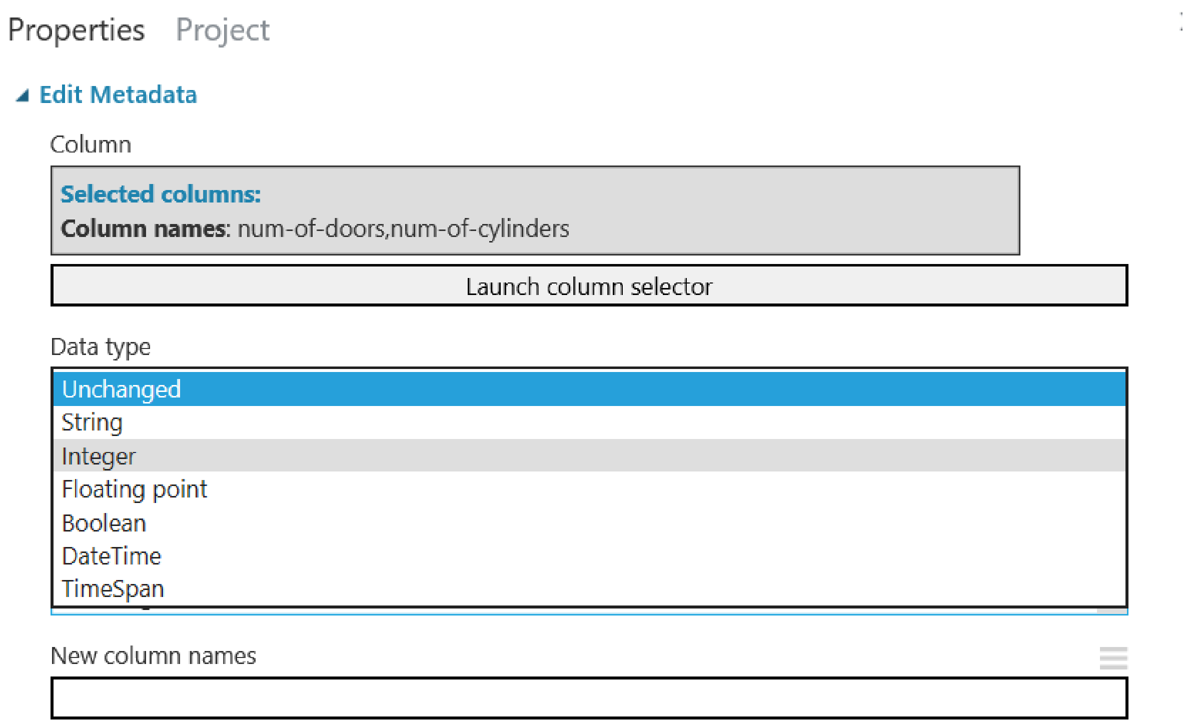
Available columns: width, height, curb-weight, engine-type, engine-size, fuel-system, bore, stroke, compression-ratio, horsepower, peak-rpm, city-mpg, highway-mpg

23 columns available

Selected columns: num-of-doors, num-of-cylinders

2 columns selected

Next change the data type to integer using the drop down available. Then run the experiment.

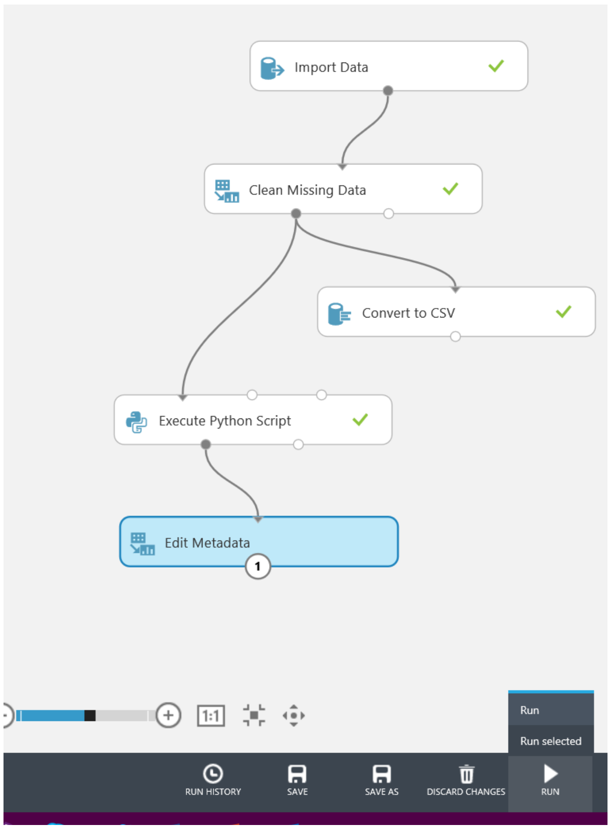


Column

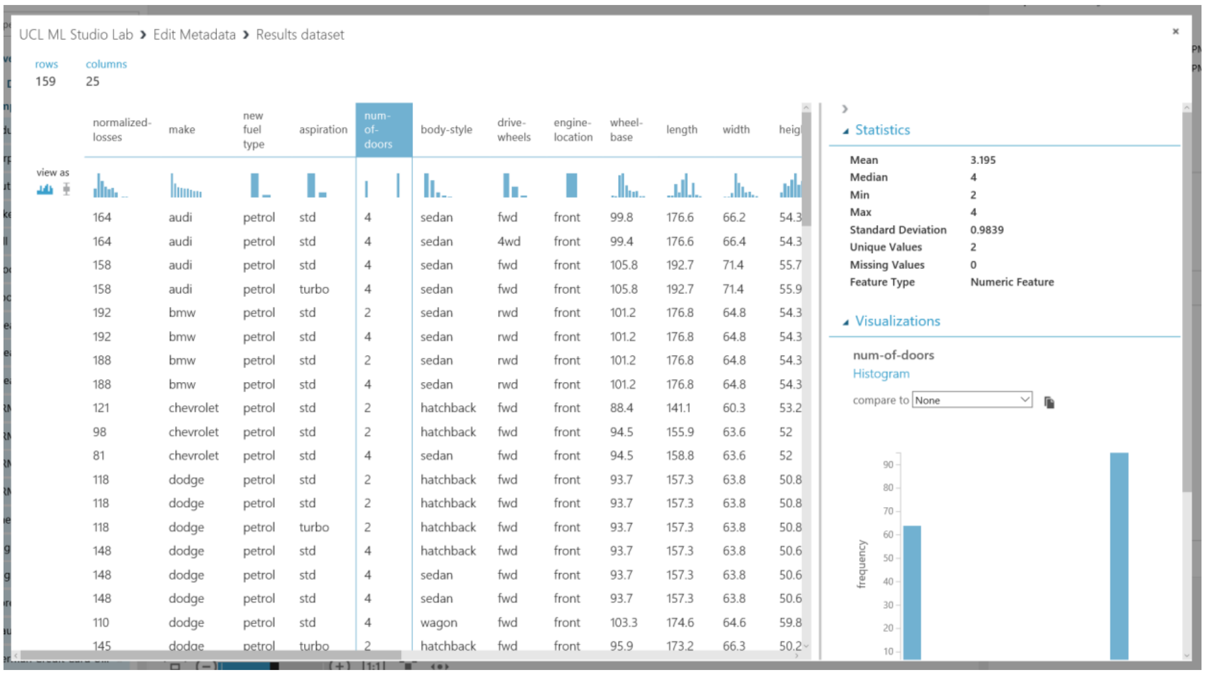
Selected columns:
Column names: num-of-doors,num-of-cylinders

Data type

Note: when only running a subsection of the experiment in Azure ML Studio, you can hover over the run button and choose 'Run Selected' and the selected modules will light up as below. Try this out now



Once run, visualise the data from the Edit Metadata module and you should see that the columns selected have now been changed to numeric data type:


UCL ML Studio Lab › Edit Metadata › Results dataset

rows 159 columns 25

normalized-losses	make	new fuel type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	heig
164	audi	petrol	std	4	sedan	fwd	front	99.8	176.6	66.2	54.3
164	audi	petrol	std	4	sedan	4wd	front	99.4	176.6	66.4	54.3
158	audi	petrol	std	4	sedan	fwd	front	105.8	192.7	71.4	55.7
158	audi	petrol	turbo	4	sedan	fwd	front	105.8	192.7	71.4	55.9
192	bmw	petrol	std	2	sedan	rwd	front	101.2	176.8	64.8	54.3
192	bmw	petrol	std	4	sedan	rwd	front	101.2	176.8	64.8	54.3
188	bmw	petrol	std	2	sedan	rwd	front	101.2	176.8	64.8	54.3
188	bmw	petrol	std	4	sedan	rwd	front	101.2	176.8	64.8	54.3
121	chevrolet	petrol	std	2	hatchback	fwd	front	88.4	141.1	60.3	53.2
98	chevrolet	petrol	std	2	hatchback	fwd	front	94.5	155.9	63.6	52
81	chevrolet	petrol	std	4	sedan	fwd	front	94.5	158.8	63.6	52
118	dodge	petrol	std	2	hatchback	fwd	front	93.7	157.3	63.8	50.8
118	dodge	petrol	std	2	hatchback	fwd	front	93.7	157.3	63.8	50.8
118	dodge	petrol	turbo	2	hatchback	fwd	front	93.7	157.3	63.8	50.8
148	dodge	petrol	std	4	hatchback	fwd	front	93.7	157.3	63.8	50.6
148	dodge	petrol	std	4	sedan	fwd	front	93.7	157.3	63.8	50.6
148	dodge	petrol	std	4	sedan	fwd	front	93.7	157.3	63.8	50.6
110	dodge	petrol	std	4	wagon	fwd	front	103.3	174.6	64.6	59.8
145	dodge	petrol	turbo	2	hatchback	fwd	front	95.9	173.2	66.3	50.2

Statistics

Mean	3.195
Median	4
Min	2
Max	4
Standard Deviation	0.9839
Unique Values	2
Missing Values	0
Feature Type	Numeric Feature

Visualizations

num-of-doors
Histogram
compare to None

In this section, we have cleaned missing data from the dataset, edited values in the dataset using a python script and then changed the meta data associated with the columns in the dataset to understand when a column is numeric.

Now we have our dataset in good shape after pre-processing to start training a model in the next section

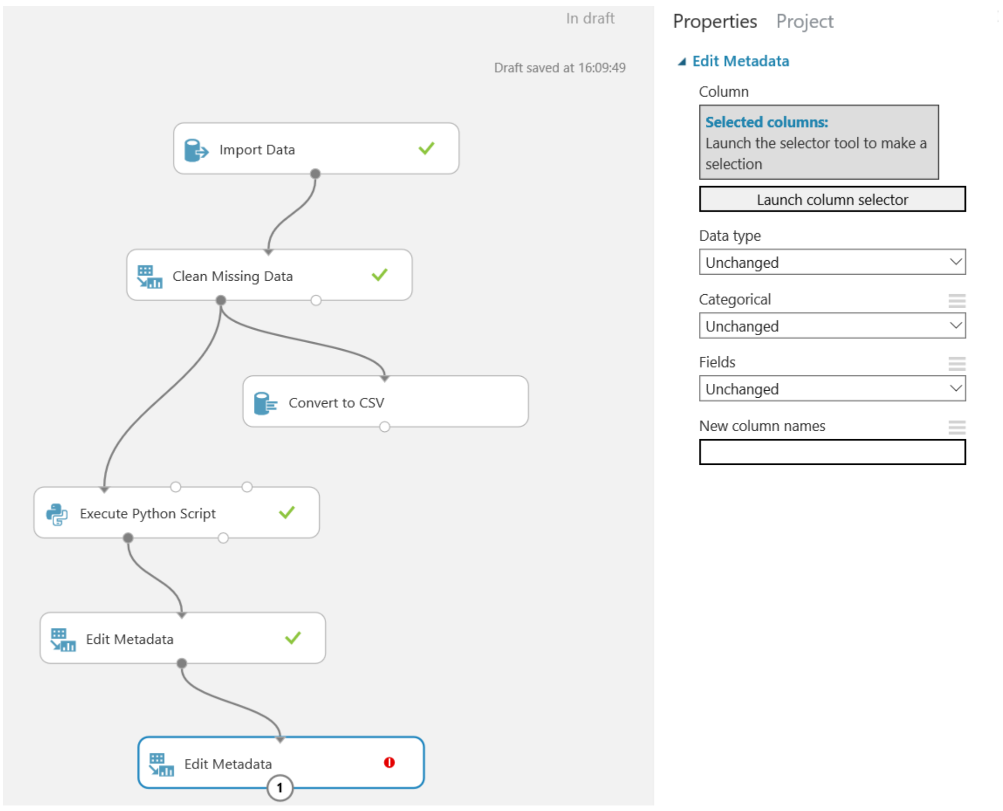
Training the Model

In this section, we will start training a regression model using an algorithm and our pre-processed data to predict the price of a car.

We need to tell the Azure ML studio which columns in our dataset are labels and which columns are features to be operated on.

- Labels: Our label is price – as we want to predict the value of a car given all its attributes
- Features: The features in our dataset are the rest of the columns, not including the price. Things like make, engine size, MPG, horse power etc.

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



Edit Metadata

Column

Selected columns:
Launch the selector tool to make a selection

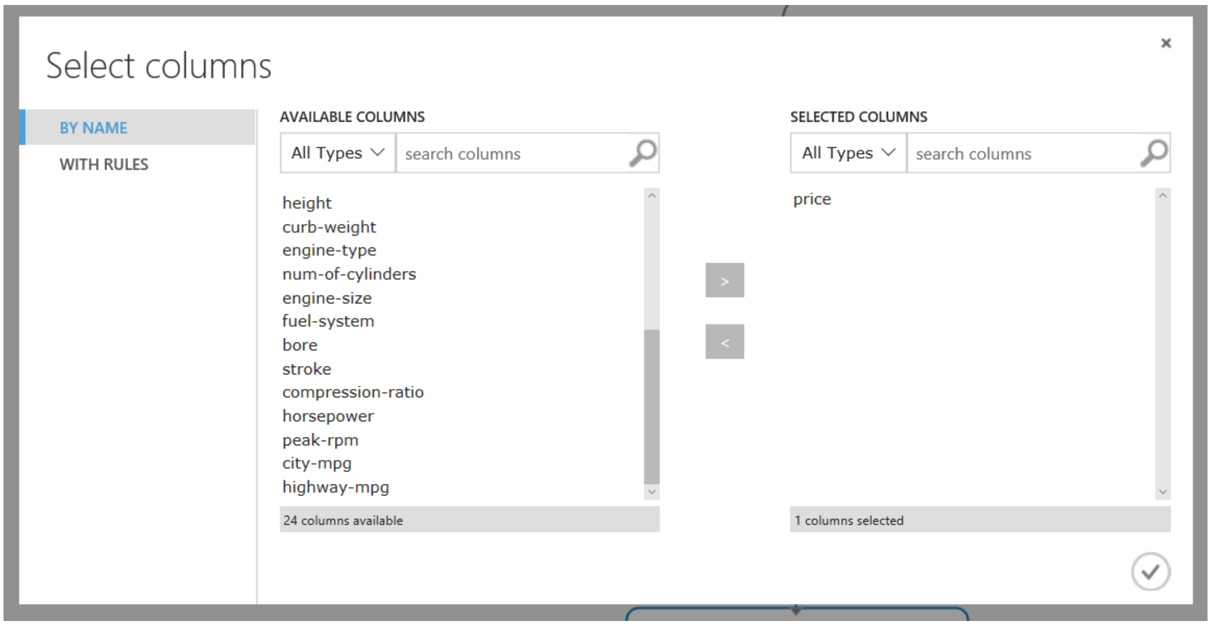
Data type: Unchanged

Categorical: Unchanged

Fields: Unchanged

New column names

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


Select columns

BY NAME
WITH RULES

AVAILABLE COLUMNS
All Types | search columns

height
curb-weight
engine-type
num-of-cylinders
engine-size
fuel-system
bore
stroke
compression-ratio
horsepower
peak-rpm
city-mpg
highway-mpg

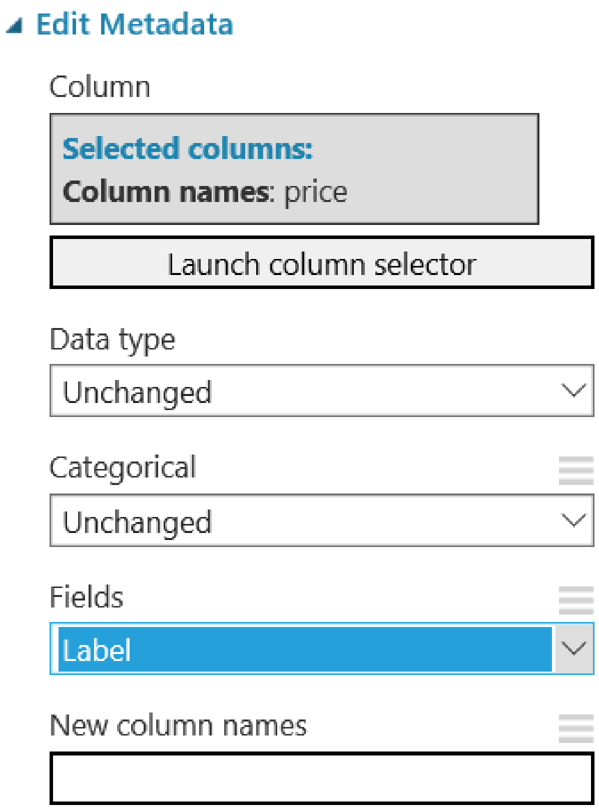
24 columns available

SELECTED COLUMNS
All Types | search columns

price

1 columns selected

Next choose the Field property and select ‘Label’ from the drop-down menu


Edit Metadata

Column
Selected columns:
Column names: price
Launch column selector

Data type: Unchanged

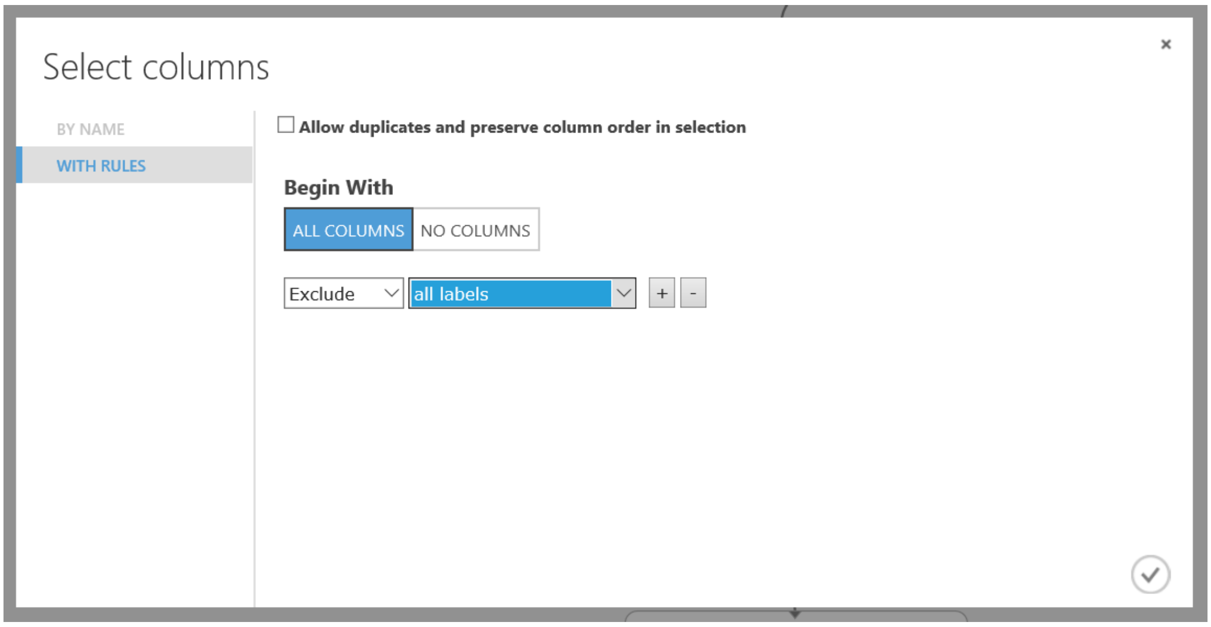
Categorical: Unchanged

Fields: Label

New column names:

Now let’s do the same process but 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. Then you want to ‘Begin With’ ‘All Columns’ and ‘Exclude’ ‘All Labels’. See the screenshot below for clarification:


Select columns

BY NAME

WITH RULES

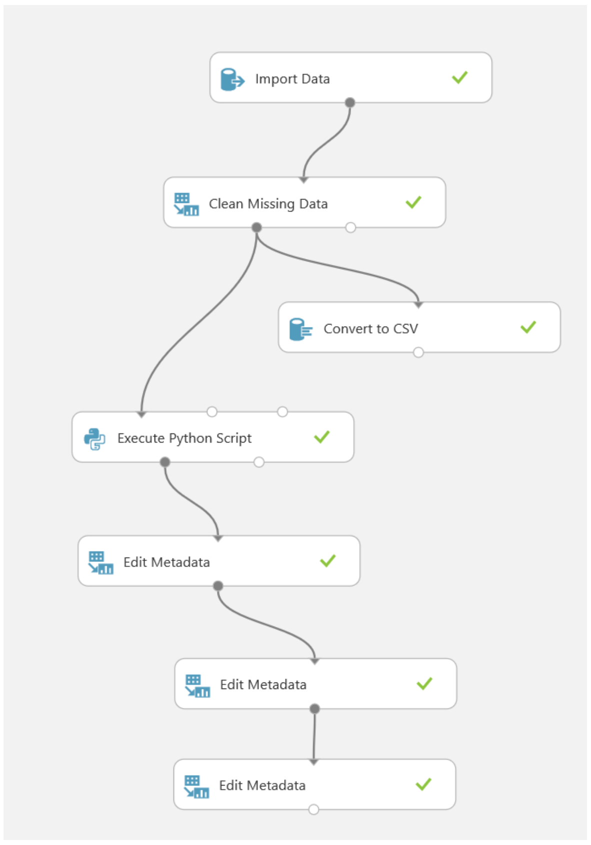
☐ Allow duplicates and preserve column order in selection

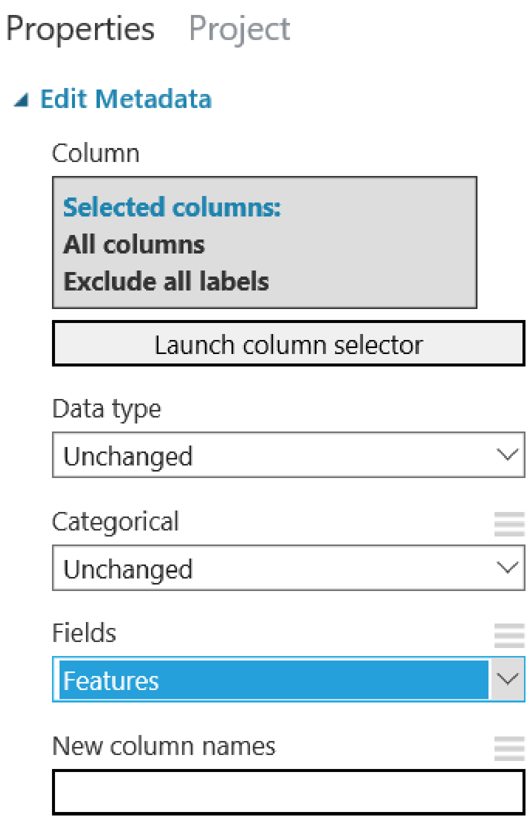
Begin With

[✓ ALL COLUMNS] NO COLUMNS

Exclude | all labels

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




Properties Project

Edit Metadata

Column

Selected columns:
All columns
Exclude all labels

Launch column selector

Data type: Unchanged

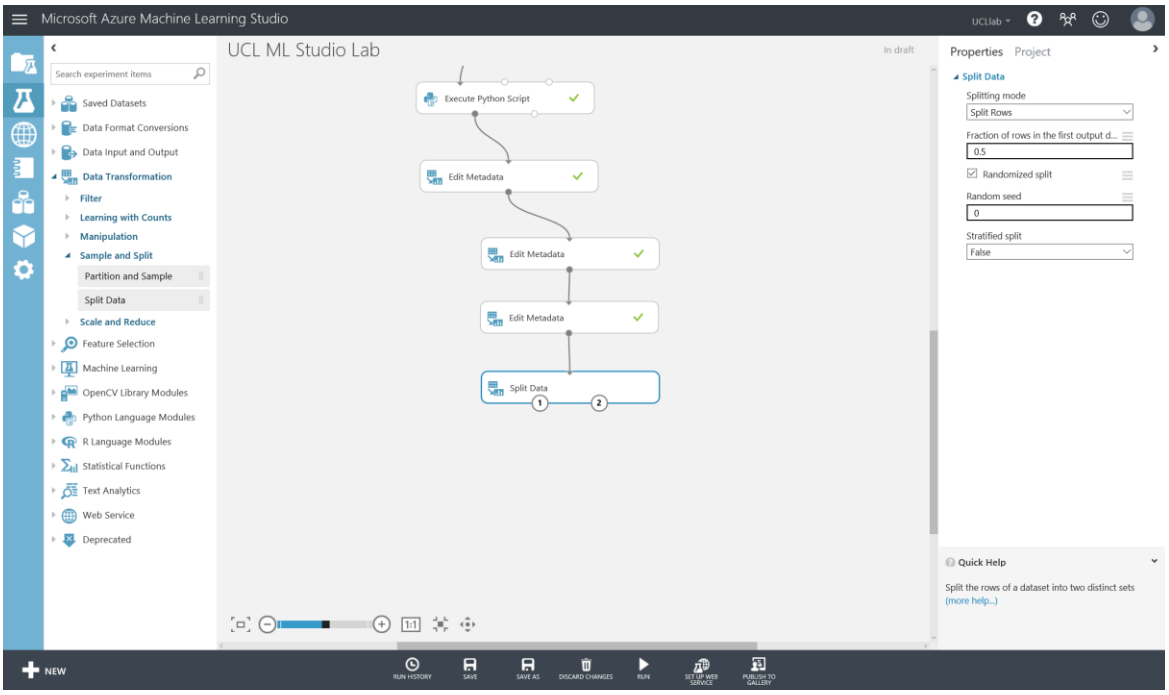
Categorical: Unchanged

Fields: Features

New column names:

Next we need to split the dataset into training and testing datasets. One dataset will be used to train the model on the patterns and correlations in the data, the other set will be held back and not trained on, to score the accuracy of the model built from the training data.

Find the 'Split Data' module under the Data Transformation -> Sample and Split headings in the module list. Drag onto the experiment space and connect to the output port of the last module in the experiment.



Properties Project

Split Data

Splitting mode
Split Rows

Fraction of rows in the first output d...
0.5

☑ Randomized split

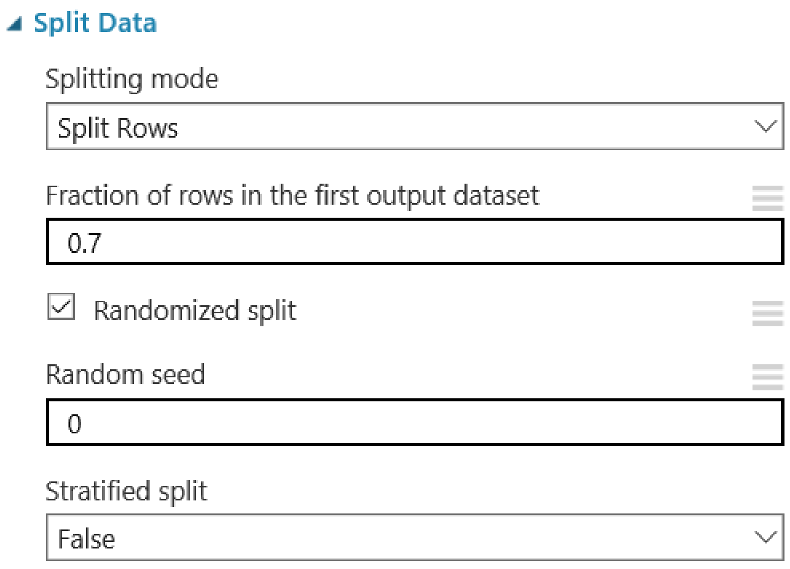
Random seed
0

Stratified split
False

Quick Help

Split the rows of a dataset into two distinct sets
(more help...)

In the properties pane you want to keep the 'splitting method' as 'split rows'. However we want to split the dataset 70% training data and 30% testing data. Therefore 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. You don't need to set a random seed and keep 'Stratified Split' as False.



Split Data

Splitting mode
Split Rows

Fraction of rows in the first output dataset
0.7

☑ Randomized split

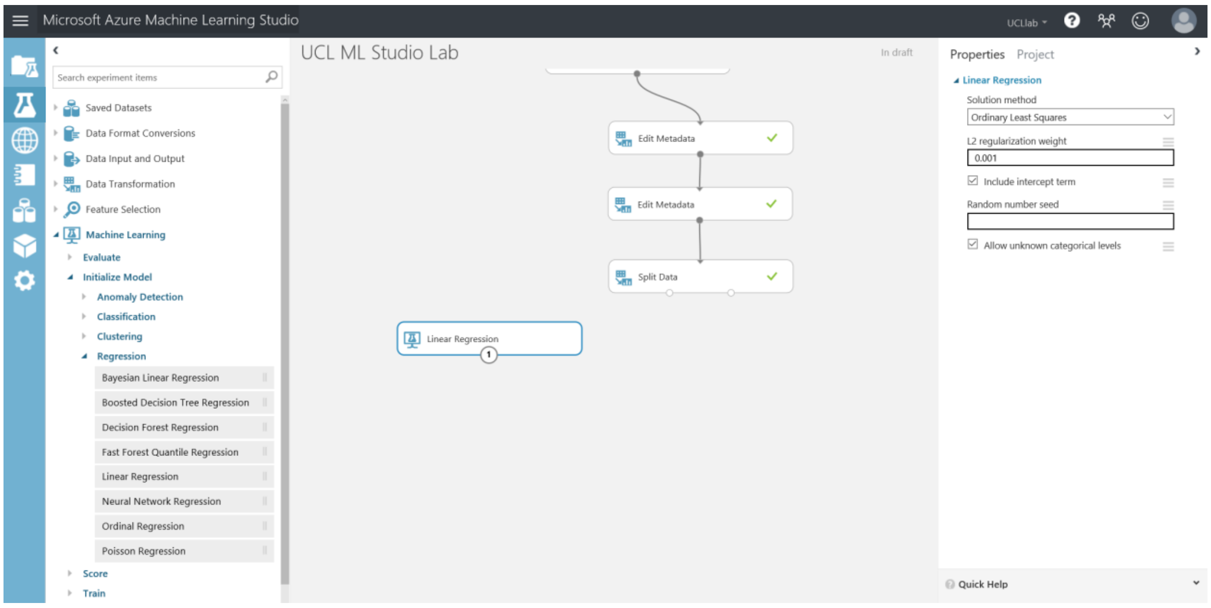
Random seed
0

Stratified split
False

Now 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 onto the experiment surface as a pre-built algorithm inside of Azure ML studio.



Linear Regression

Solution method: Ordinary Least Squares

L2 regularization weight: 0.001

☑ Include intercept term

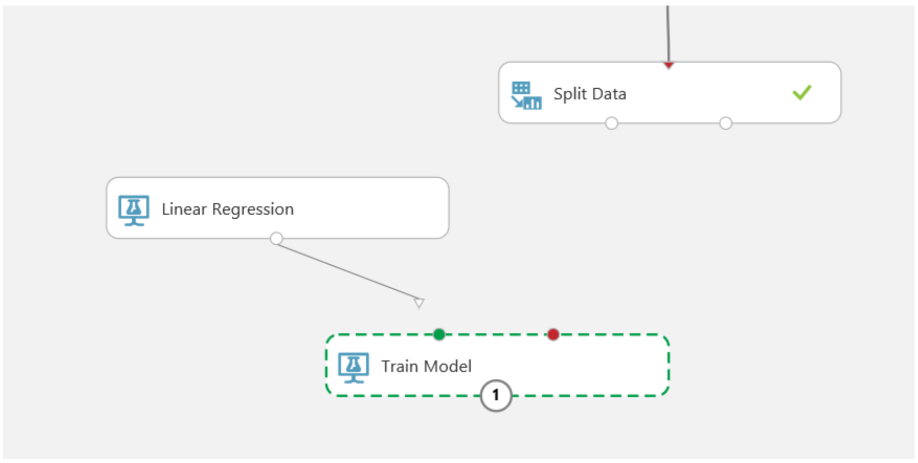
Random number seed:

☑ Allow unknown categorical levels

Each one of the pre-built algorithms provided in Azure Machine Learning Studio allows you to tweak the parameters associated. In every case default parameters are supplied for each algorithm so that you can quickly build out experiments. We will keep the parameters of the Linear Regression algorithm as default for now.

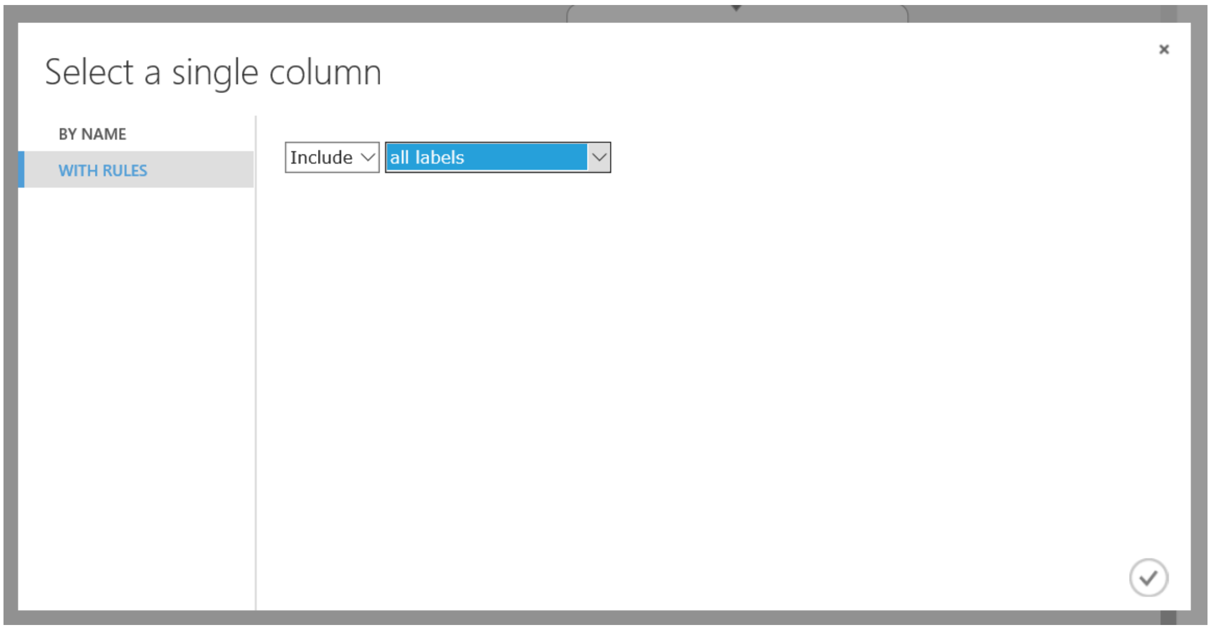
To combine the algorithm and our dataset we need to bring on a 'Train Model' module. This takes an untrained algorithm and dataset as inputs and produces a trained model out of the module output port.

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. You won't be able to wire these up the wrong way, as the model gives you guidance on the input type its expects as seen in the screenshots below:



Once connected up, 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 will allow the train model module to know it needs to focus on learning the price of the car (label) given the attributes and patterns in the dataset (features)



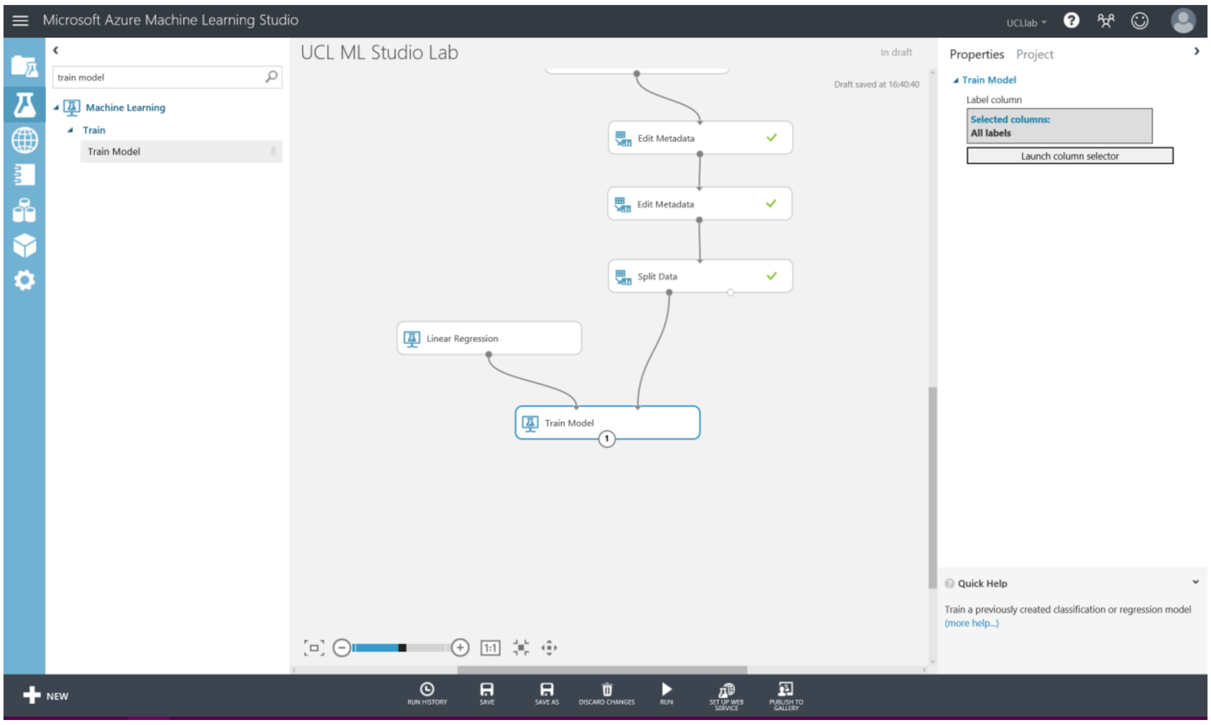
Select a single column

BY NAME

WITH RULES

Include | all labels

Select the tick box



Properties Project

Train Model

Label column

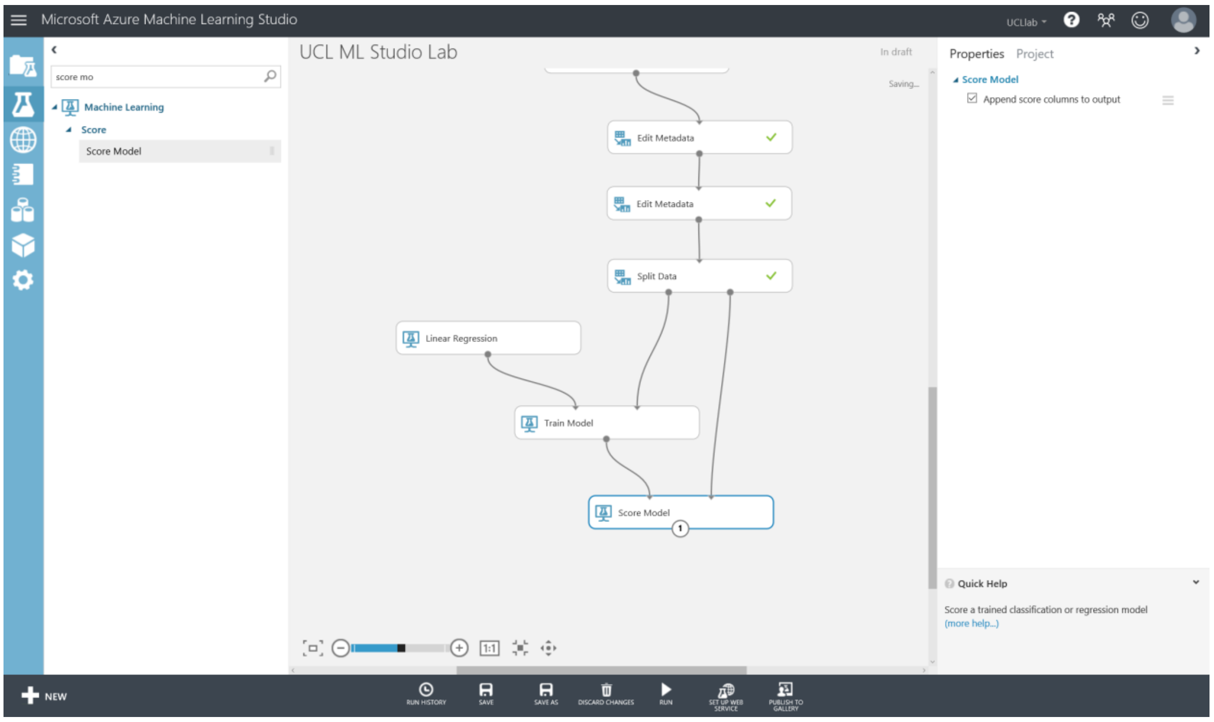
Selected columns:
All labels

Launch column selector

Quick Help

Train a previously created classification or regression model
(more help...)

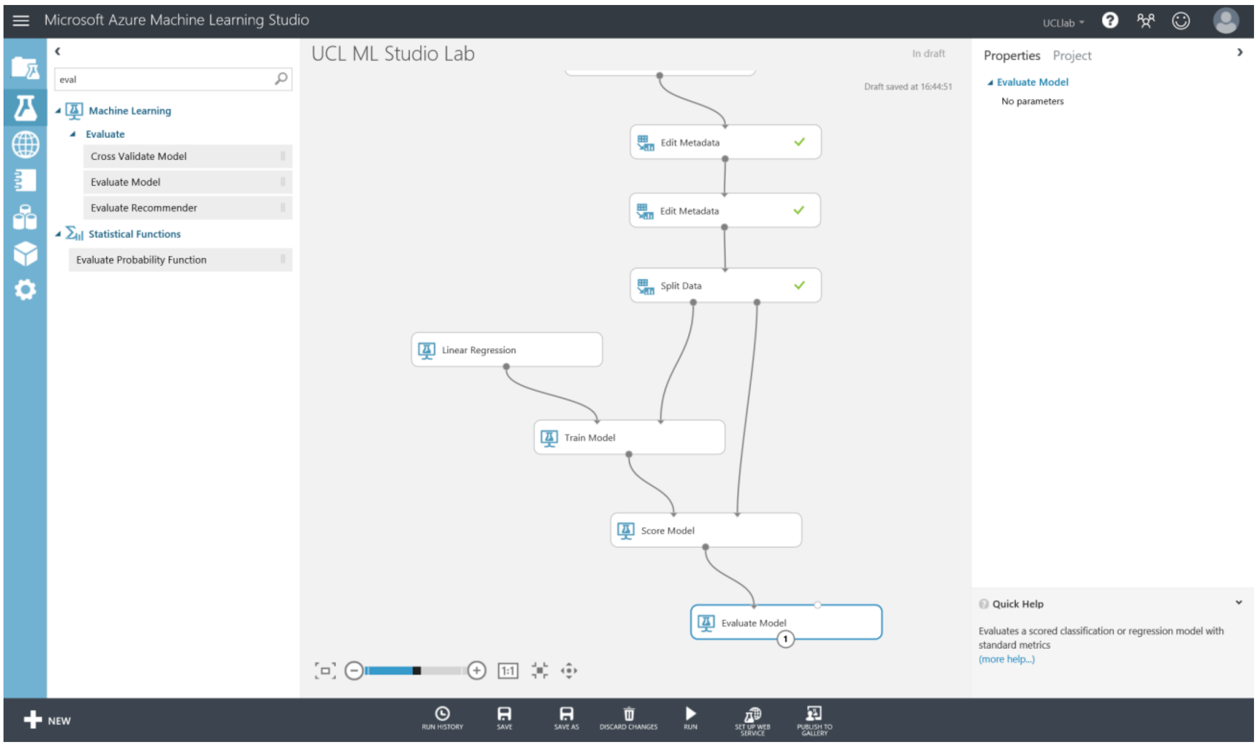
Once the model is trained (output port of the Train Model module) we need to score the model on how well it can predict the price of a car. This is where we connect the trained model to a 'Score Model' module and the testing dataset (the right output port of the Split Data module).



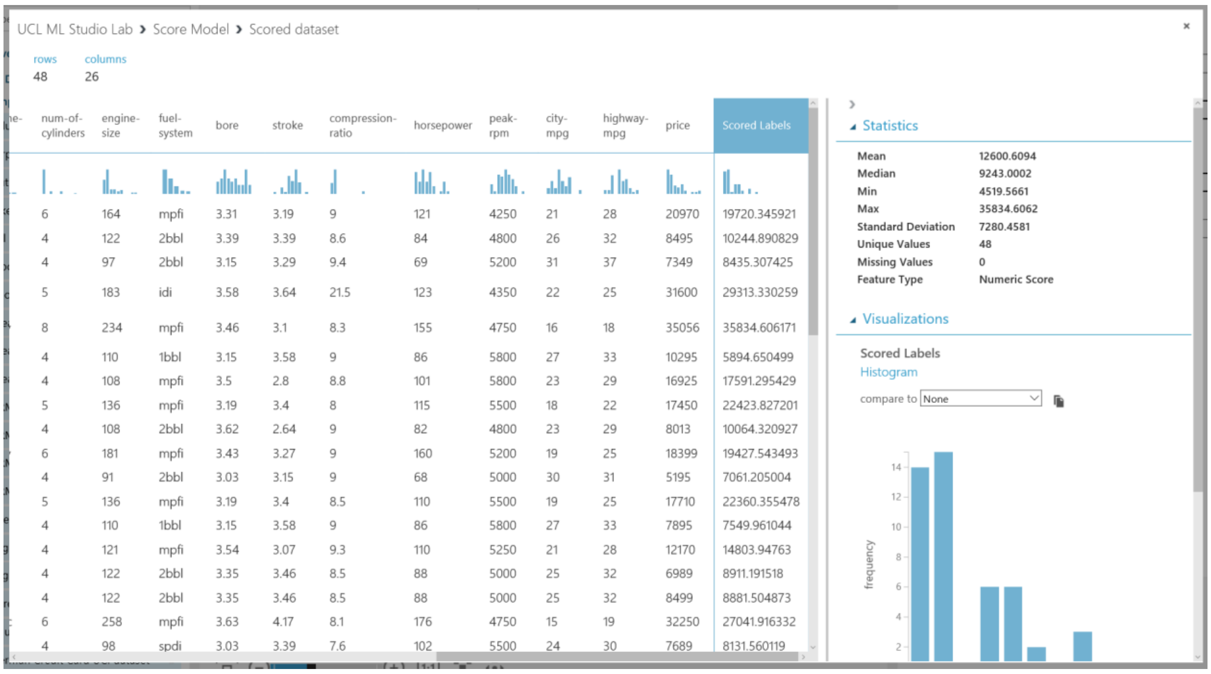
There are no parameters to set on this module as it takes the testing dataset and queries the trained model using that data, it asks it to predict the price of each car in that testing dataset. Inside the testing data we have the actual price of the car still, therefore we can start to measure the error value of the regression model – how far is the predicted value of the car price compared to the actual price of the car.

Finally, in order to view some simple statistics on this model connect an '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.



Once run, let's look at the outcome of the model. Select the output of the Score Model module and right click -> visualise the data. Scroll the end of the dataset and find the Scored Labels column. This is what the trained regression model has predicted compared to the actual value column next to it, 'price'.


UCL ML Studio Lab › Score Model › Scored dataset

rows	columns
48	26

num-of-cylinders	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price	Scored Labels
6	164	mpfi	3.31	3.19	9	121	4250	21	28	20970	19720.345921
4	122	2bbl	3.39	3.39	8.6	84	4800	26	32	8495	10244.890829
4	97	2bbl	3.15	3.29	9.4	69	5200	31	37	7349	8435.307425
5	183	idi	3.58	3.64	21.5	123	4350	22	25	31600	29313.330259
8	234	mpfi	3.46	3.1	8.3	155	4750	16	18	35056	35834.606171
4	110	1bbl	3.15	3.58	9	86	5800	27	33	10295	5894.650499
4	108	mpfi	3.5	2.8	8.8	101	5800	23	29	16925	17591.295429
5	136	mpfi	3.19	3.4	8	115	5500	18	22	17450	22423.827201
4	108	2bbl	3.62	2.64	9	82	4800	23	29	8013	10064.320927
6	181	mpfi	3.43	3.27	9	160	5200	19	25	18399	19427.543493
4	91	2bbl	3.03	3.15	9	68	5000	30	31	5195	7061.205004
5	136	mpfi	3.19	3.4	8.5	110	5500	19	25	17710	22360.355478
4	110	1bbl	3.15	3.58	9	86	5800	27	33	7895	7549.961044
4	121	mpfi	3.54	3.07	9.3	110	5250	21	28	12170	14803.94763
4	122	2bbl	3.35	3.46	8.5	88	5000	25	32	6989	8911.191518
4	122	2bbl	3.35	3.46	8.5	88	5000	25	32	8499	8881.504873
6	258	mpfi	3.63	4.17	8.1	176	4750	15	19	32250	27041.916332
4	98	spdi	3.03	3.39	7.6	102	5500	24	30	7689	8131.560119

Statistics

Mean	12600.6094
Median	9243.0002
Min	4519.5661
Max	35834.6062
Standard Deviation	7280.4581
Unique Values	48
Missing Values	0
Feature Type	Numeric Score

Visualizations

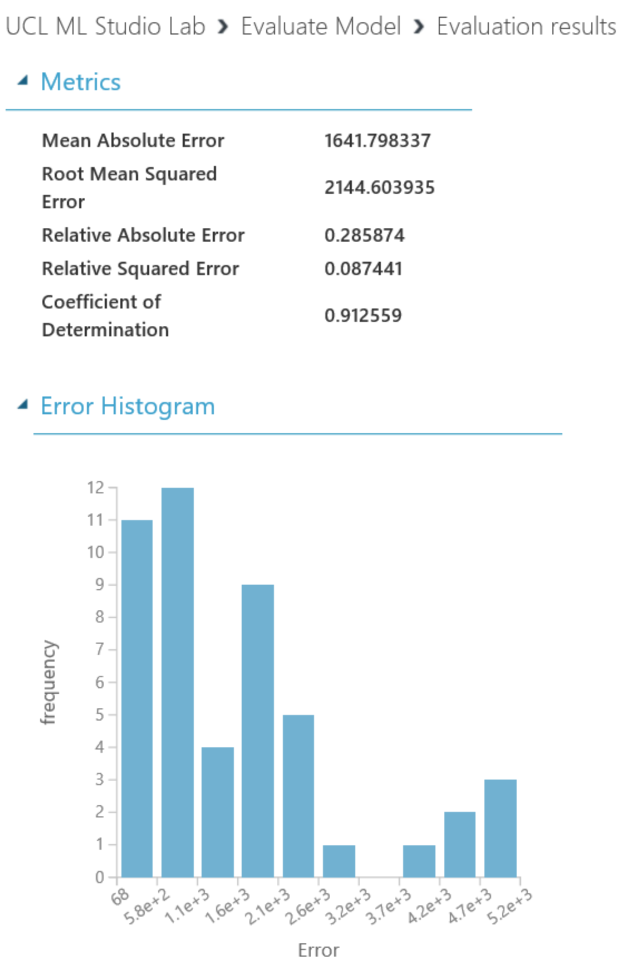
Scored Labels
Histogram

compare to None

Now if we look at the output of the Evaluate Model module, this will provide us with more statistics relating to Scored Labels. Select Evaluate Model module and right click -> visualise

The metrics returned for regression models are generally designed to estimate the amount of error. A model is considered to fit the data well if the difference between observed and predicted values is small. However, looking at the pattern of the residuals (the difference between any one predicted point and its corresponding actual value) can tell you a lot about potential bias in the model.

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.



UCL ML Studio Lab › Evaluate Model › Evaluation results

Metrics

Mean Absolute Error	1641.798337
Root Mean Squared Error	2144.603935
Relative Absolute Error	0.285874
Relative Squared Error	0.087441
Coefficient of Determination	0.912559

Error Histogram

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.

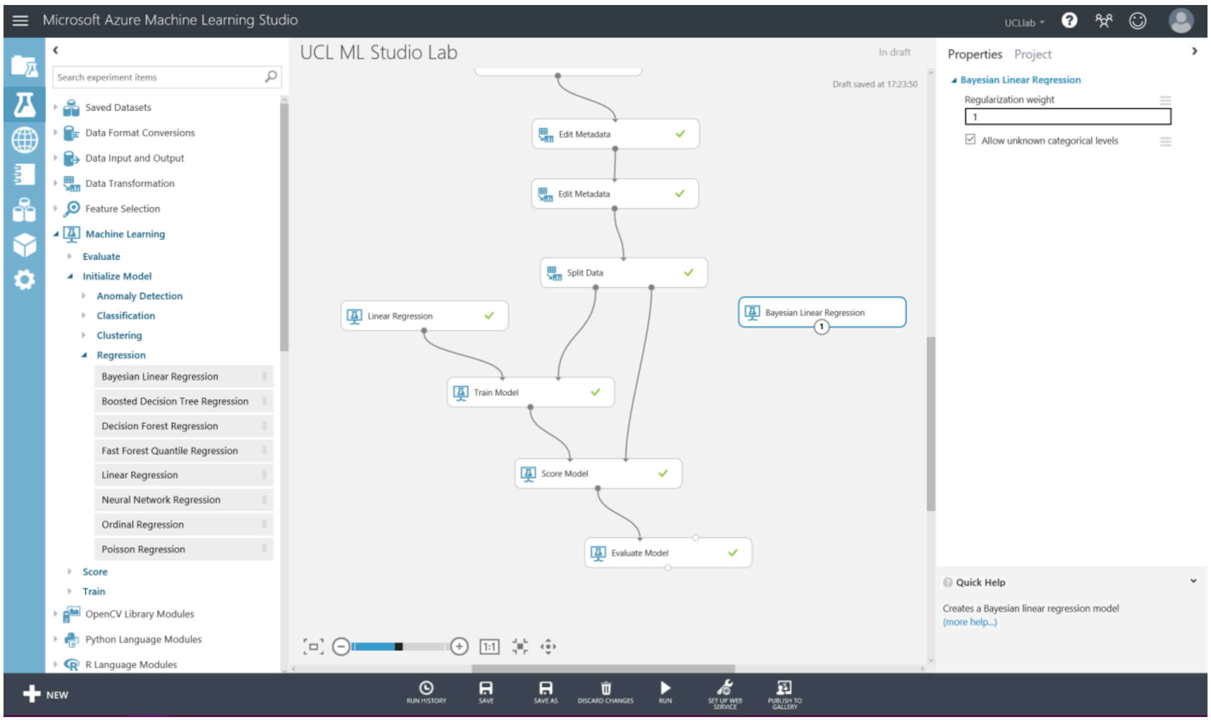
Comparing Algorithms

Now we have created an initial machine learning model, however to improve this model we need to experiment with different algorithms, features, removing outliers, further data processing etc.

In this section, you will see how to compare multiple regression algorithms and evaluate them against in other to understand which one is the best to move forward with. This is a great advantage of Azure Machine Learning Studio, the ability to quickly test new experiment setups.

Carrying on from the experiment above we are going to add 'Bayesian Linear Regression' algorithm. Go to the modules pane on the left of the screen and find the section:

Machine Learning -> Initialise Model -> Regression -> Bayesian Linear Regression

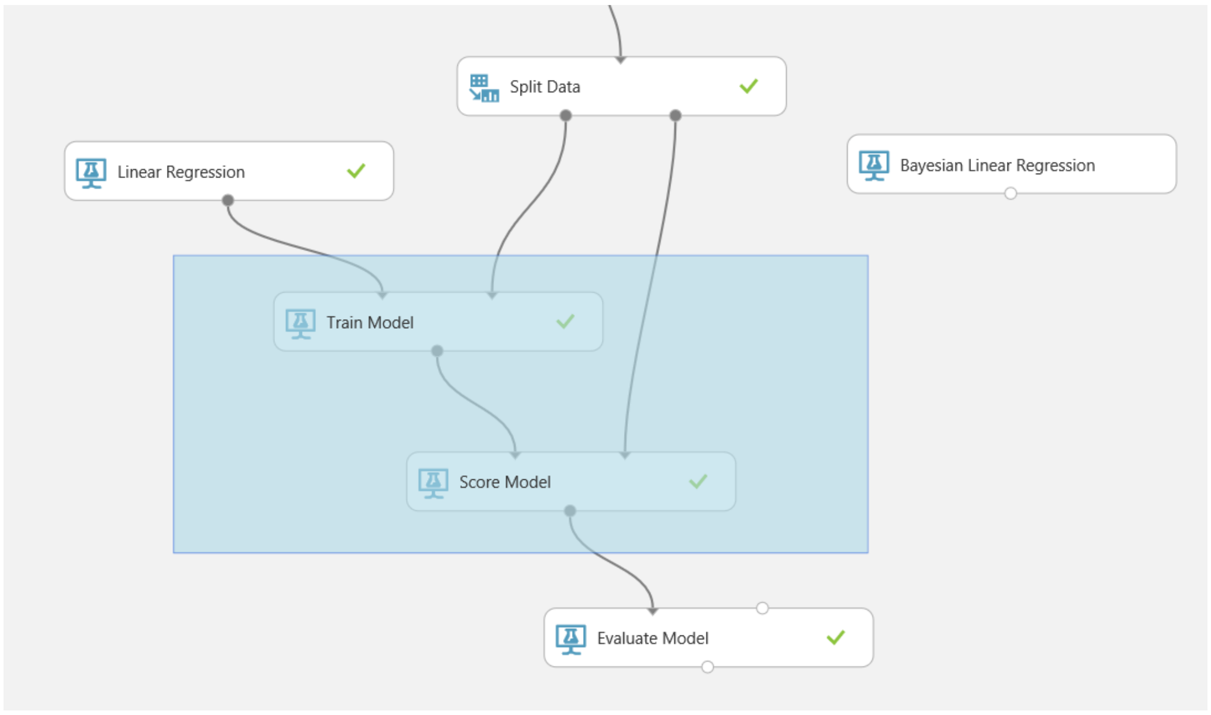


Bayesian Linear Regression

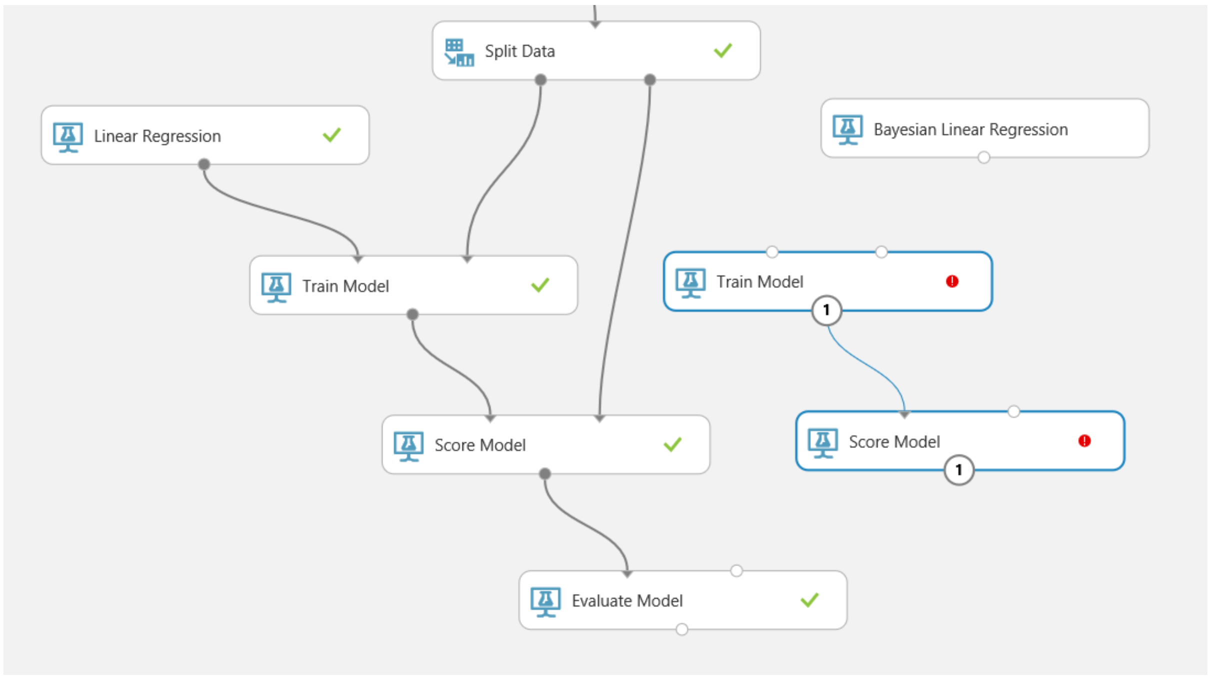
Regularization weight: 1

☑ Allow unknown categorical levels

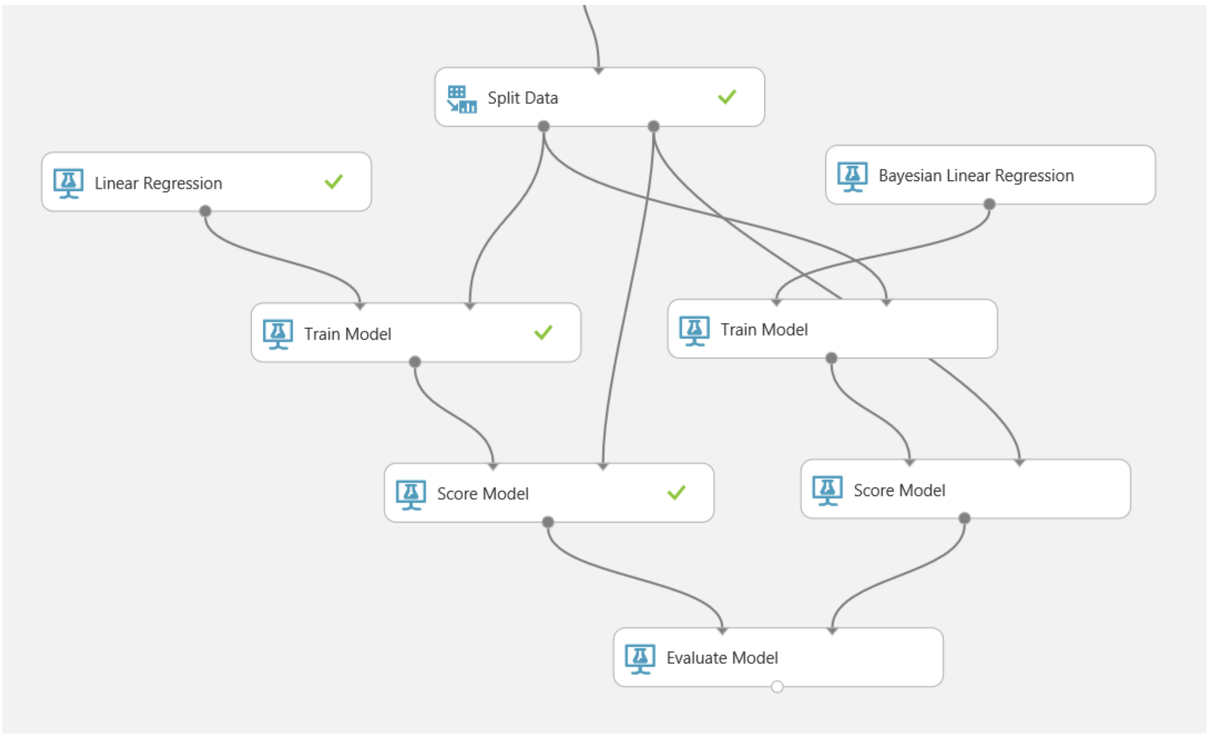
To quickly recreate the same train, score and evaluate setup with the same parameters you can click and select multiple modules:



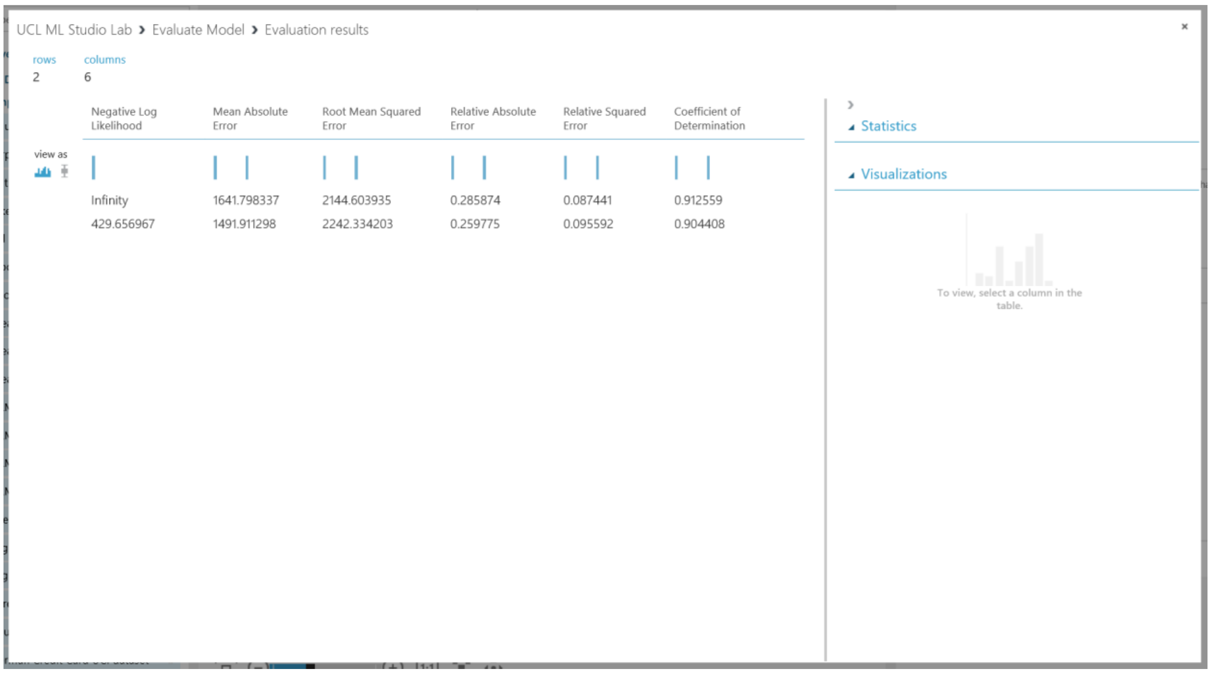
Now use the hot keys: ctrl+c and ctrl+v to copy and paste the models



And connect the modules up as shown below, so that the algorithm and training dataset are connected to then Train Model module, and the testing dataset is connected to the Score Model module. Finally, the output of the recently copied Score Model module should be connected to the right input port of the Evaluate Model module.



Now run the experiment. Once complete, right click and visualise the Evaluate module.


UCL ML Studio Lab › Evaluate Model › Evaluation results
view as
Statistics
Visualizations
To view, select a column in the table.

rows 2 columns 6

Negative Log Likelihood	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
Infinity	1641.798337	2144.603935	0.285874	0.087441	0.912559
429.656967	1491.911298	2242.334203	0.259775	0.095592	0.904408

When comparing the two algorithms we can see the Mean absolute error is slightly lower for the Bayesian linear regression however the Root Mean Squared error is slightly higher, so these two models are similar in accuracy and decisions must be made around what evaluation metric is important for the model/question you are building.

Conclusion

This lab was intended to introduce you to the basic steps in building a Machine Learning model in the Azure Machine Learning Studio. We covered data input, data pre-processing, regression, training, testing and evaluating a model and using Azure Machine Learning.

Next Steps:

- Check out the Azure Machine Learning Gallery and download, edit and run other types of machine learning experiments (classification, clustering): https://gallery.cortanaintelligence.com/
- Find other examples of Jupyter Notebooks in Azure Machine Learning: https://notebooks.azure.com/
- Keep up to date with new features in Azure Machine Learning and see how others use the service through the blog: https://blogs.technet.microsoft.com/machinelearning/