Microsoft

Machine Learning

Azure Machine Learning Studio
- Publishing Lab

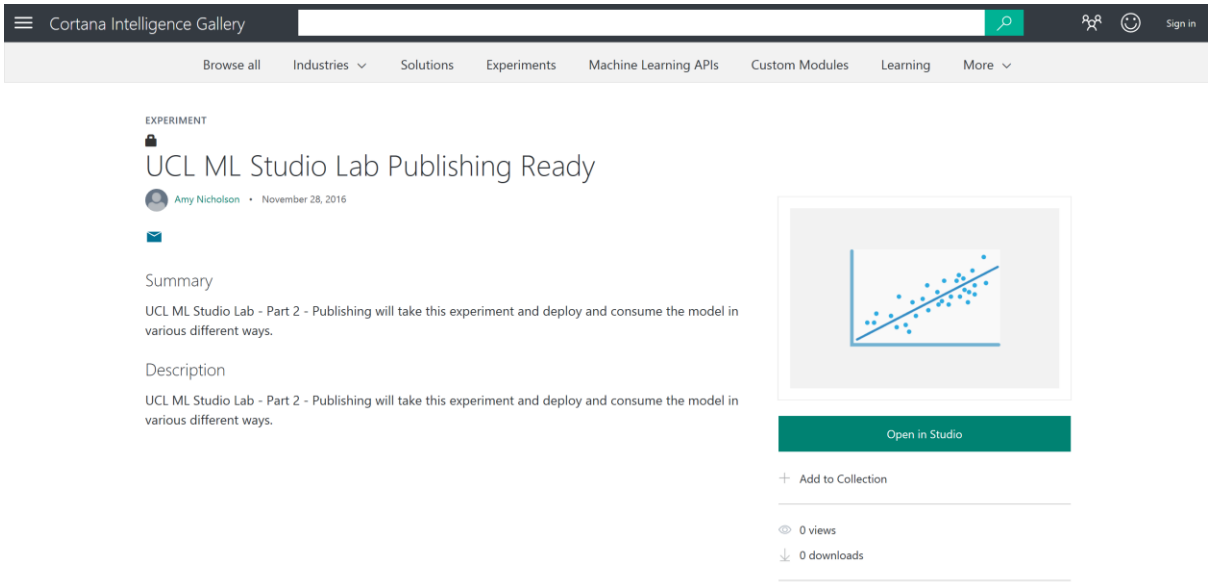
Contents

What Did We Cover Last Time?

The previous lab 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 multiple models and evaluating them. Throughout your course you have also been looking at Jupyter Notebook as part of Azure Machine Learning.

This lab will carry on from the previous lab on Azure Machine Learning and look at publishing the regression model from within the studio to create an API, as well as via Jupyter Notebooks. Then review the sample code for a published experiment and test the output from an Excel add-in/application. Then finally look at how you can manage the usage of your machine learning models from the web service portal.

If you completed the last lab you can continue building on your experiment named 'UCL ML Studio Lab' or get the completed experiment from the gallery here: http://gallery.cortanaintelligence.com/Experiment/UCL-ML-Studio-Lab-Publishing-Ready-1



Choose 'Open in Studio' place into your workspace and 'Run' the model from the bottom toolbar before starting the rest of the lab.

The Problem Domain Explained

During the previous lab, we created and evaluated two models, that given past data collected about cars and their values, we predicted 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 were trying to predict is the price of the car. In this dataset, the values range from £5,000 to £45,000.

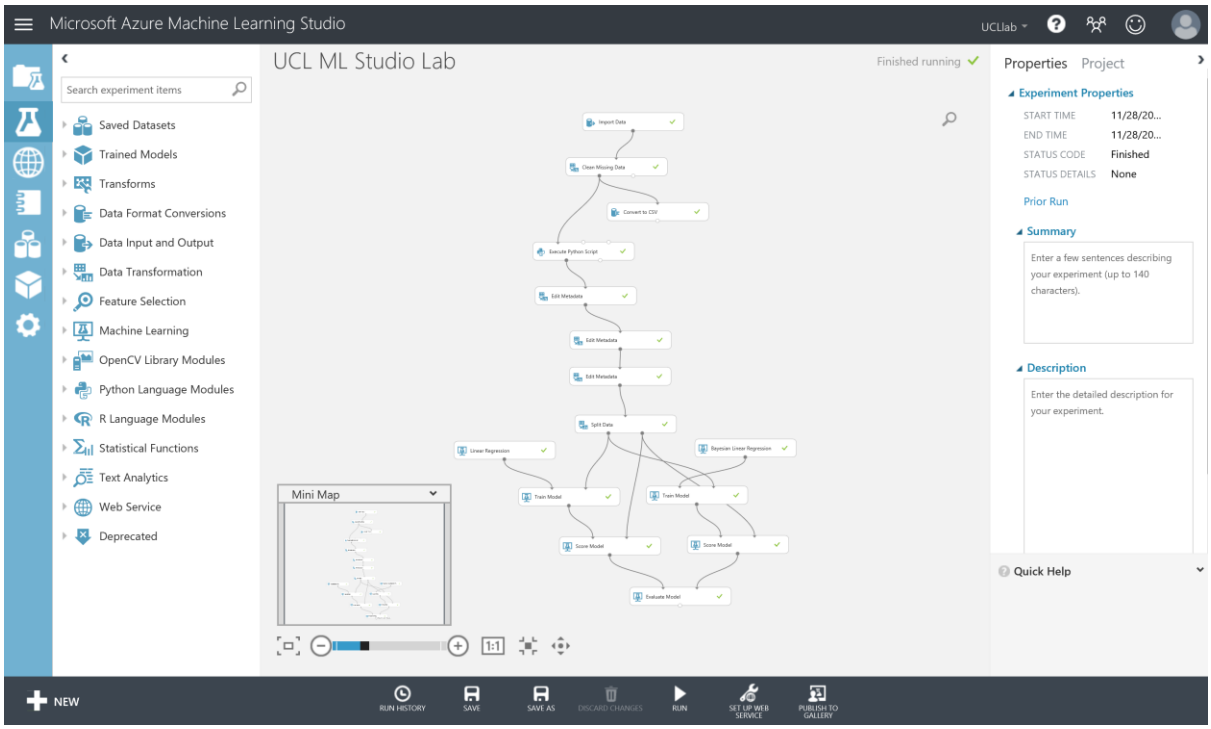
We retrieved data from an Azure Blob Storage account and started to pre-process the dataset ready to train a machine learning model. The model we created is a form of supervised learning so we used historical car attributes and values to predict the price of future cars we might receive. This model performs 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.

Now assume we are happy with our car prediction model and we want to build this capability into our applications, we need to be able to deploy the model. In this lab we will create a scoring experiment and deploy a model to create an API that we can call using a HTTP request.

Publishing a Web Service - UI

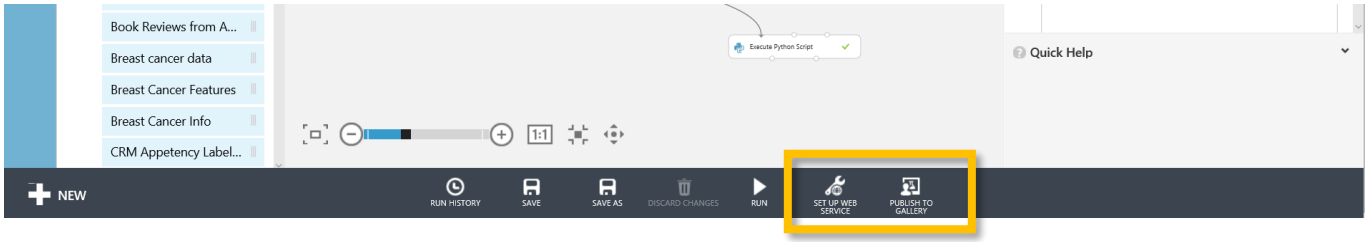
We have trained a model to predict a car's price given some attributes of the car, with the least error possible (a regression experiment). Now we want to use one of Azure Machine Learning's key features – Deploying the model, by publishing an API to expose the model we have created.

We have the 'UCL ML Studio Lab' experiment ready to be published and should look something like below (every module should have green ticks next to it):

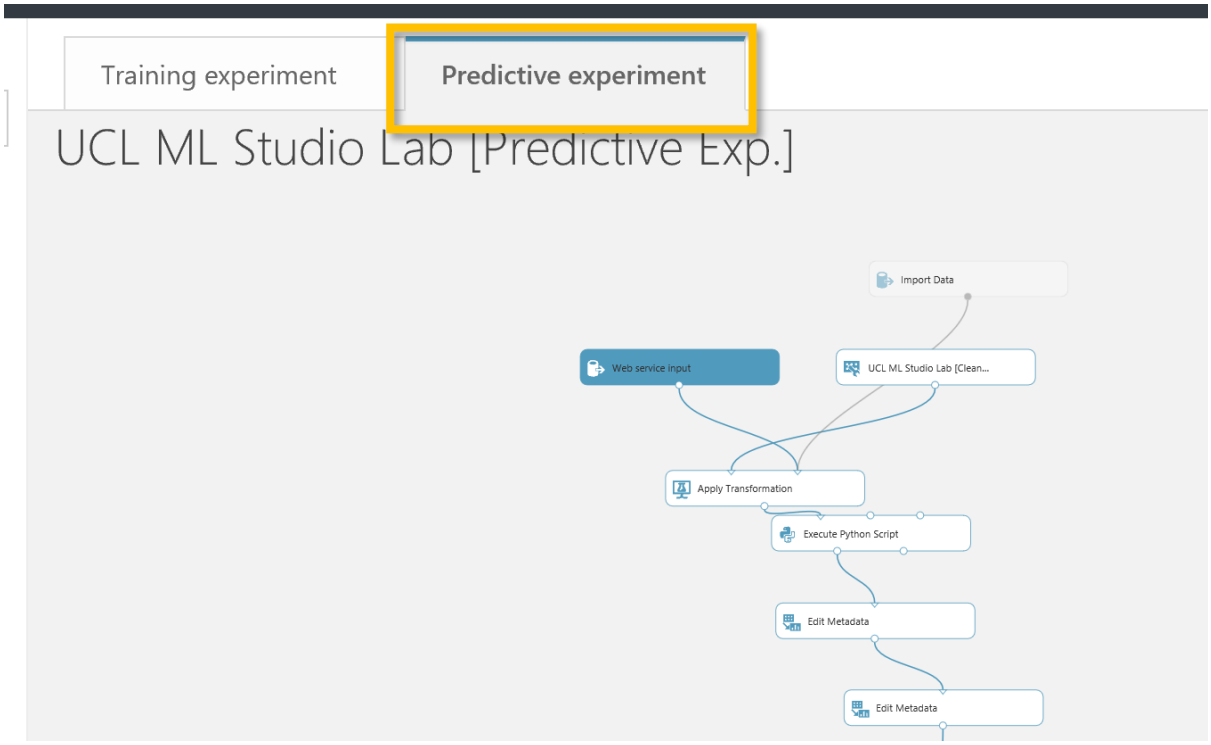


This model can then be published to the Azure ML Web API service to make it available for other users or applications to use as a web service or a REST endpoint.

This may sound like a hard task, however once your experiment has been run successfully - you will see a button on the toolbar at the bottom of the screen become active:



Select the train model module in your experiment associated with the Linear Regression model, then choose **'Set up web service'** and choose the **'Predictive Web Service (Recommended)'** option on the bottom tool bar. From here our experiment appears to get redrawn and consolidated. What has really happened, is that ML has created a new Predictive experiment tab at the top of the screen.



Our original experiment is still there, but is in the Training experiment tab. Click the training experiment tab to see this in action.

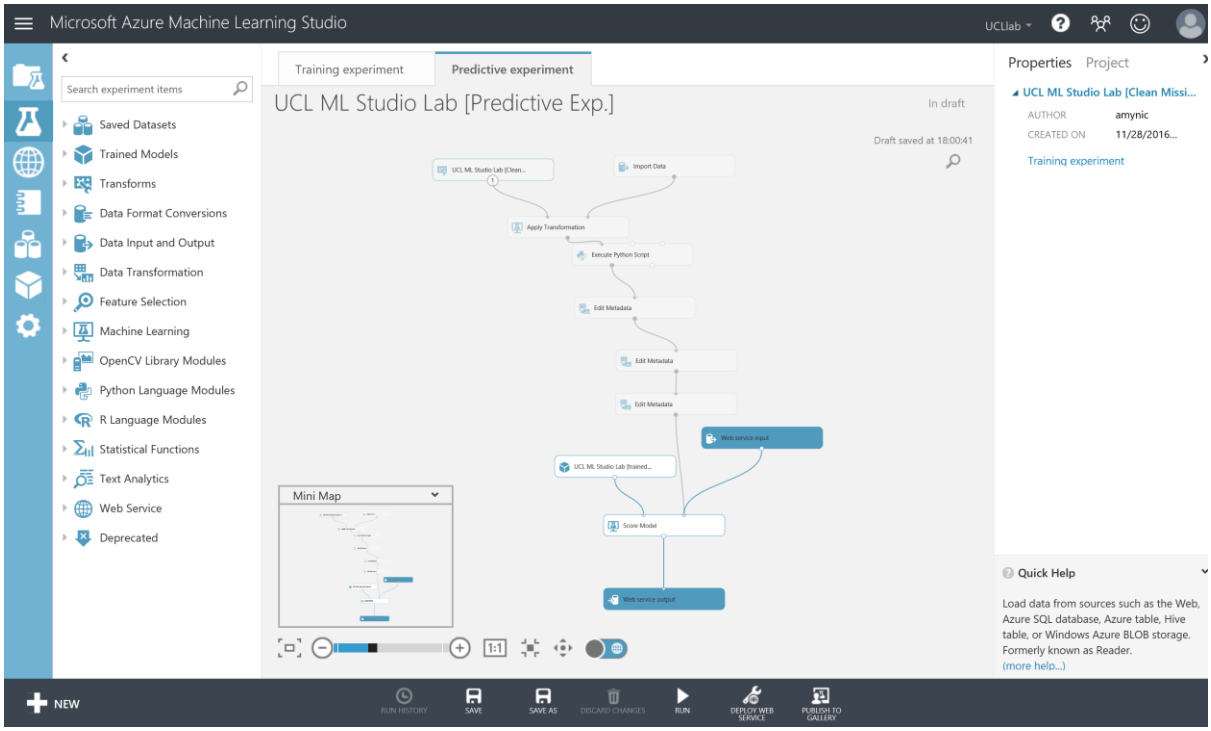
What is a predictive experiment and why do we need it? The purpose of the predictive experiment is to use your trained model to score new data, with the goal of eventually becoming operationalized as an Azure Web service. This conversion is done for you through the following steps:

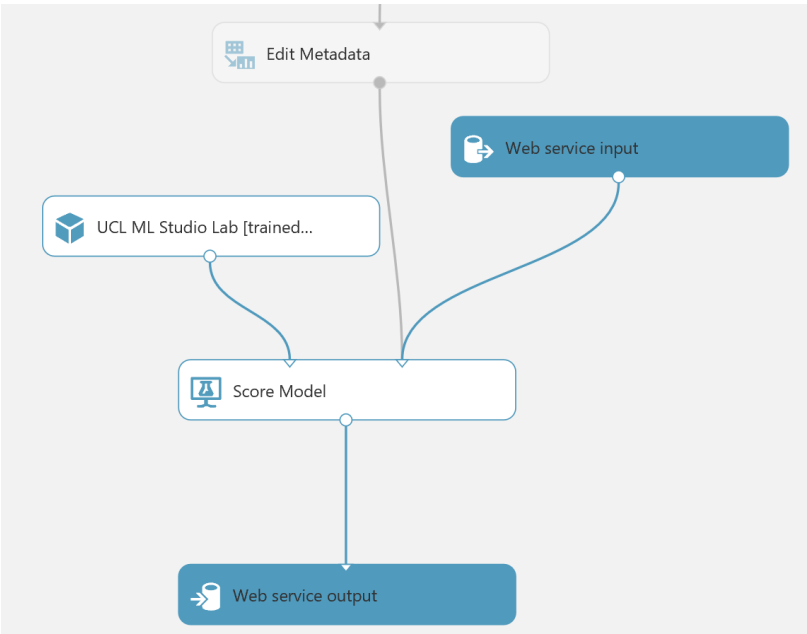
- Convert the set of modules used for training into a single module and save it as a trained model
- Eliminate any extraneous modules not related to scoring
- Add input and output ports that the eventual Web service will use

The added input and output ports, "Web service input" and "Web service output", respectively represent the data format that will flow into and out of the web service we are creating.

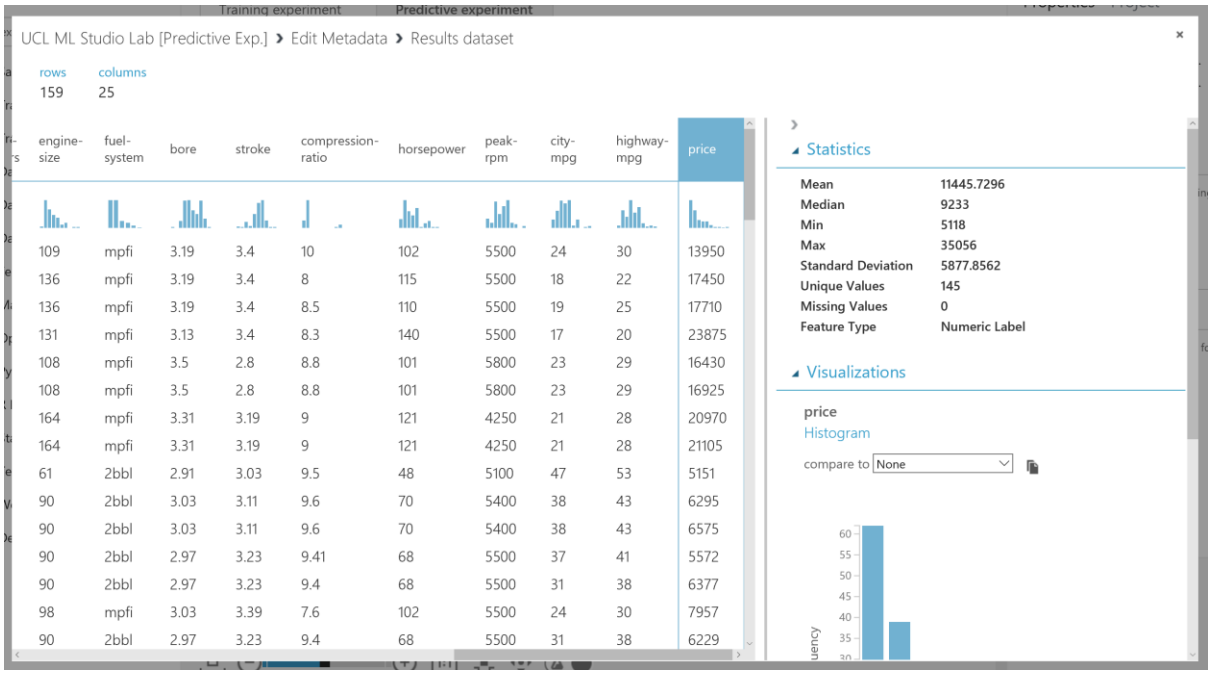
While the wizard has done a basic job of placing where it thinks the input and outputs on our predictive experiment are, it is not perfect. We will need to think about the usage of our API and the data we will receive from an application for example. In our case we will assume we receive only the data we need to query the API and that the data has been input checked before querying the API. Therefore, the scoring experiment does not need to perform all of the data transformation steps we performed in the training experiment to get the data in a format to train.

Move the 'Web Service Input' module down the experiment and connect to the input of the 'Score Model' module. Your experiment should now look something like below:





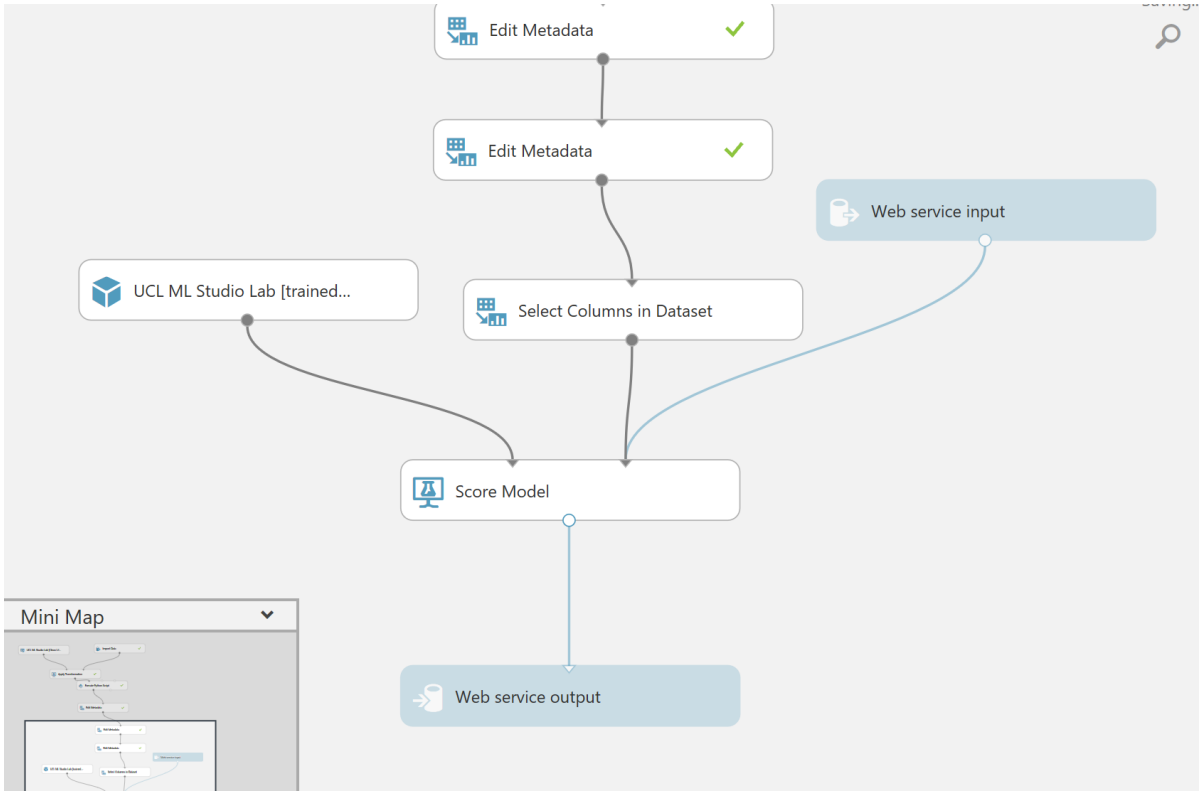
The data flowing through the grey path, from Edit Metadata to Score Model, in the above diagram, now contains the schema of the input to our web service. Run this experiment and then right click and visualise the output of the Edit Metadata module.

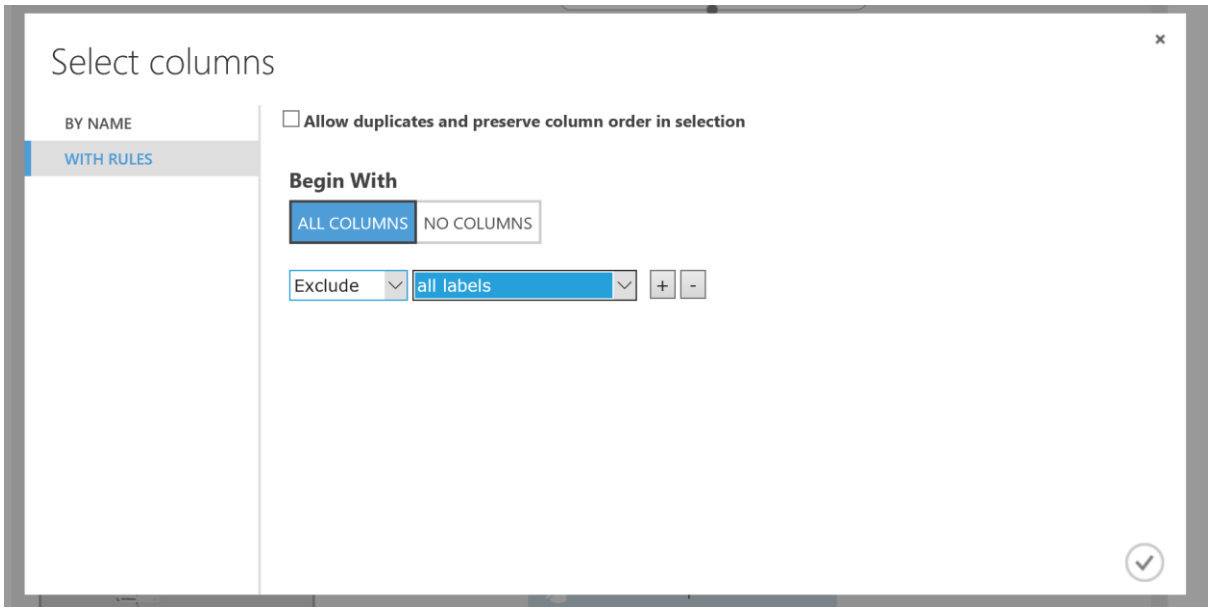


All the columns look fine, it is the right data and the data is transformed as we expect from the training experiment, however note: the data contains our label (price) which is what we are trying to predict. While this is valid for training we shouldn't have it here for scoring – as this is the outcome/value we want to pass back as an output from the web service, moving from training a supervised machine learning model to scoring/deploying a supervised machine learning model.

So we can eliminate this column by adding a 'Select columns in a Dataset' module as shown below that excludes all labels from the experiment:

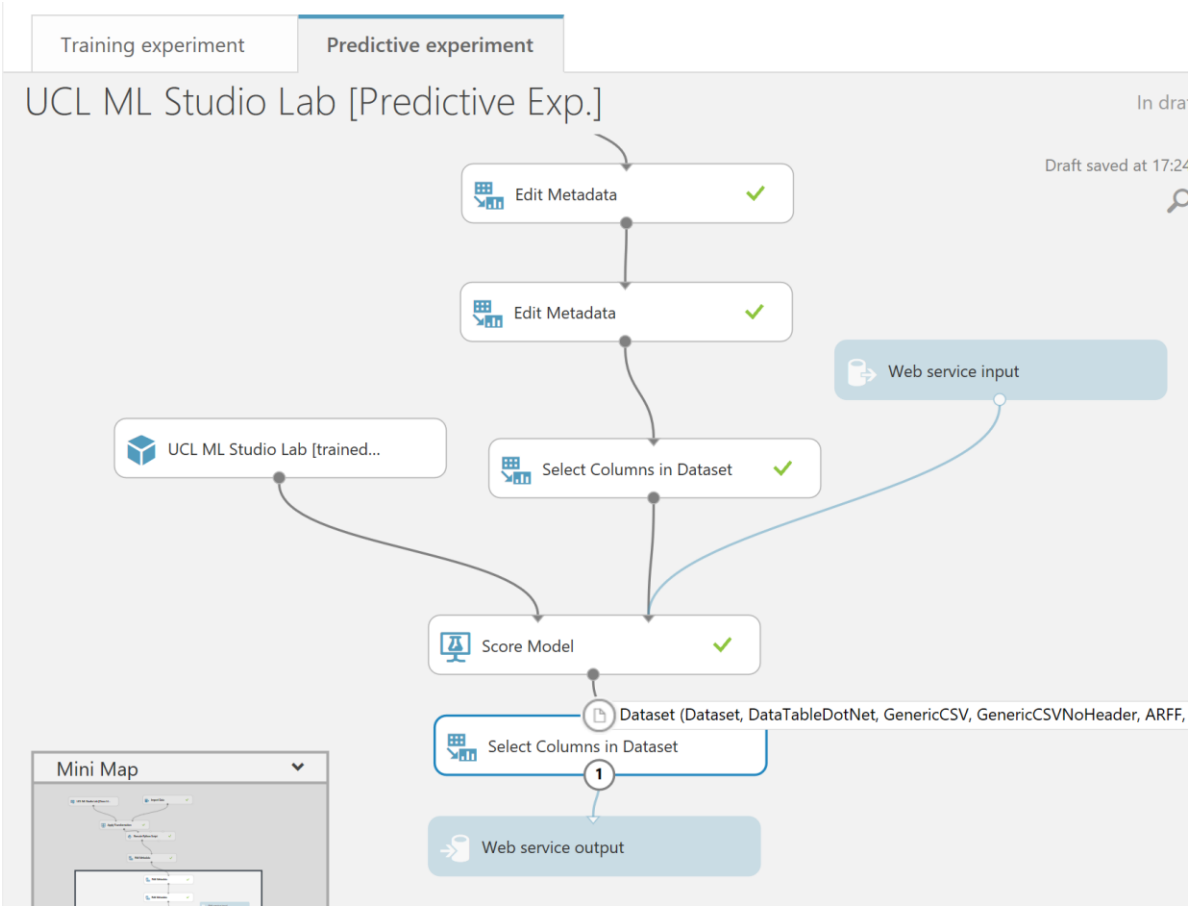
*** remember this will mean that the schema for the wbe service input going into the score model module only contains your features(dimensions) ***

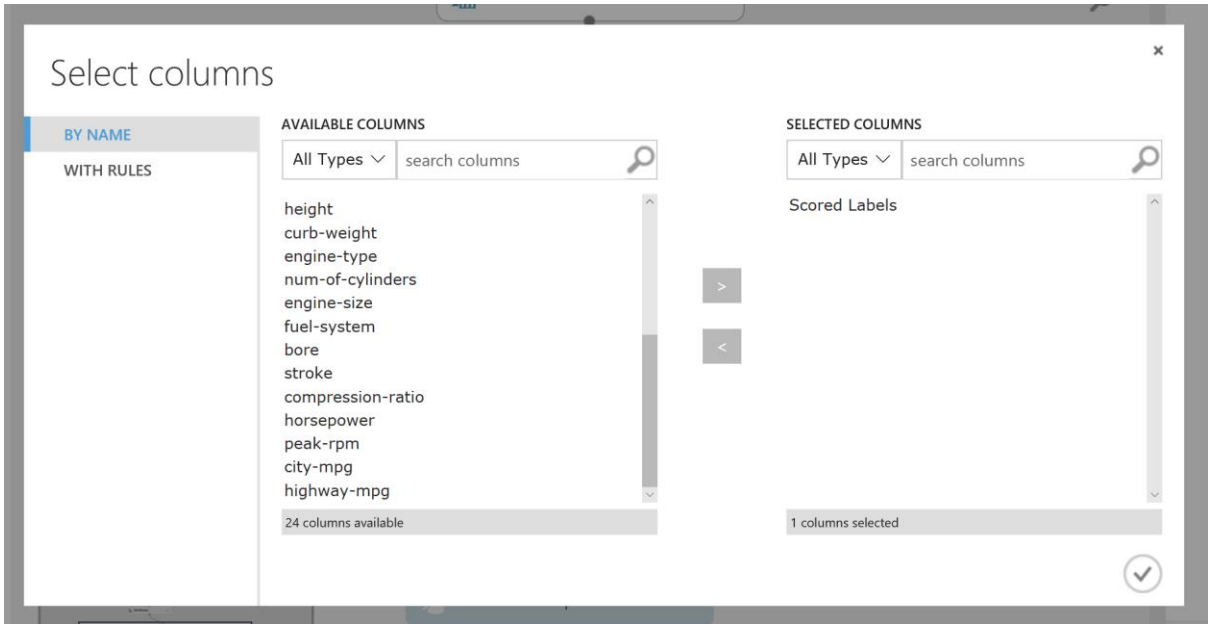




If we now think about what fields we want to return to the application or web site that will call our web service we are creating, then all we technically need is the scored label (the prediction) for the regression experiment. This makes an assumption that we keep hold of nay data needed on the front end to understand the information/prediction that will be passed back from the web service.

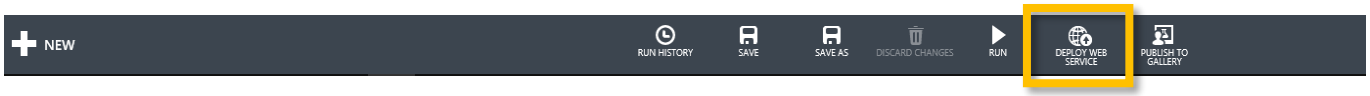
So we should add another 'Select columns in a Dataset' module as shown to just return that field:



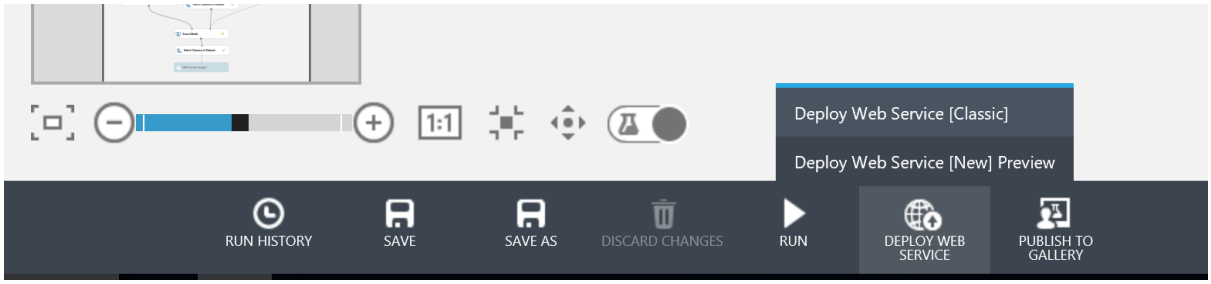


We must now **Run** this predictive experiment as this allows ML to validate the predictive experiment before we can publish it as web service, choose the **RUN option** from the bottom toolbar.

After a successful run we'll see that the deploy web service icon is available on the bottom toolbar

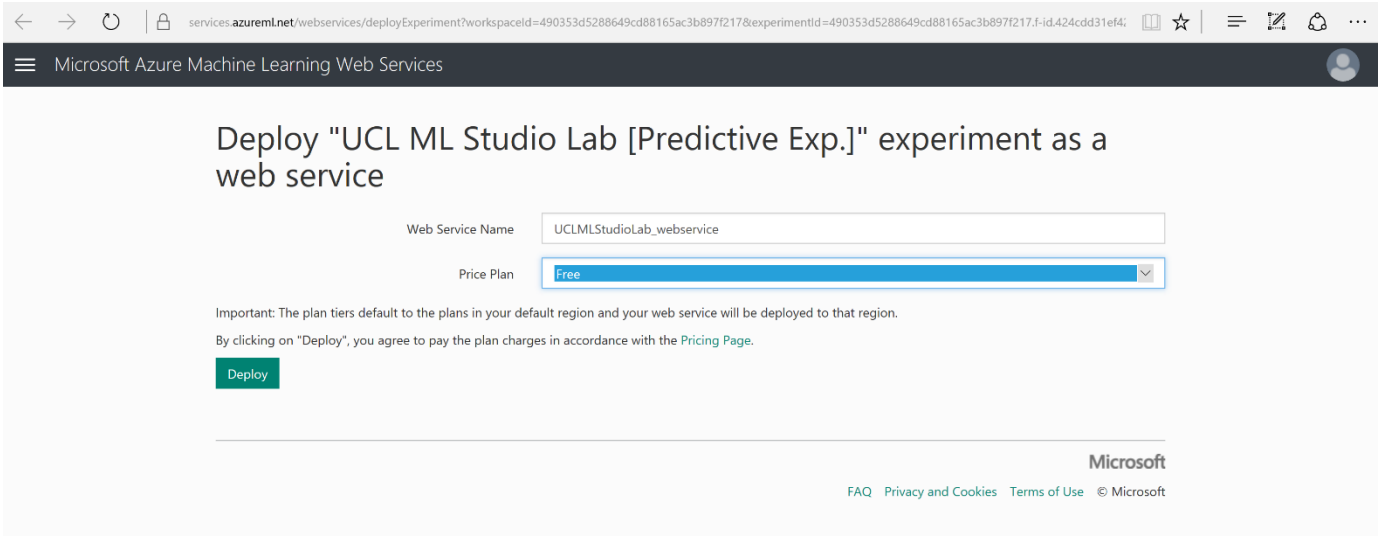


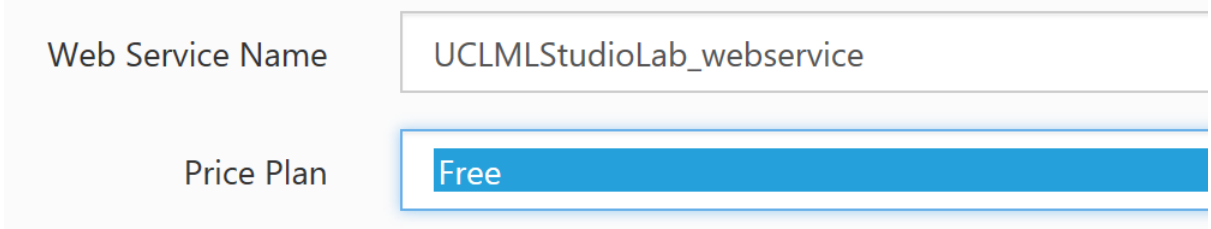
All we need to do is click on it and select deploy web service (new).



After a few seconds we'll be taken to a new tab in the browser and into the web service management portal for azure machine learning.

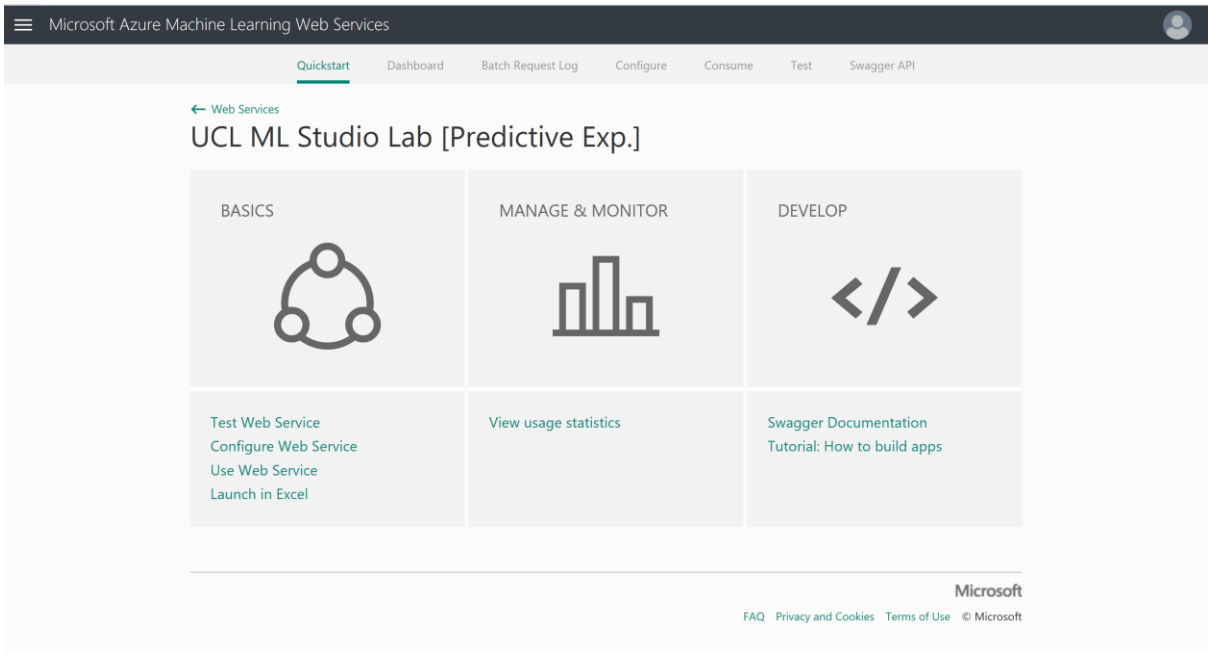
In here create the name of your web service (see example below) and choose the free pricing tier.



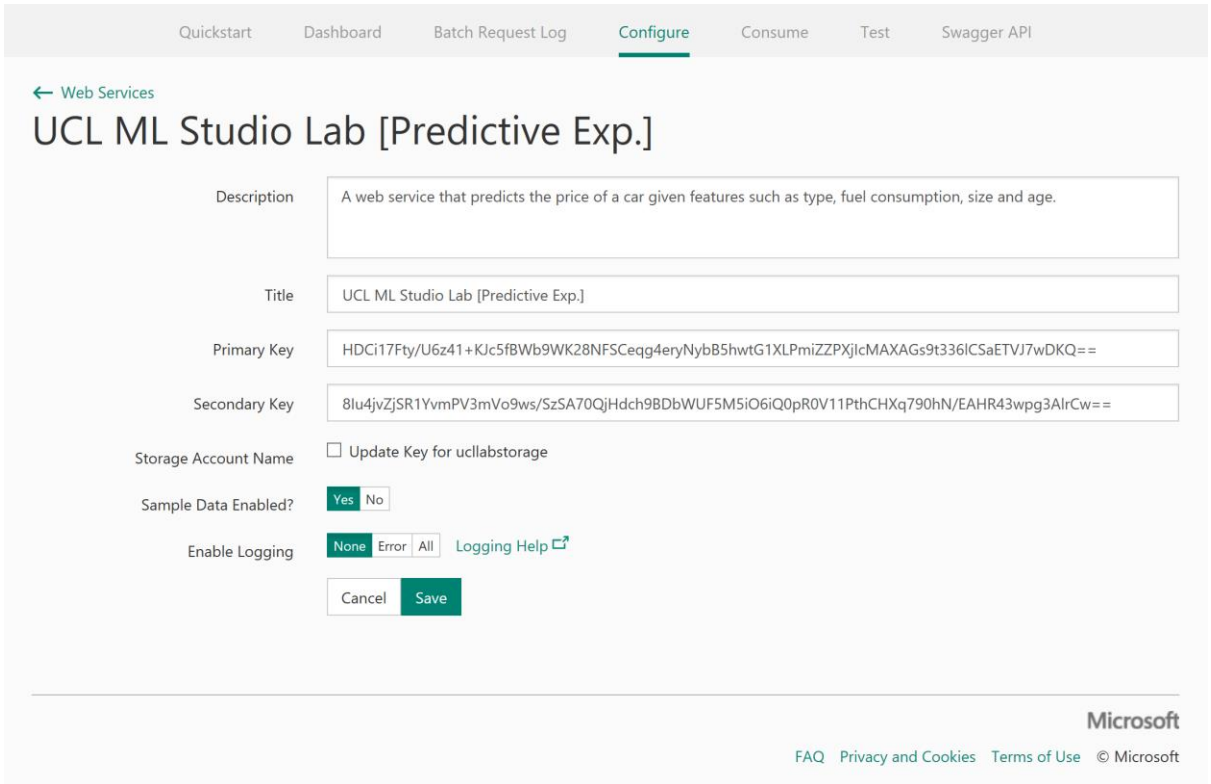


Click the deploy option and wait for the web service management portal to load with your web service information:

*** Note: if there is an initial swagger API error, wait a couple of minutes and then refresh the page***



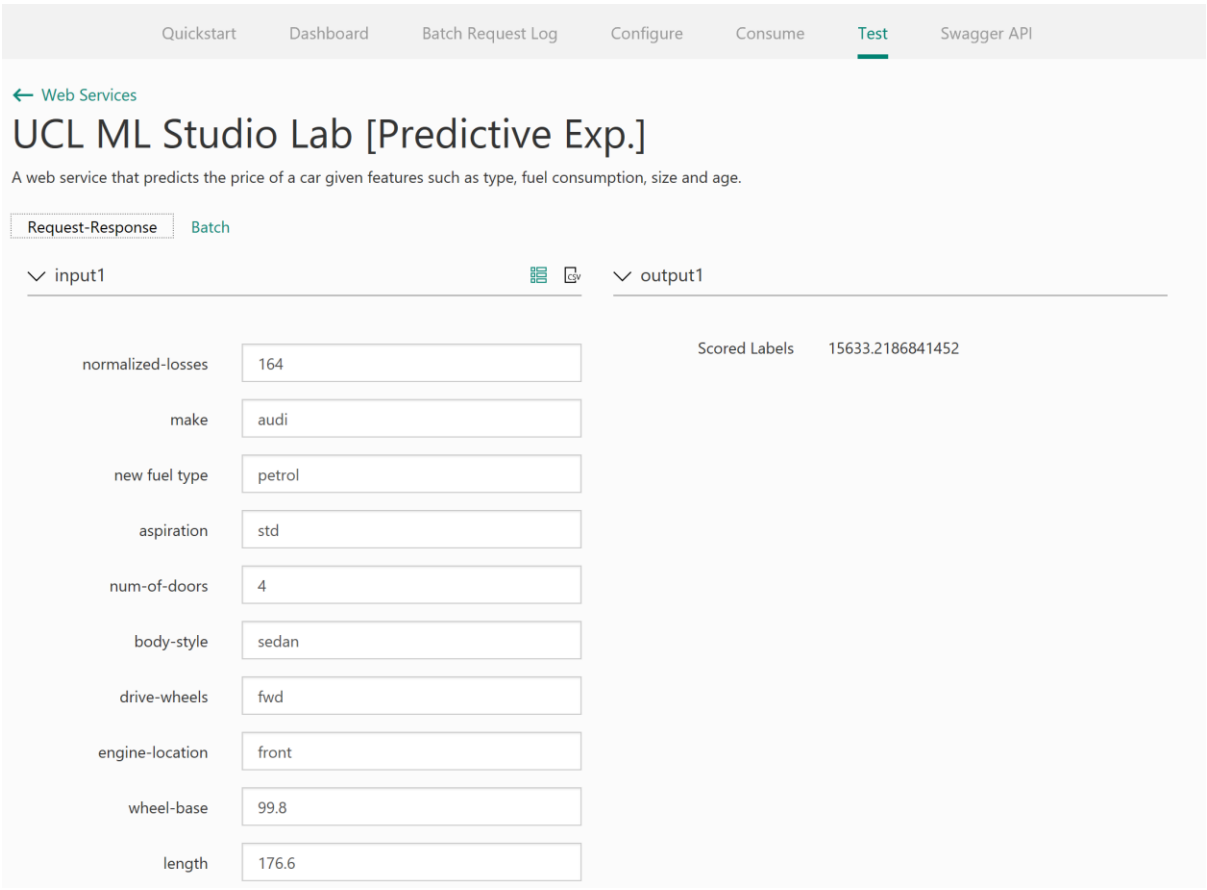
Next we need to configure our web service in order to test it. Click on 'Configure Web Service' under the BASICS section. Once open give your web service a description and make sure 'Sample Data Enabled?' check boxes are set to yes. Then save these changes.



You should receive a message stating the update was successful:

Now let's test the web service to check our input and output schema. Choose Test in the top toolbar and notice that for Request Response option is selected and our sample data has prepopulated our input data.

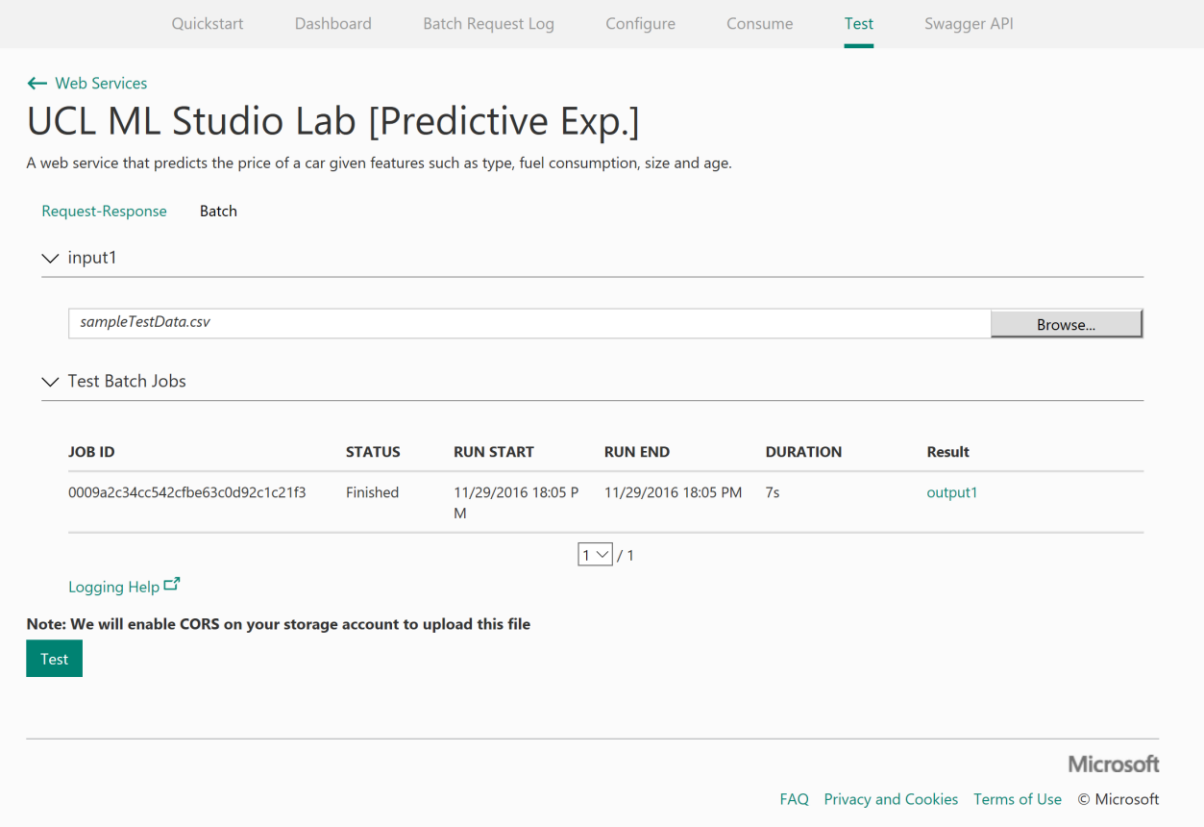
Check the input schema is as expected (not containing the label, price) and select the Test Request Response button to return your prediction on the right (output schema of web service). In the screenshot below, the Audi is predicted to cost around £15600.



You can also test batch files of data to receive multiple predictions back. Select the 'Batch' Option for testing in the test web services panel

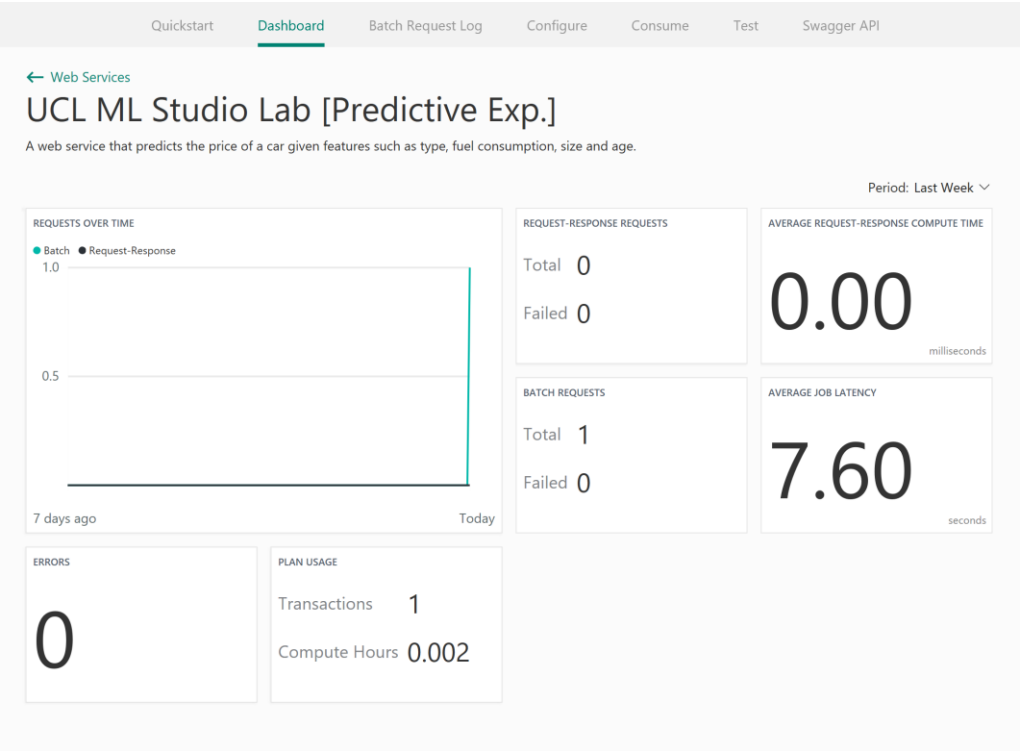
Now upload the 'sampleTestData.csv' available on the GitHub repository from your local machine and click the Test button.

Notice the job gets queued, tells you the status, start run time, end run time and duration of the job.

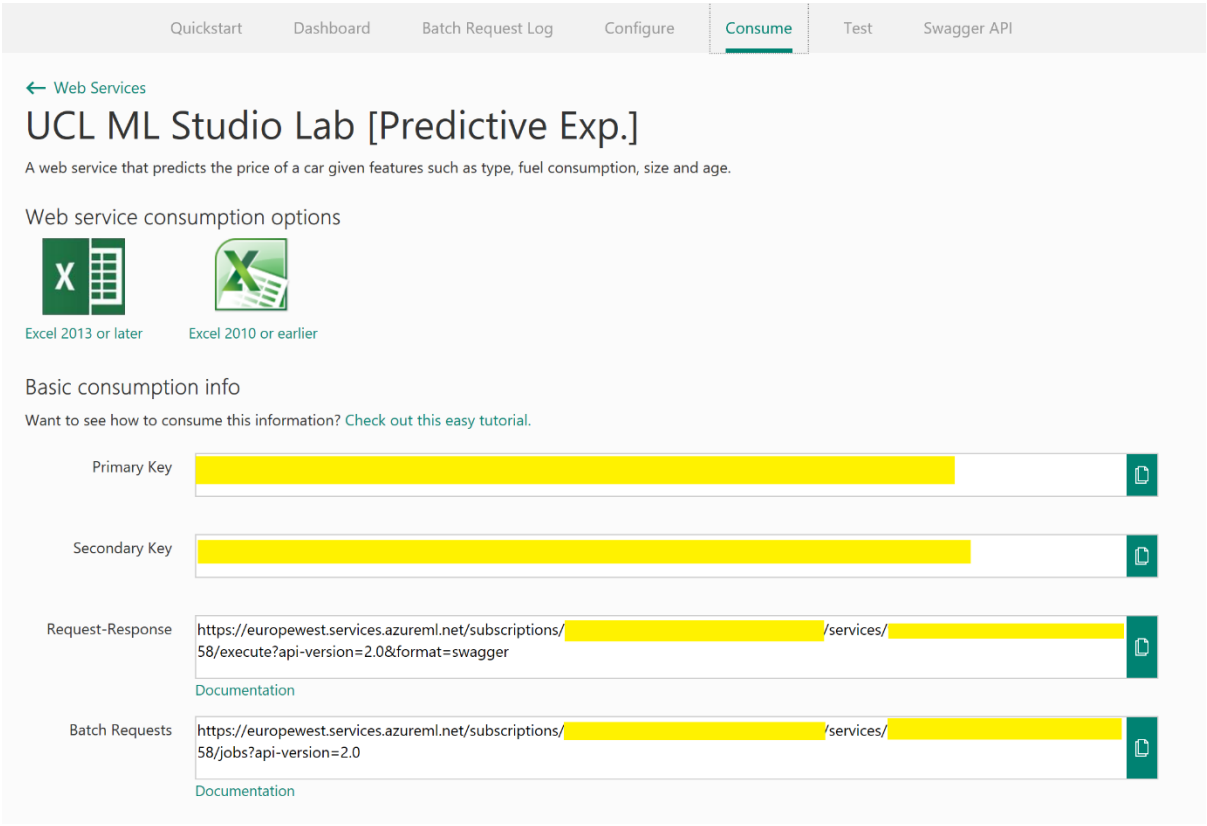


Select the output1 link and download the CSV, once downloaded open the CSV file and find the scored labels output for each of the cars.

Other features to quickly note. You have a '**Dashboard**' of usage of you web service API on the dashboard tab:



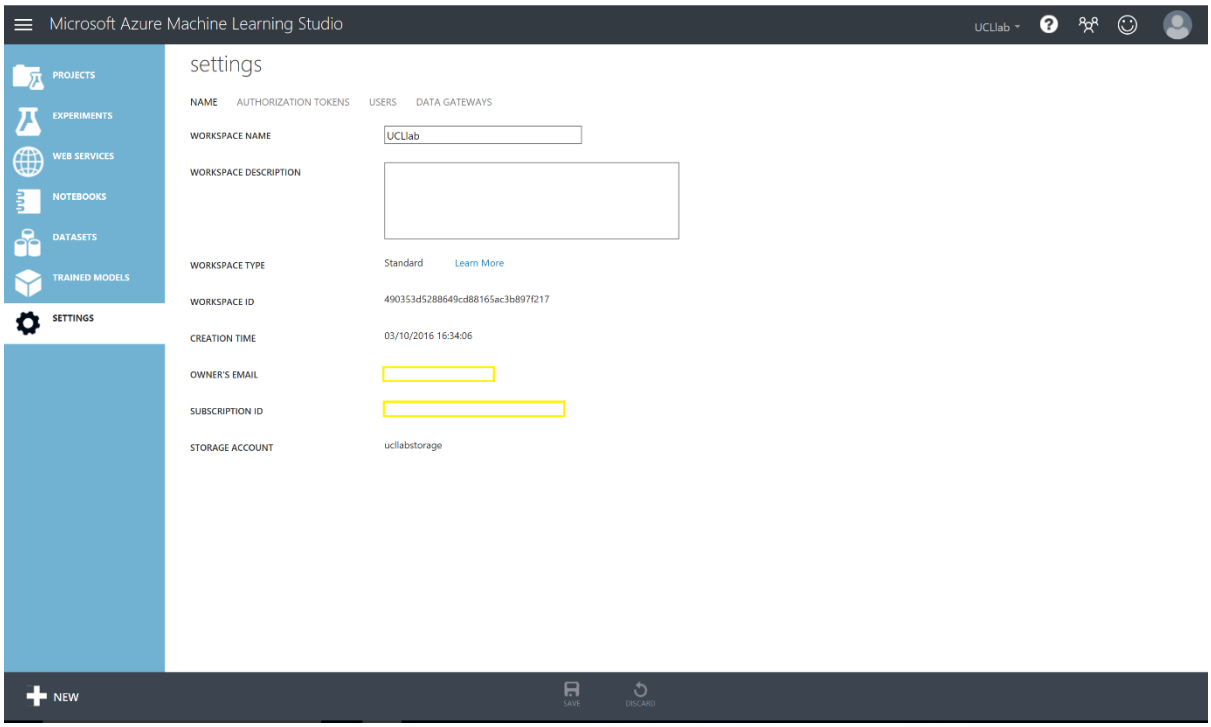
Finally, the '**Consume**' tab provides your web service API keys, excel spreadsheets with your model preloaded and also sample code in C#, Python, Python 3+ and R to implement into your applications.



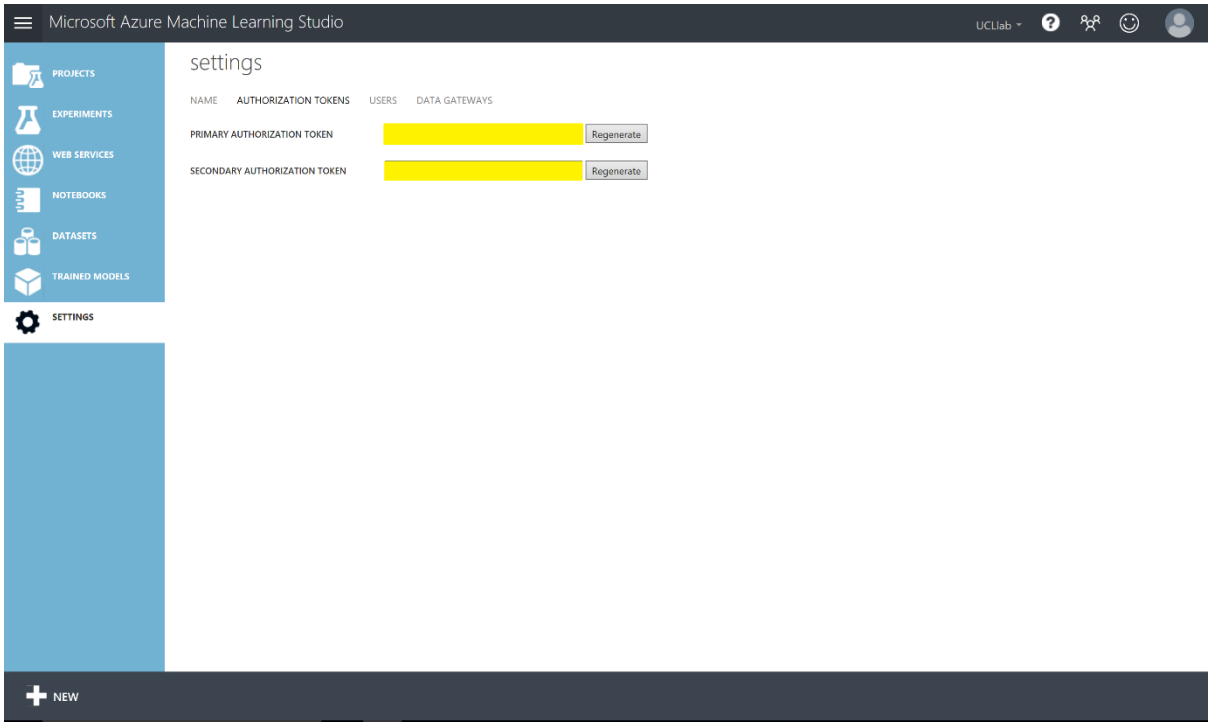
Publishing a Web Service - Jupyter

You can also publish web services directly from Jupyter Notebooks as long as you have an Azure ML Workspace ID and Authorization Token, we can get these from your Azure ML workspace.

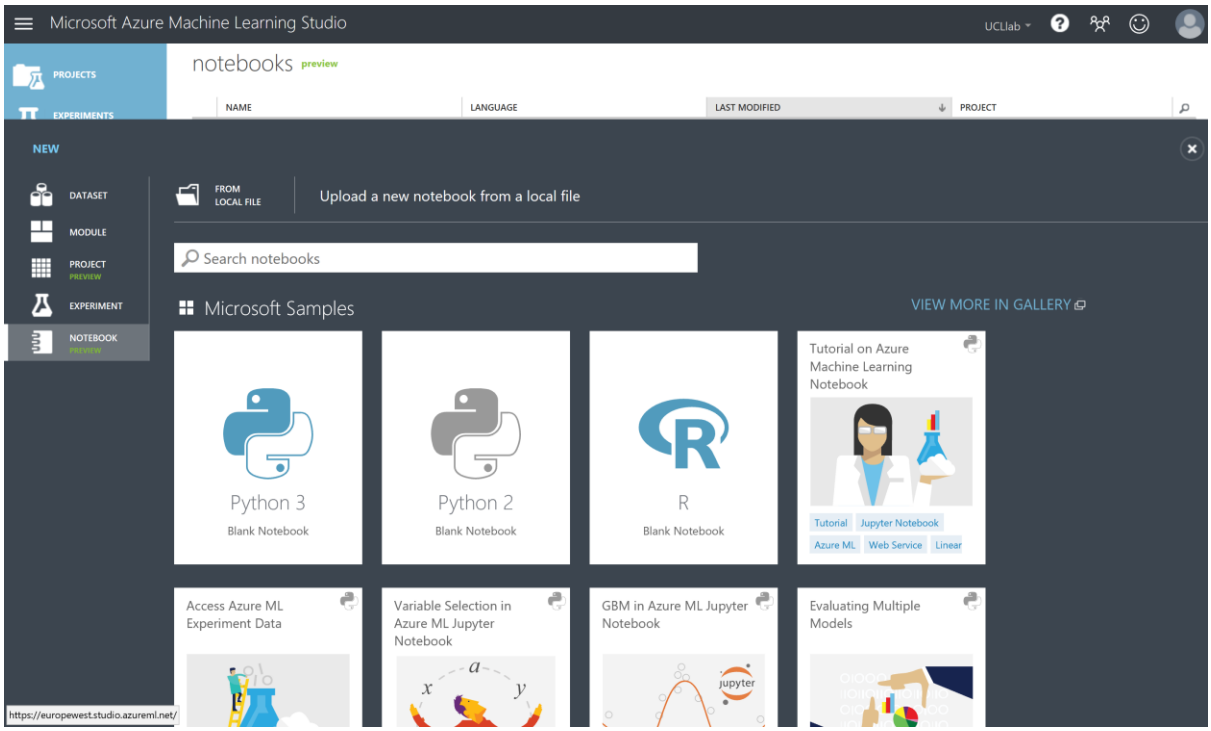
Go back to the Azure ML Studio and enter the settings tab



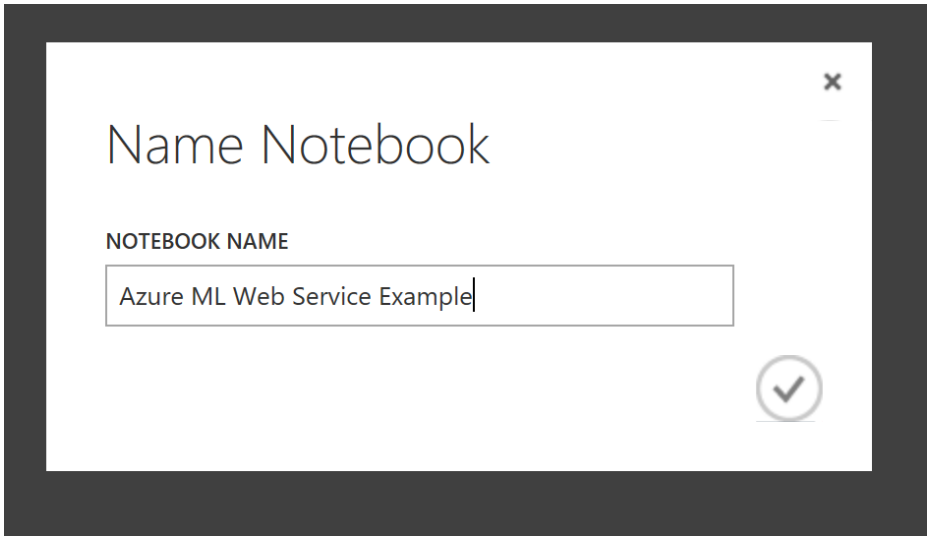
Take a note of your workspace ID from here and then enter the Authorisation Token header and take note of your Primary Authorisation token. You will need these later on in the lab.



Now create a new Python 2 notebook from the New menu in Azure ML Studio



Give it a name like below: Azure ML Web Service Example



Because we are inside the Azure ML Studio notebooks **we will not need to install any packages**. However, if you are using other versions of Jupyter Python notebooks you will need to download the Azure ML SDK here: https://pypi.python.org/pypi/azureml/0.1.1 and use a pip install command to be able to setup web services

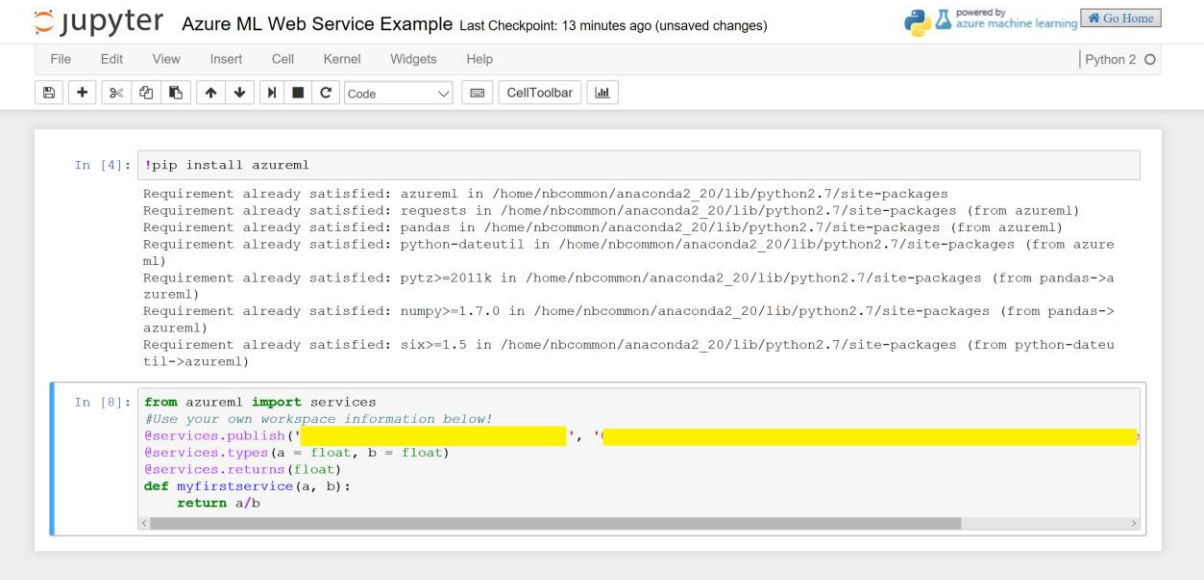
```
In [4]: !pip install azureml
        Requirement already satisfied: azureml in /home/nbcommon/anaconda2_20/lib/python2.7/site-packages
        Requirement already satisfied: requests in /home/nbcommon/anaconda2_20/lib/python2.7/site-packages (from azureml)
        Requirement already satisfied: pandas in /home/nbcommon/anaconda2_20/lib/python2.7/site-packages (from azureml)
        Requirement already satisfied: python-dateutil in /home/nbcommon/anaconda2_20/lib/python2.7/site-packages (from azure
        ml)
        Requirement already satisfied: pytz>=2011k in /home/nbcommon/anaconda2_20/lib/python2.7/site-packages (from pandas->a
        zureml)
        Requirement already satisfied: numpy>=1.7.0 in /home/nbcommon/anaconda2_20/lib/python2.7/site-packages (from pandas->
        azureml)
        Requirement already satisfied: six>=1.5 in /home/nbcommon/anaconda2_20/lib/python2.7/site-packages (from python-dateu
        til->azureml)
```

Next we need to import the services package and your credentials from earlier in the lab (workspace ID and authorisation primary token).

Enter the code below and substitute in your keys:

```
from azureml import services
#Use your own workspace information below!
@services.publish('<you workspace id>', '<your auth. token>')
@services.types(a = float, b = float)
@services.returns(float)
def myfirstservice(a, b):
    return a / b
```

this code creates a web service which returns a float value from the function definition 'myfirstservice'. Run this code in the notebook (ctrl + enter).



Once it is published you can run many commands over your service for example a few are below:

- Get Directory

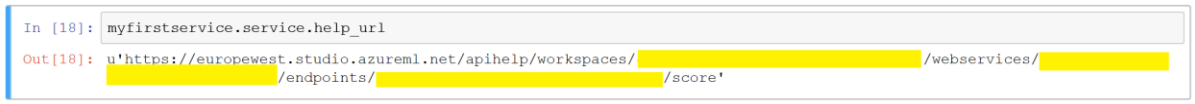
  ```
  Dir(myfirstservice)
  ```

- Post URL for the web service

  ```
  Myfirstservice.service.url
  ```

- Web service help page

  ```
  Myfirstservice.service.help_url
  ```



To call you web service function inside notebooks you can simply use the function definition, service, then followed by parameters *(note the first time you run the service it may take up to 30 secs to connect)*

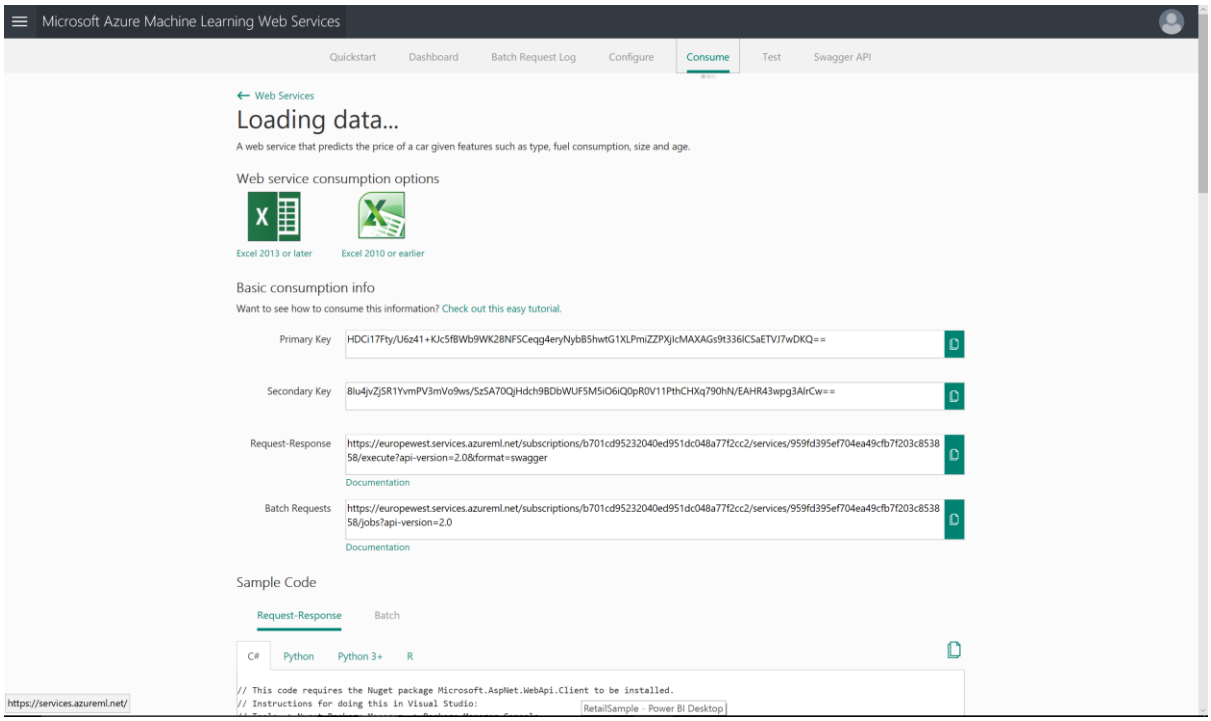
```
Myfirstservice.service(4,2)
```

In this section we saw how we could use the Azure ML Python SDK to create and call python functions as web services hosted in Azure ML. This is a very simple example of a web service – however you can see how you could extend this to create more complex functions that you can call via an API

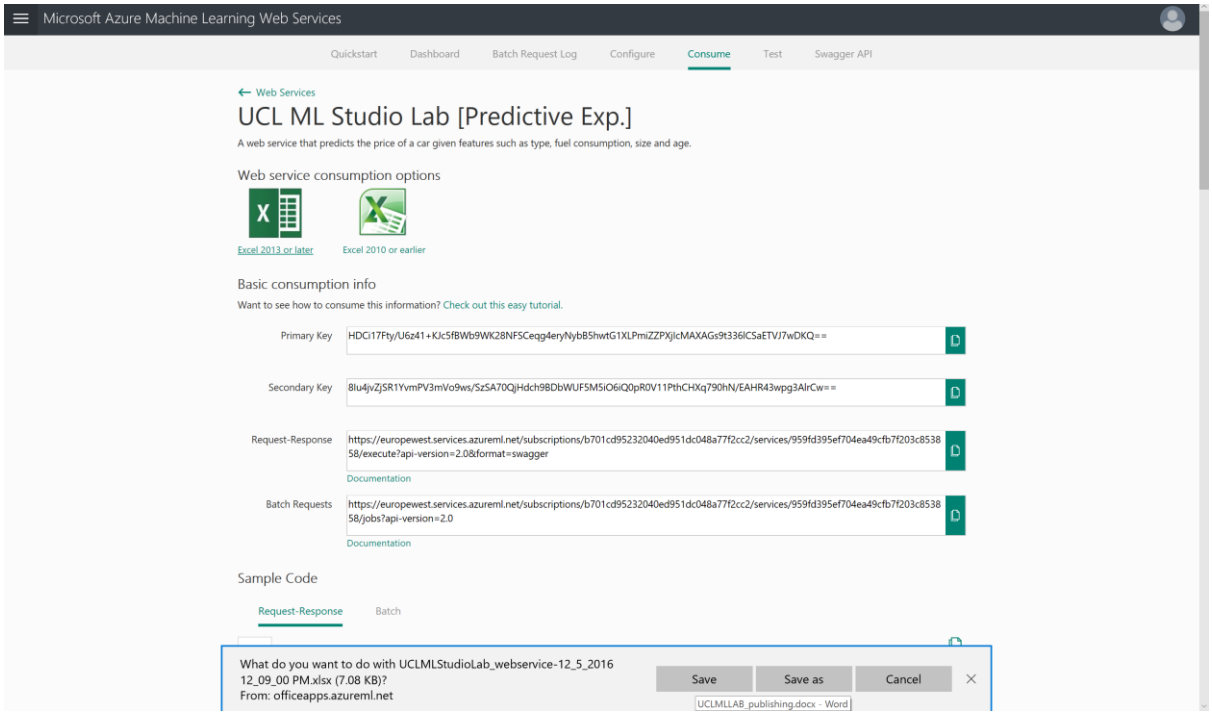
Consuming a Web Service – Excel

Excel is a very popular tool within industry still to look into data and predictions that you create. In this section we will look at how you can leverage excel to call you Azure ML Web service.

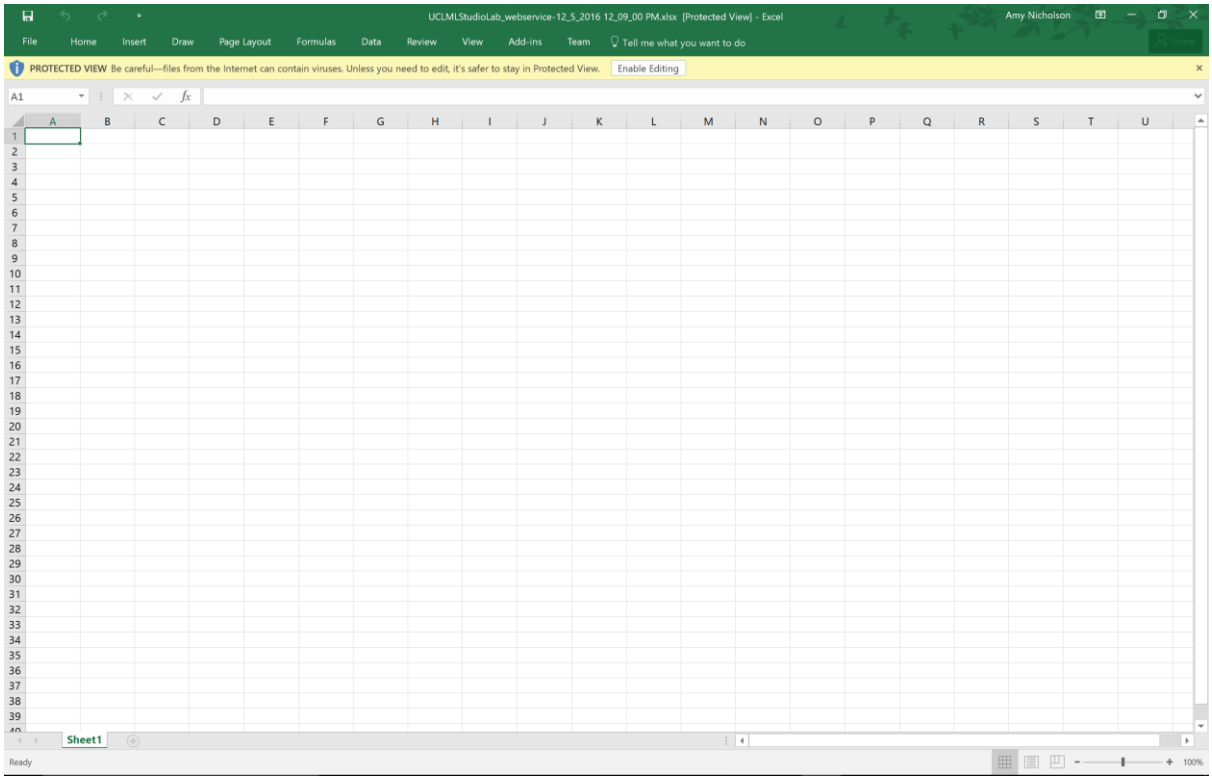
Going back to the Web service you created earlier on the http://services.azureml.net/ open the dashboard for your web service on the consumer tab and you should see a similar display as below:



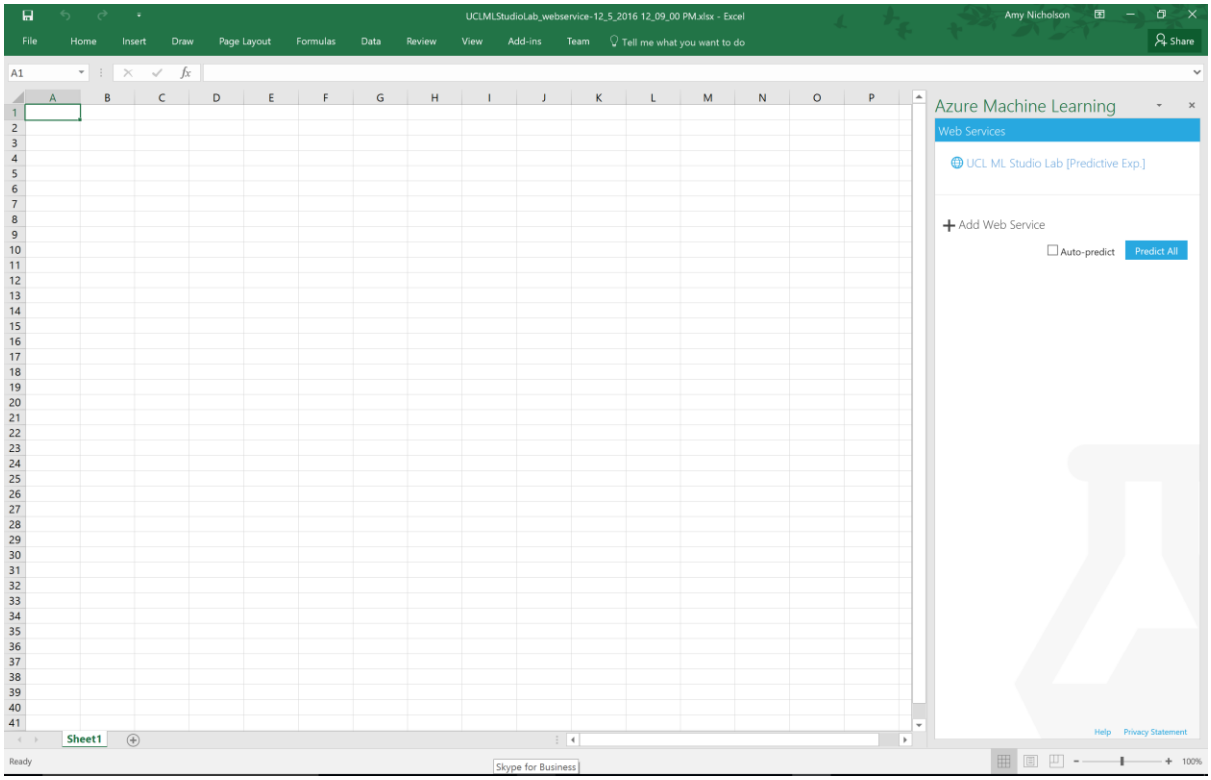
Select the excel version you have available (this tutorial will select the Excel 2013 or later option) and save the download file



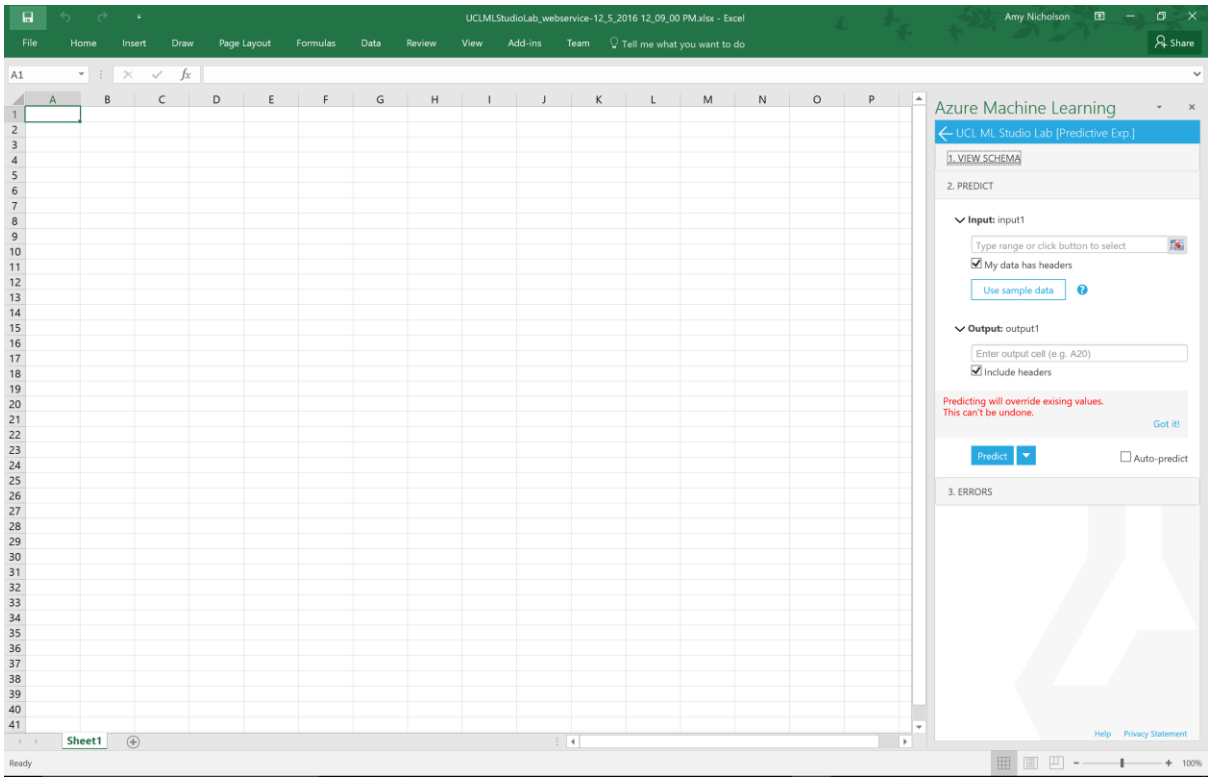
Once Excel has opened, choose enable editing in the top yellow bar



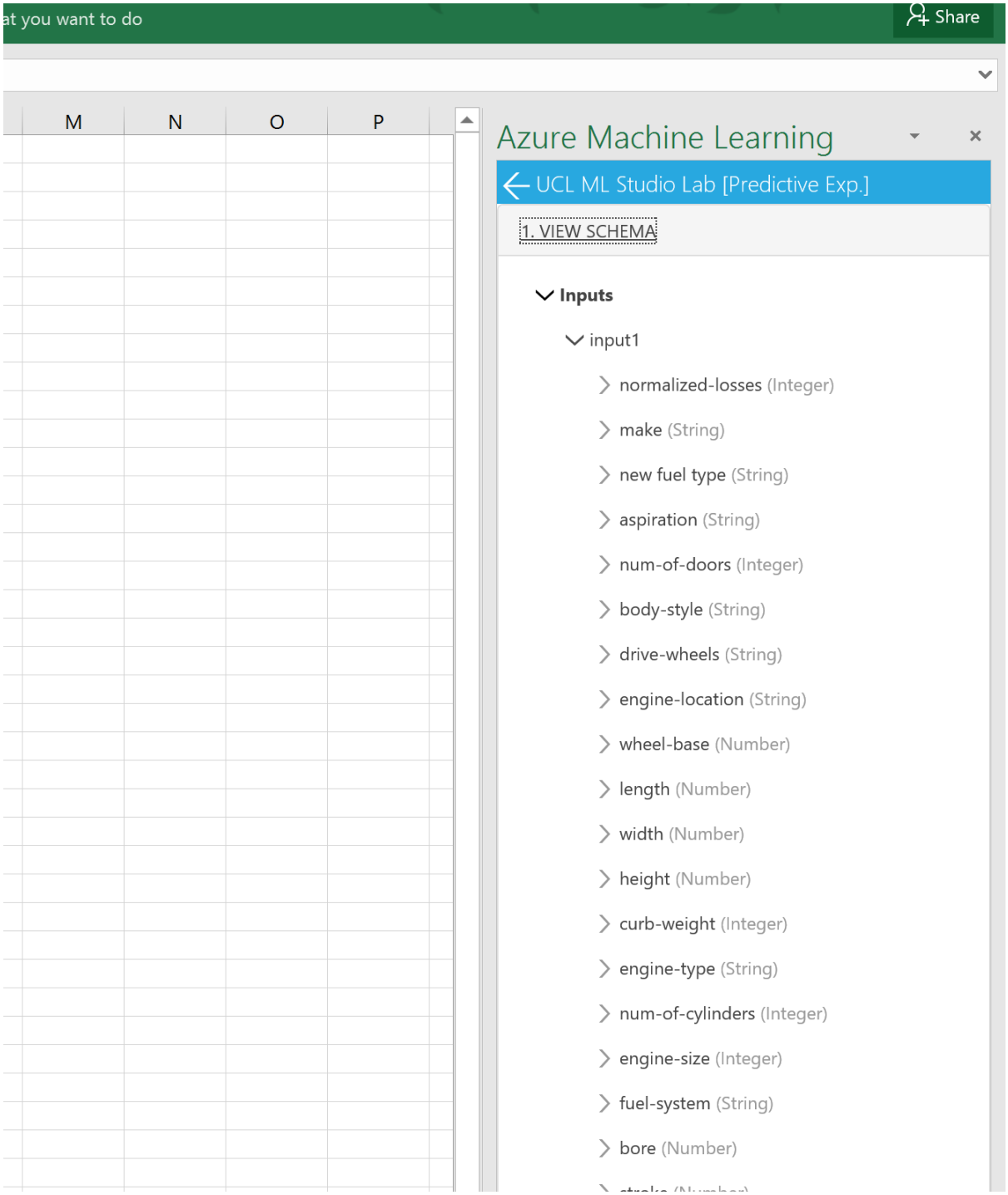
This should then load on the right of the excel an office add in. This add should also automatically connect to the web service you have created, for example in the case below "UCL ML Studio Lab [Predictive Exp]"



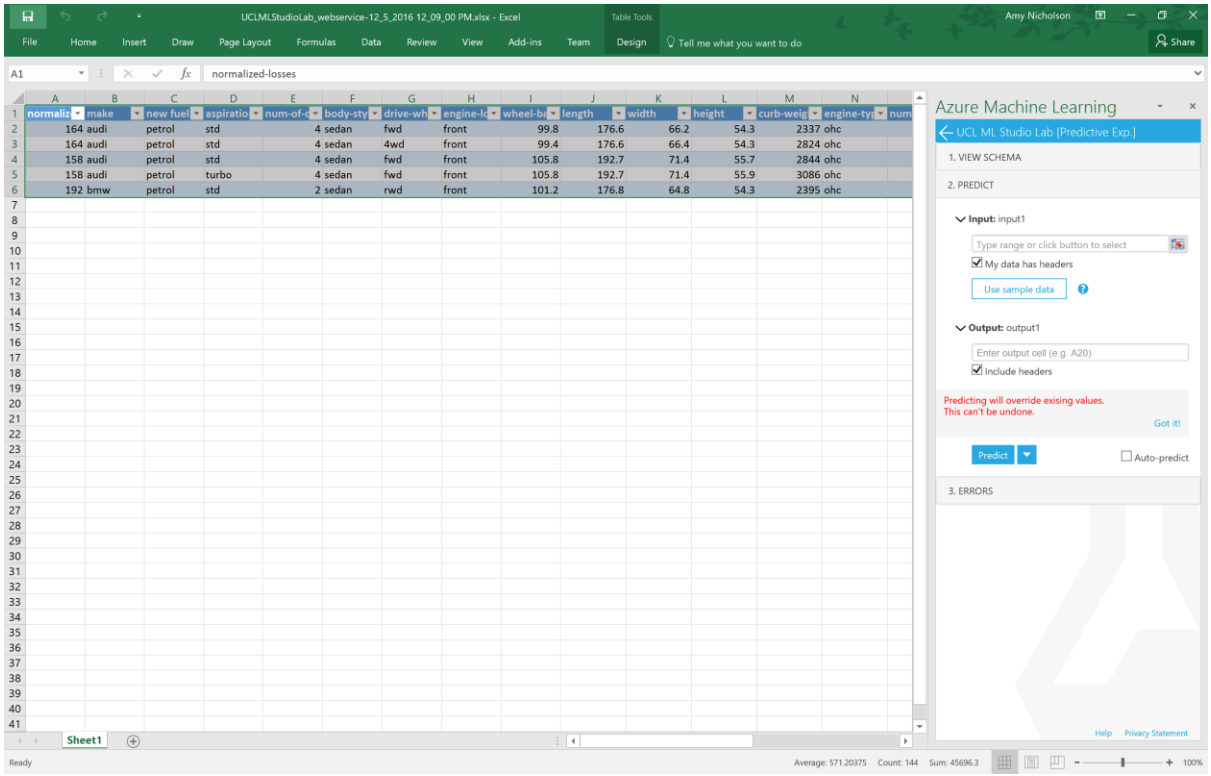
Select the service or choose to add a new web service. Note, you can add any web service you have created by providing the scoring web service URL and your API keys.



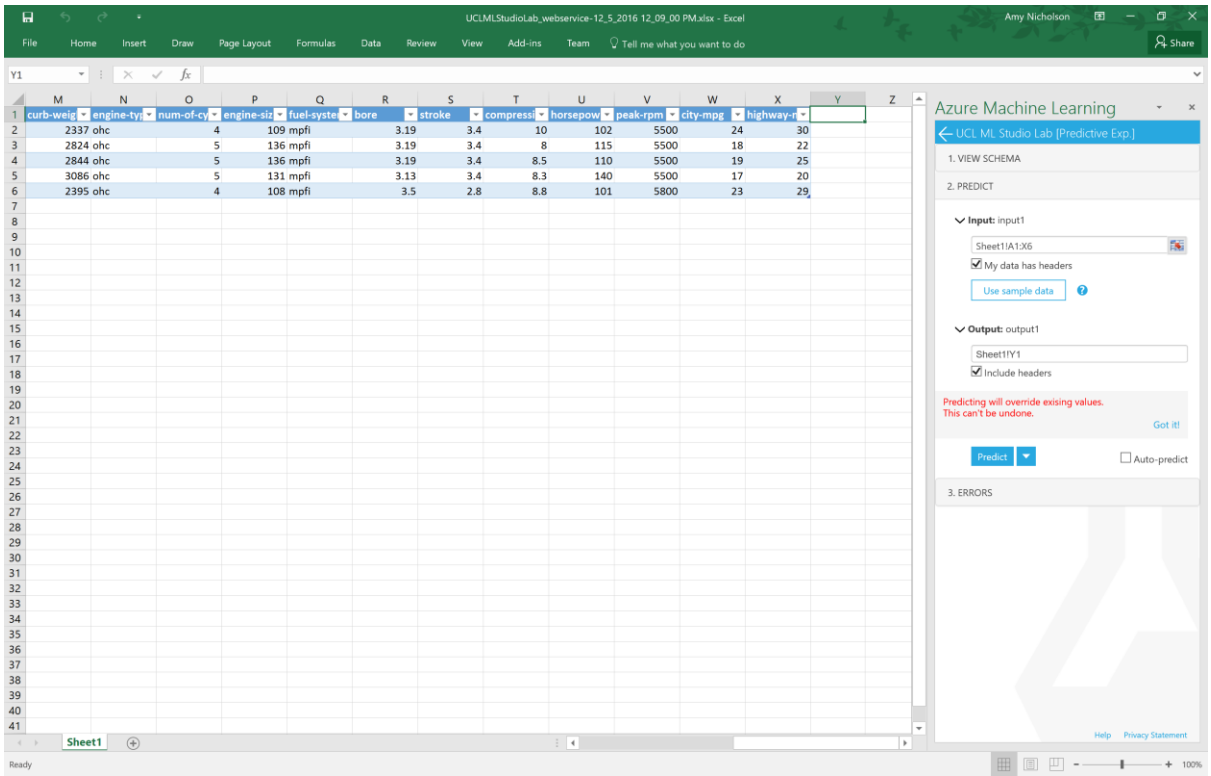
You can view the schema to check it's the correct experiment by selecting the "1.View Schema" section.



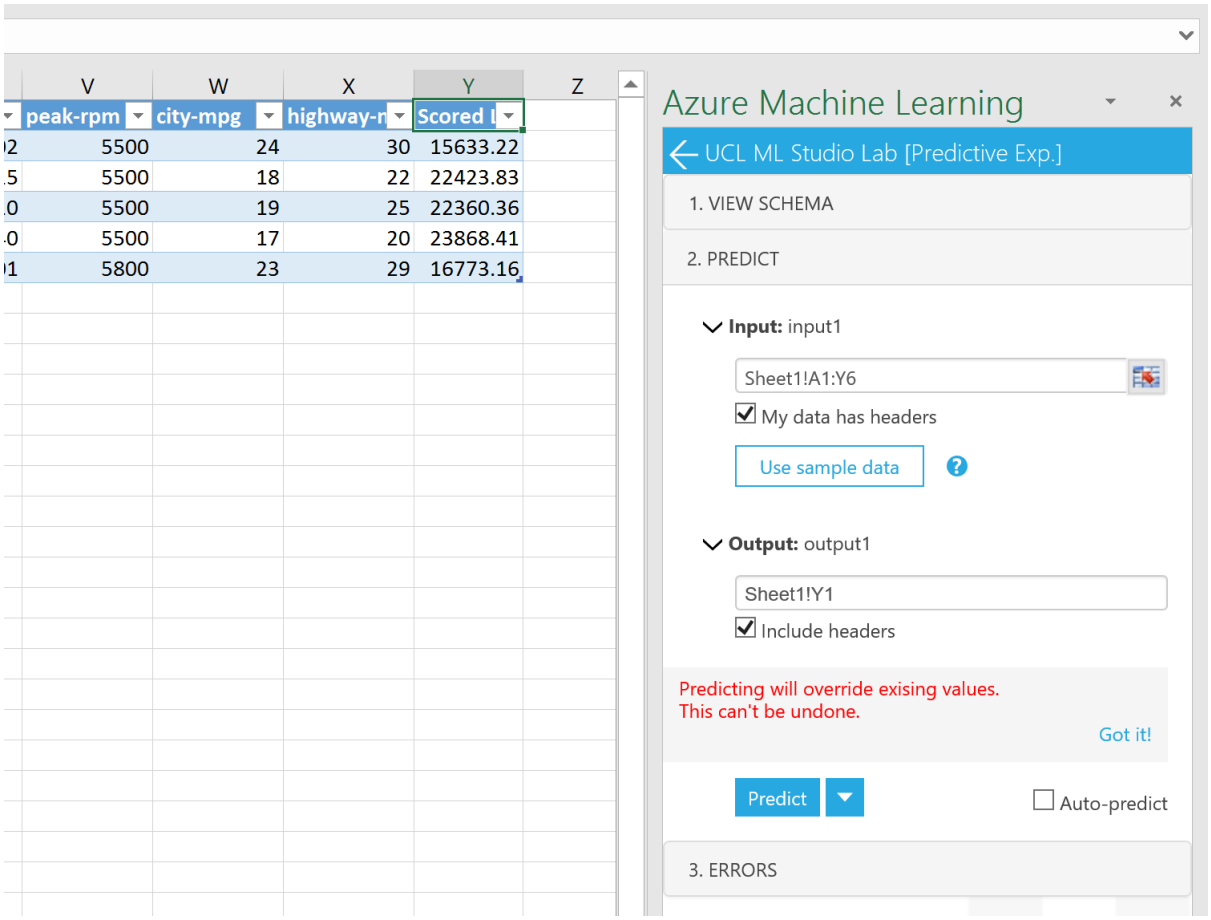
Now to start predicting values using our API. Instead of us creating the dataset you can select the "sample data" button and this will pre-populate the spreadsheet with some data and the correct schema for you to query against. If you had your own data in the spreadsheet already you can use the input range box to select this data and query against it, make sure you select the checkbox "my data has header" if you have provided a table format.



For input cells select the sample data and also the output cells – so where should the prediction go in your spreadsheet. In this case I scrolled to the end of the input data and chose cell Y1 for the prediction data to go into

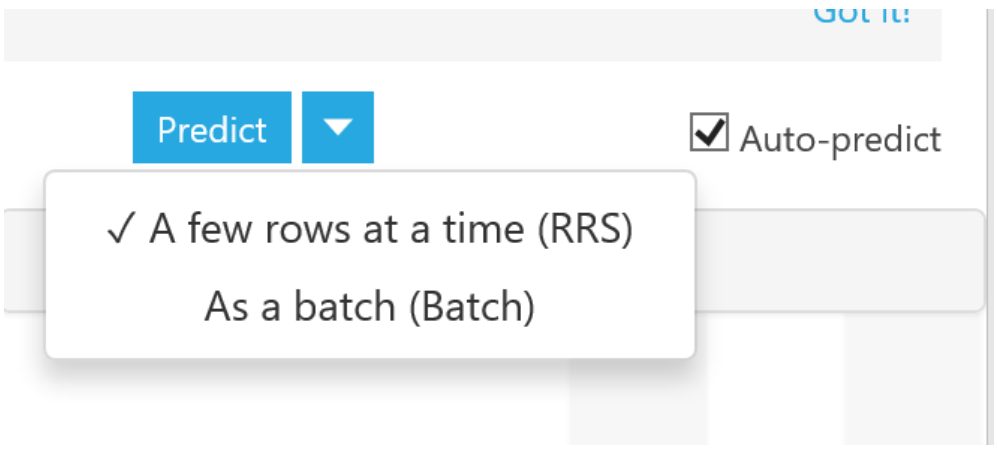


Now click the predict button and see the scored labels (regression values) populate for the new data

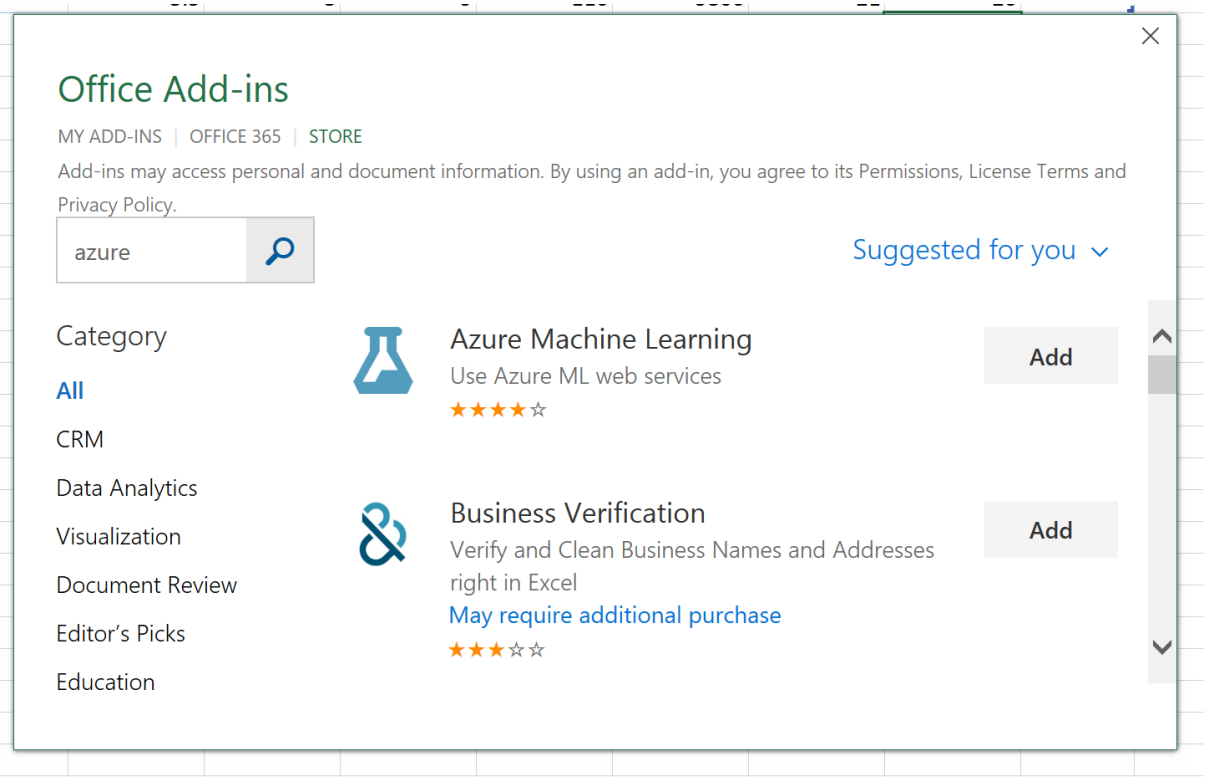


Other features to note:

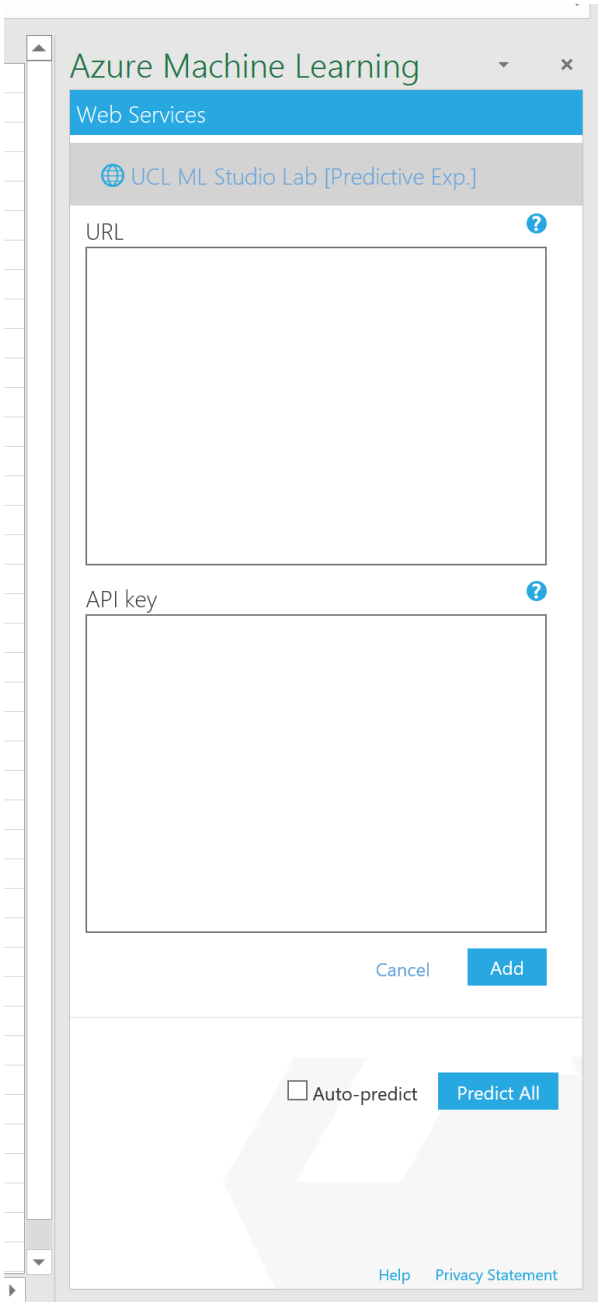
- You can choose the type of prediction you use
    - Request Response – each row of data is processed separately, more real time option
    - Batch – all the rows of data are sent as a batch and returned results as a batch.
- You can check the Auto-predict box and populate new data in the excel and the prediction will automatically be populated



You can add any web service you create to an excel sheet. We accessed it automatically from the web service management portal however the office app is available in the store so you can add it into any excel sheet from 2013 or later



And you can connect any of your Azure ML Web services using the web service URL and API key.

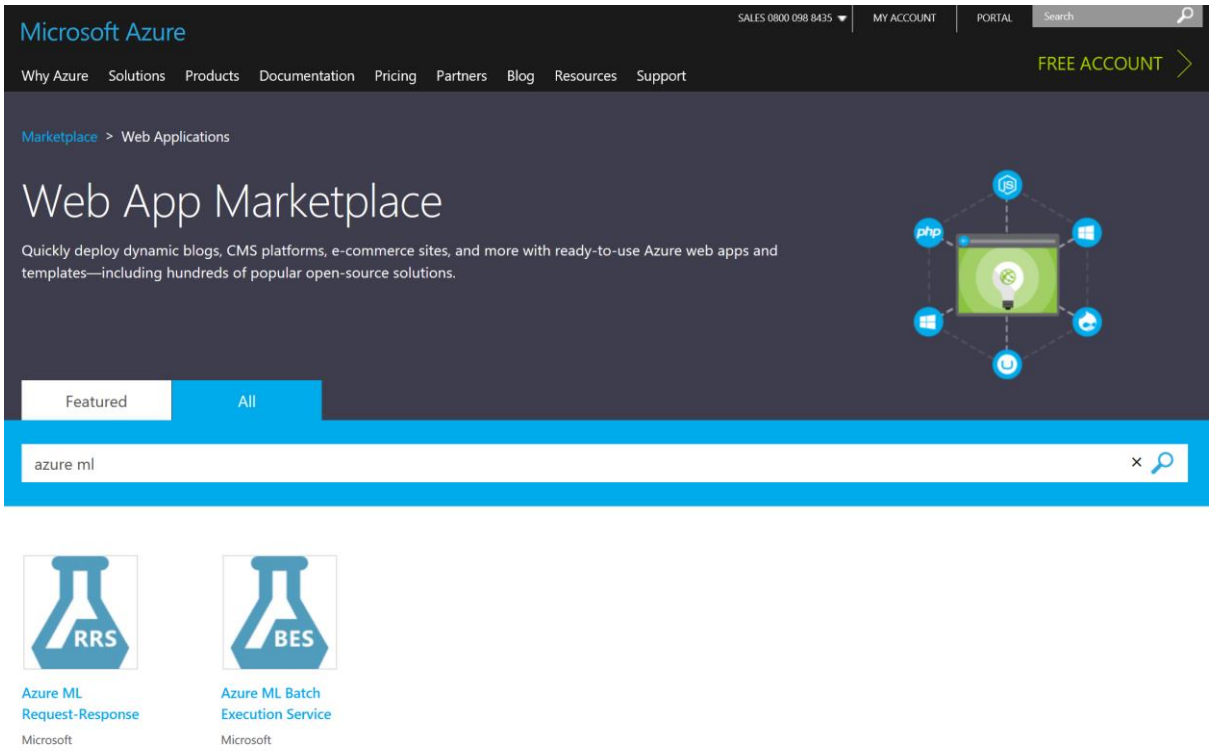

Azure Machine Learning
Web Services
UCL ML Studio Lab [Predictive Exp.]
URL
API key
Cancel
Add
Auto-predict
Predict All
Help
Privacy Statement

Consuming a Web Service - App

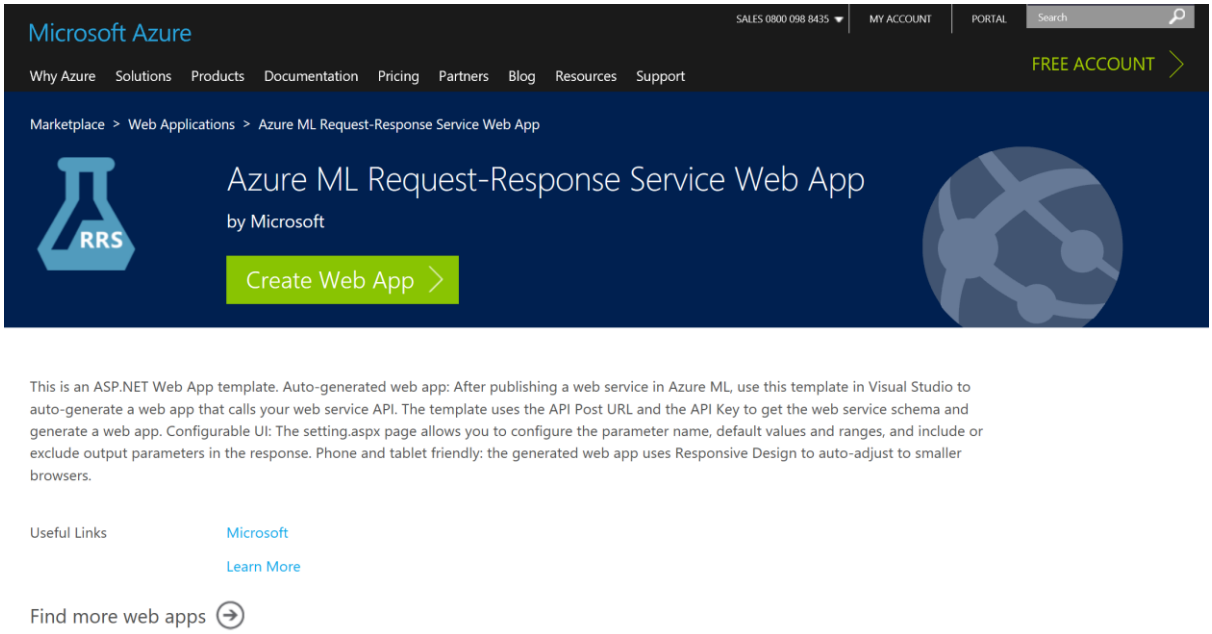
Optional Exercise – create a Web Site using the API

We can also test our new api using a partially configured web site in the Azure Web App Marketplace. Simply go to the link above and search for Azure ML.

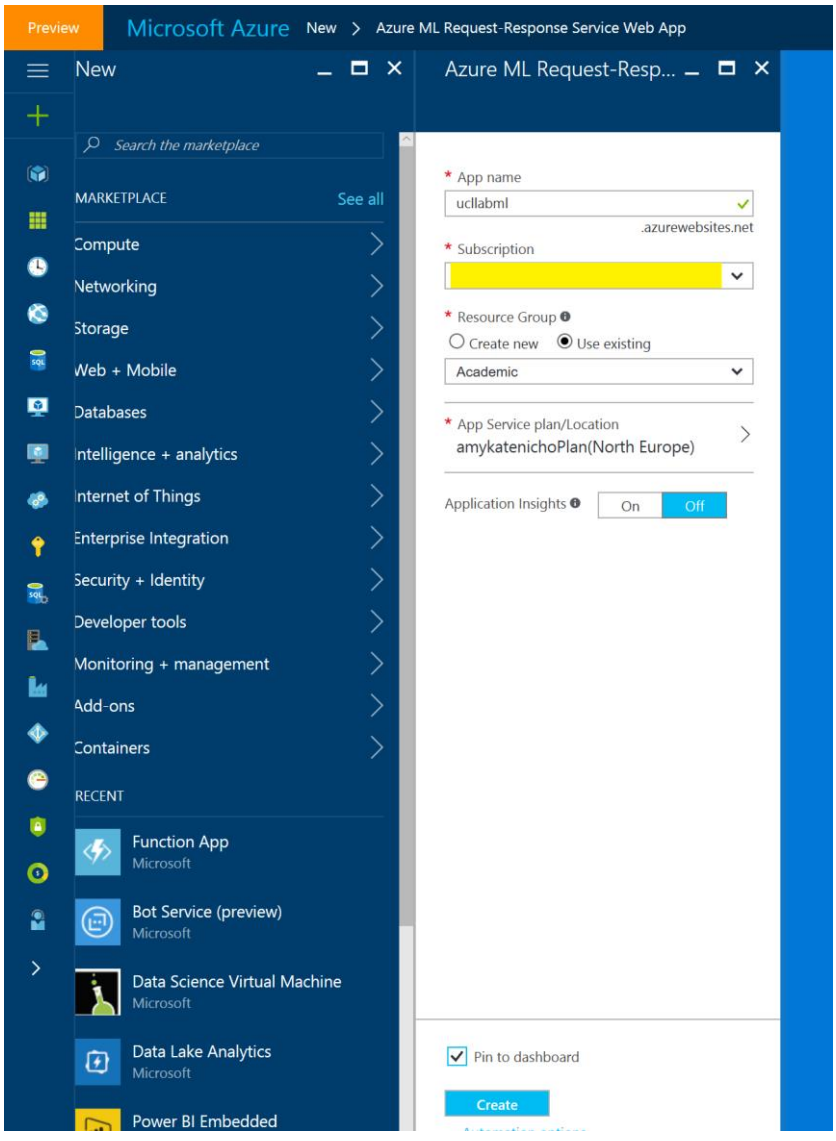
This will create a web app in your azure subscription. In this example we will use the Azure ML Request Response service.



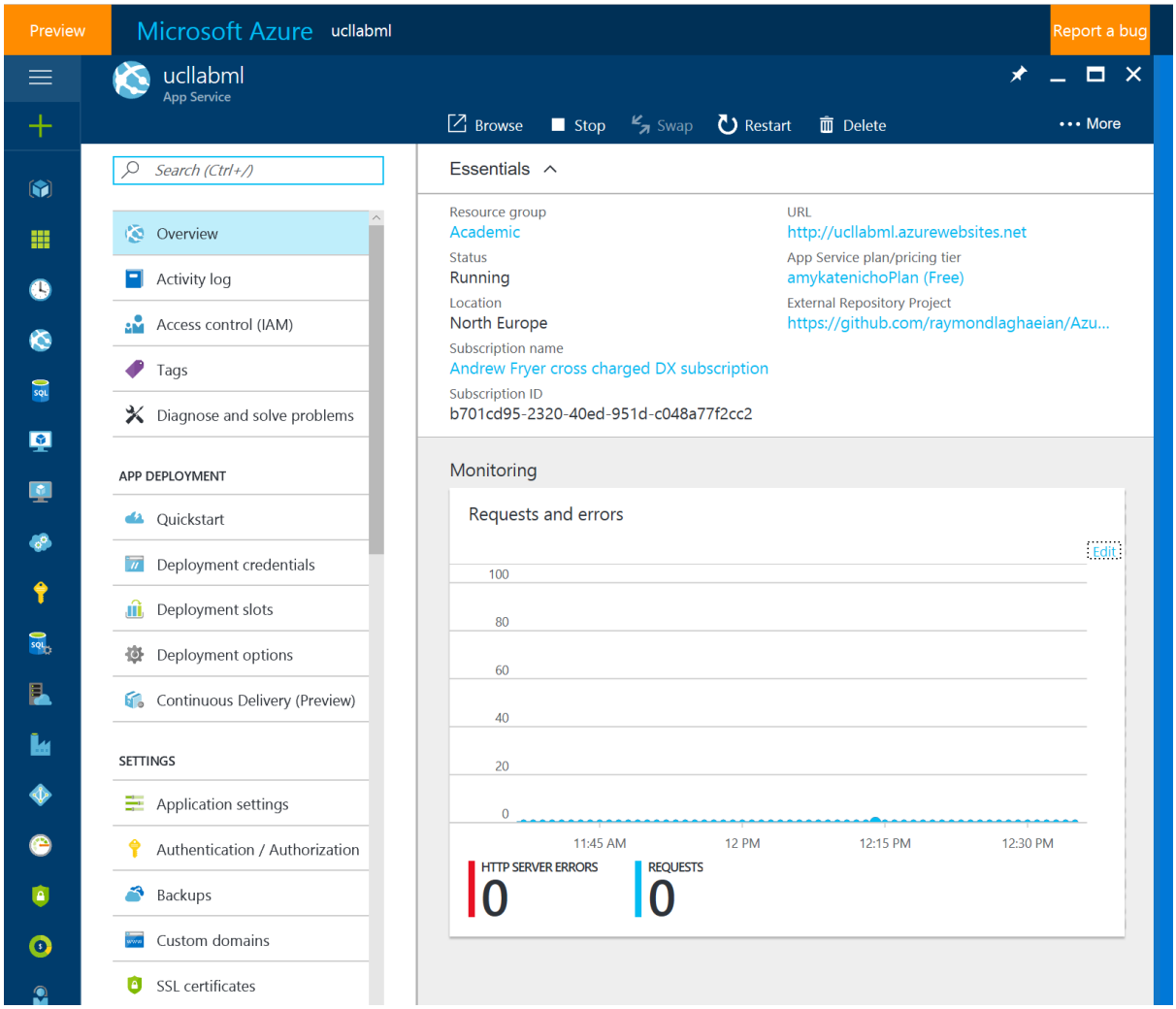
Choose create web app



Give your web app a name and fill in the subscription properties

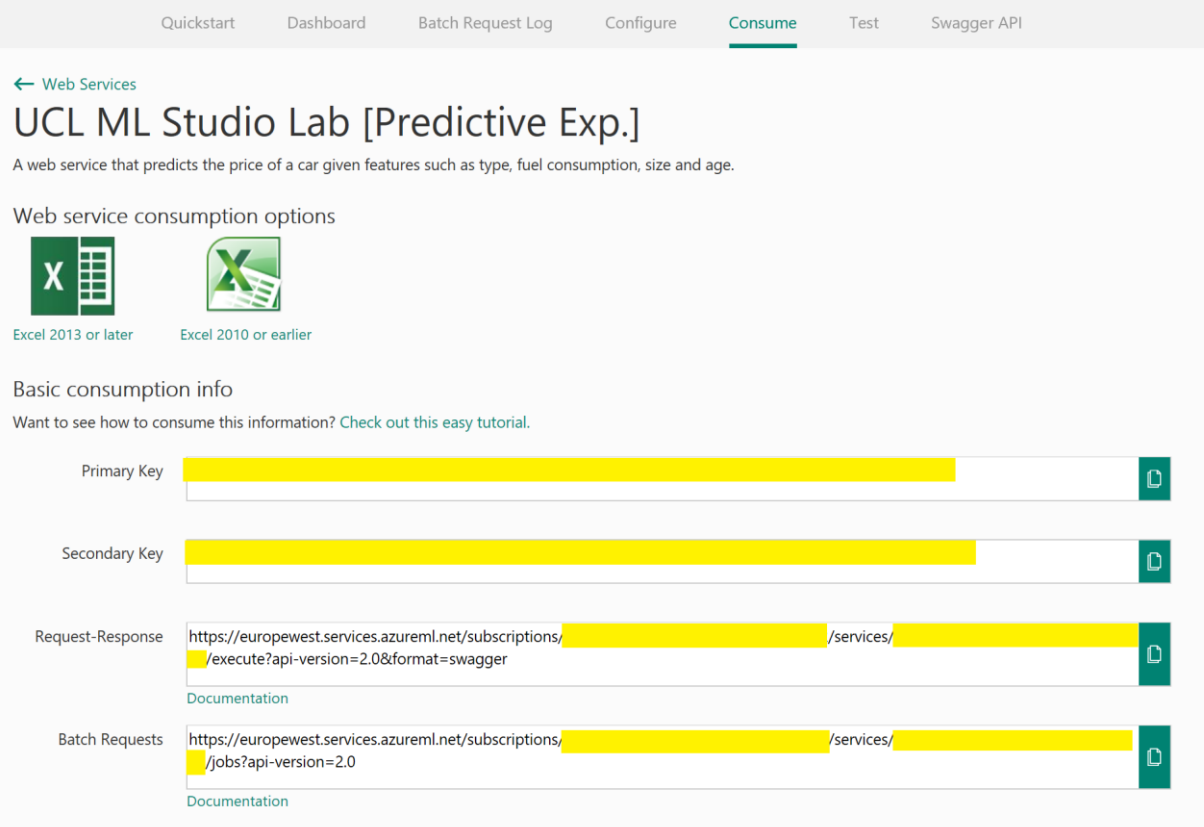


Once deployed you can browse to your website from the Azure Portal using the browse button at the top of the screen

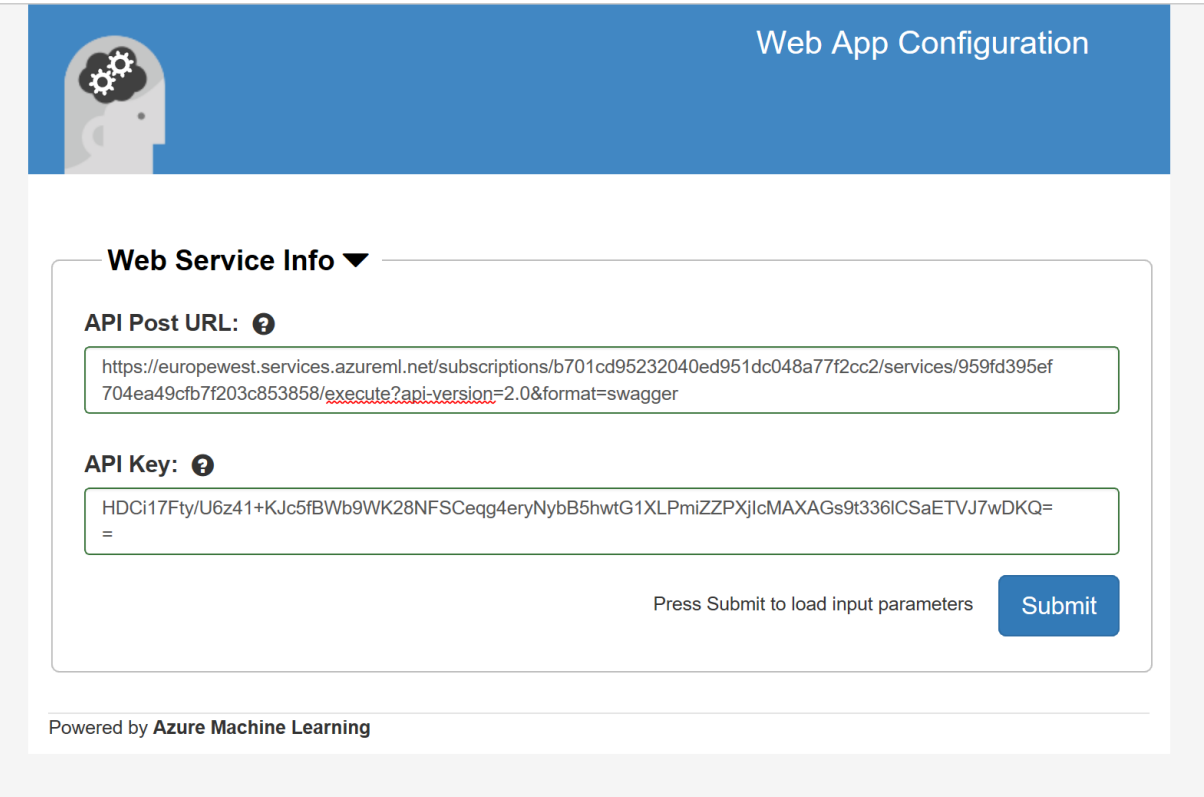


Once the web page opens you need to provide your web service URL (in this case request response) and your API keys. Remember you can get this information from the web service management portal (http://services.azureml.net/) on the Consume tab

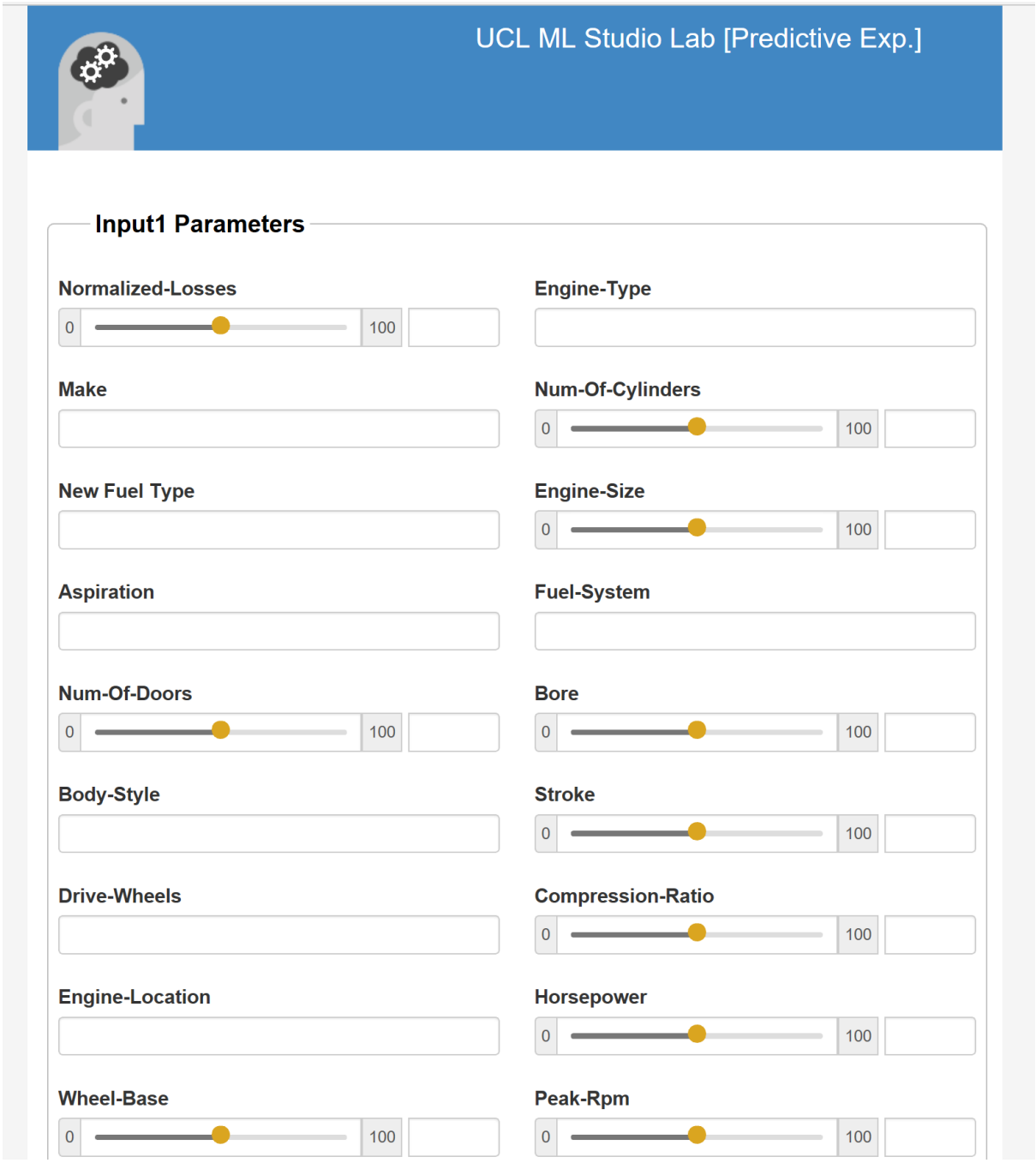
You will need the Primary Key and Request-Response URL.



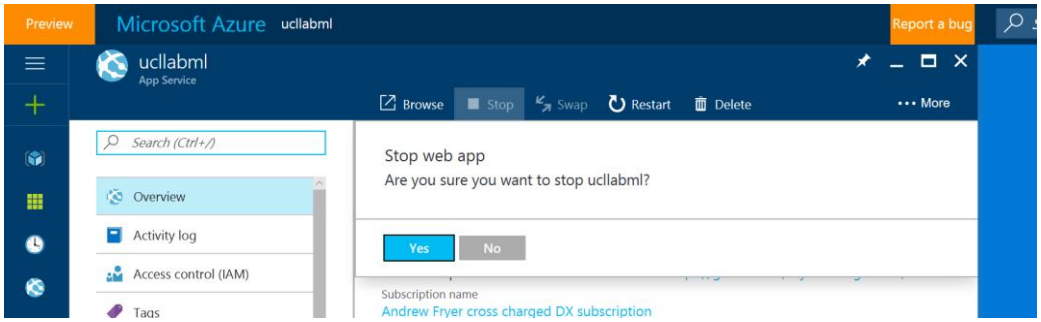
Paste them into the web app configuration as below



Once submitted you should see the UI is rebuilt containing form input and sliders for your experiment. Fill in the fields and click submit and you should be provided with your predicted price estimation at the bottom of the page



Note: remember to go back to the Azure portal and choose the stop button on your web app



Conclusion

This lab was intended to show you the options you have to take your machine learning model and python code and publish them as a web service. You should now know multiple ways of accessing and consuming your web service as well as understanding how you can manage the cost and usage using the portal.