



Machine
Learning

Azure Machine Learning
Studio
- Publishing Lab



Contents

- What Did We Cover Last Time?
- The Problem Domain Explained
- Publishing a Web Service – UI based
- Publishing a Web Service – Jupyter Notebooks
- Consuming a Web Service – in Excel
- [OPTIONAL] Consuming a Web Service – in an App
- Conclusion

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>

The screenshot shows the Cortana Intelligence Gallery interface. At the top, there's a navigation bar with 'Cortana Intelligence Gallery' on the left, a search bar in the center, and user icons on the right. Below the navigation bar, there's a secondary menu with links like 'Browse all', 'Industries', 'Solutions', 'Experiments', 'Machine Learning APIs', 'Custom Modules', 'Learning', and 'More'. The main content area is titled 'EXPERIMENT' and features a thumbnail image of a scatter plot with a regression line. Below the thumbnail, the experiment name 'UCL ML Studio Lab Publishing Ready' is displayed, along with the author's name 'Amy Nicholson' and the date 'November 28, 2016'. There are also 'Summary' and 'Description' sections with some text and a small icon. On the right side of the experiment card, there's a large green button labeled 'Open in Studio'. At the bottom of the card, there are links for 'Add to Collection' and statistics showing '0 views' and '0 downloads'.

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.

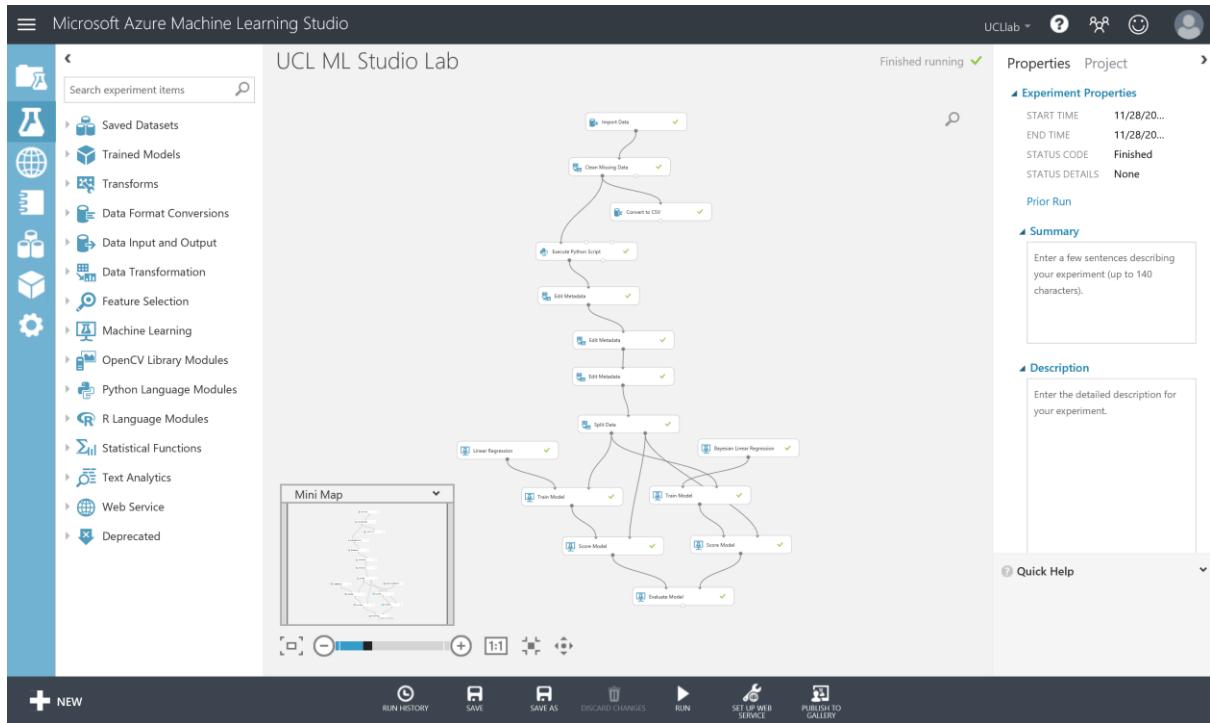
We retrieved data from an Azure Blob Storage account and started to pre-process the dataset ready to train a machine learning model. The model we created is a form of supervised learning so we used historical car attributes and values to predict the price of future cars we might receive. This model performs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

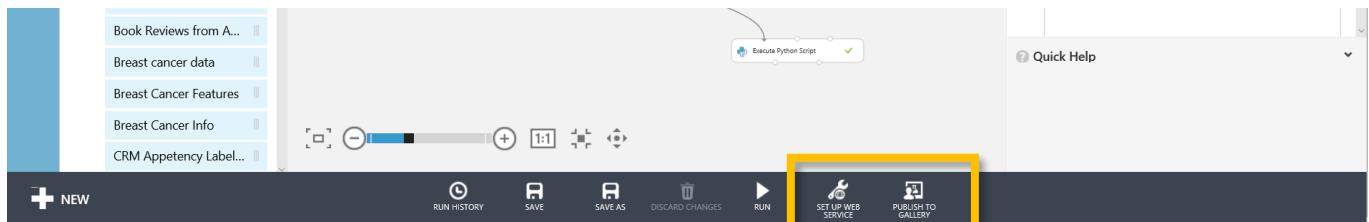
We have trained a model to predict a car's price given some attributes of the car, with the least error possible (a regression experiment). Now we want to use one of Azure Machine Learning's key features – Deploying the model, by publishing an API to expose the model we have created.

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):

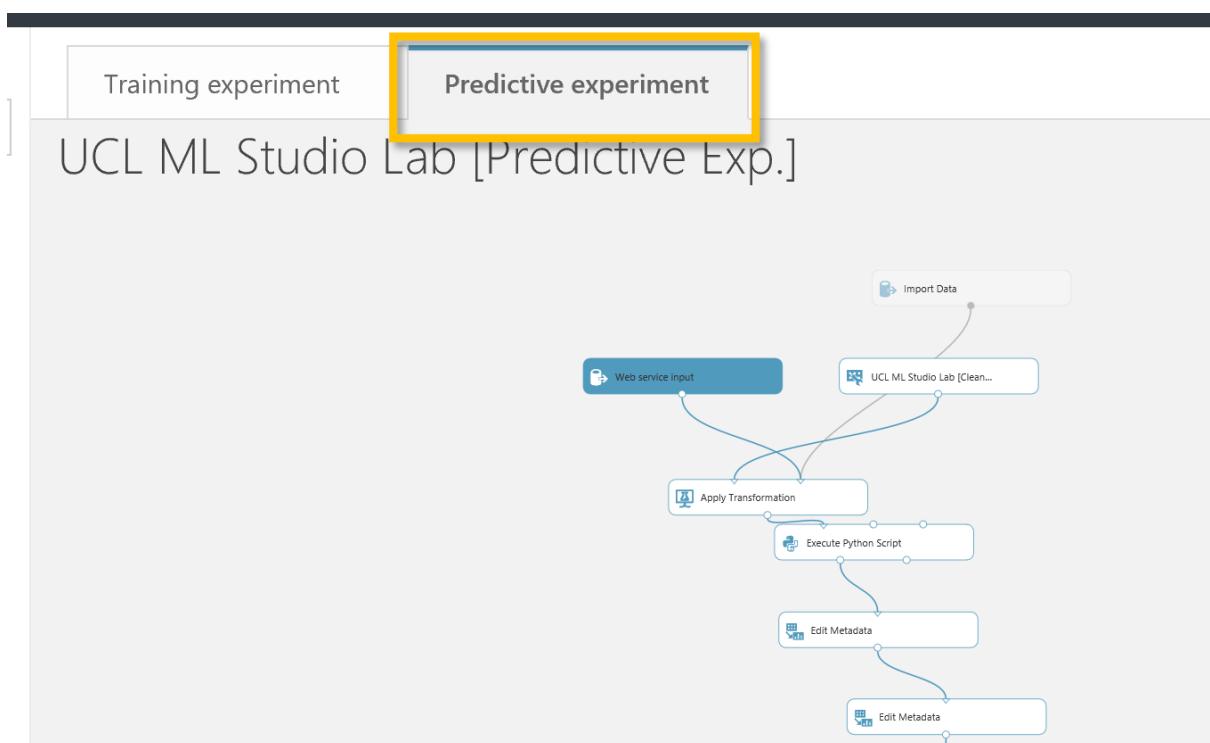


This model can then be published to the Azure ML Web API service to make it available for other users or applications to use as a web service or a REST endpoint.

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.

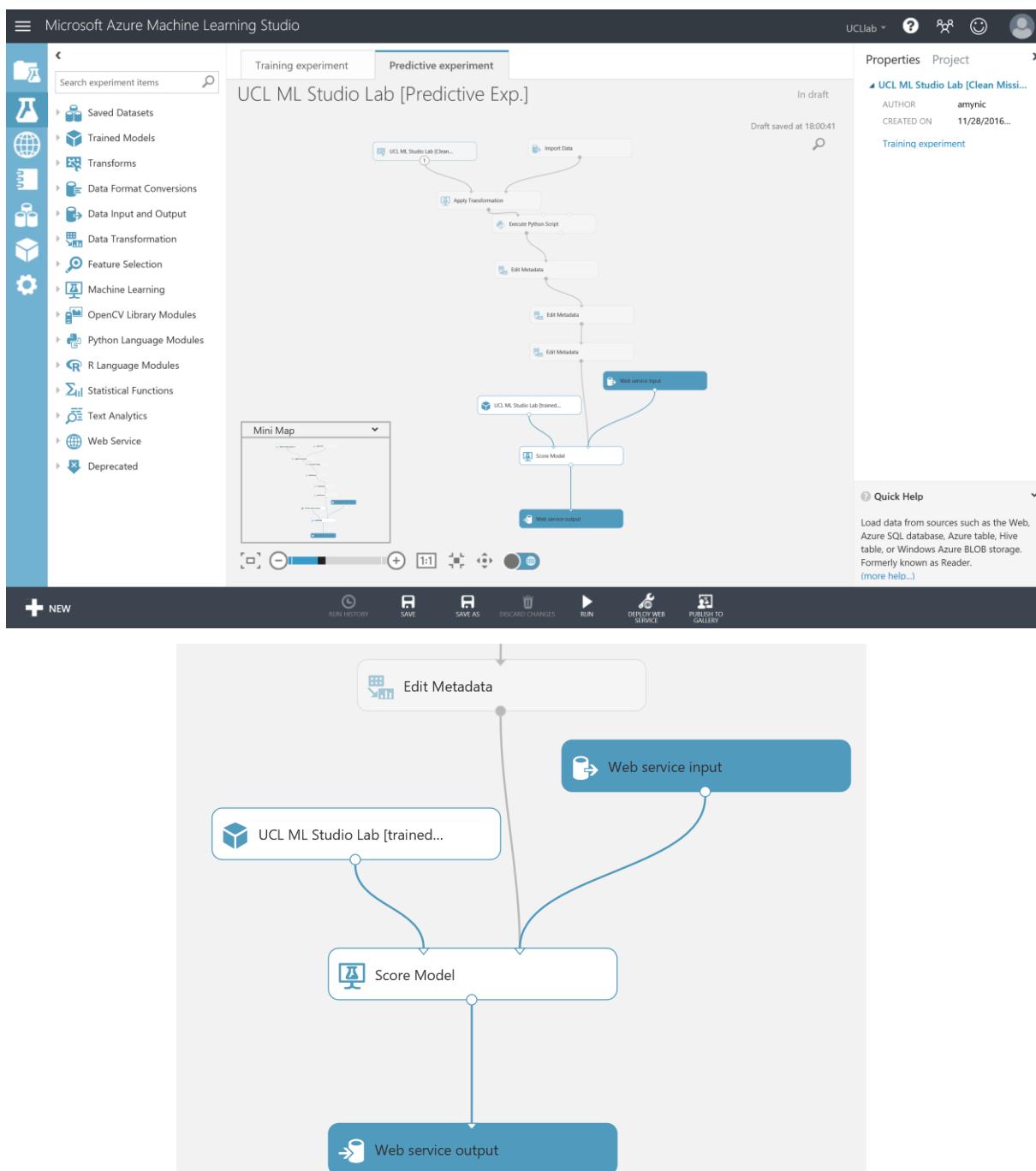
What is a predictive experiment and why do we need it? The purpose of the predictive experiment is to use your trained model to score new data, with the goal of eventually becoming operationalized as an Azure Web service. This conversion is done for you through the following steps:

- 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

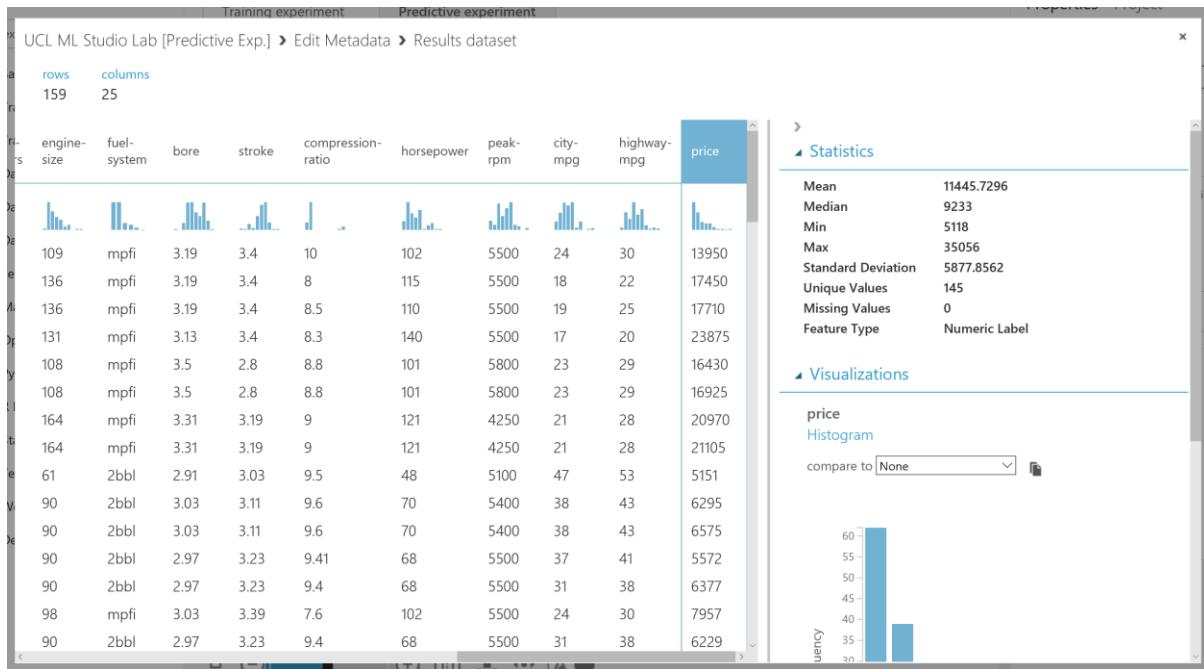
The added input and output ports, "Web service input" and "Web service output", respectively represent the data format that will flow into and out of the web service we are creating.

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:



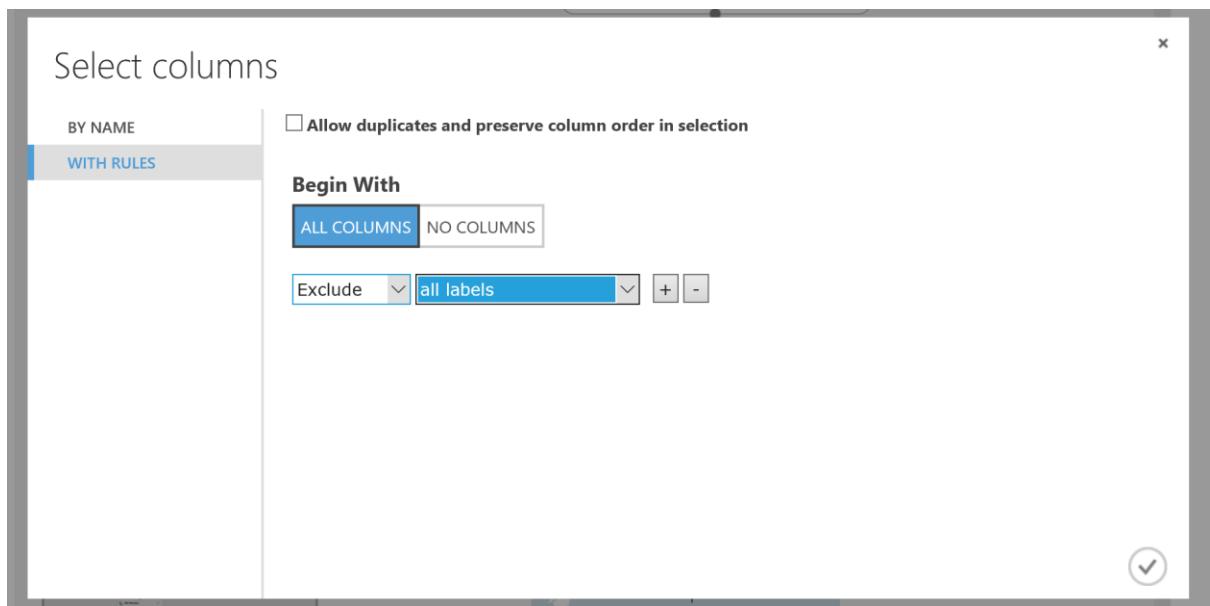
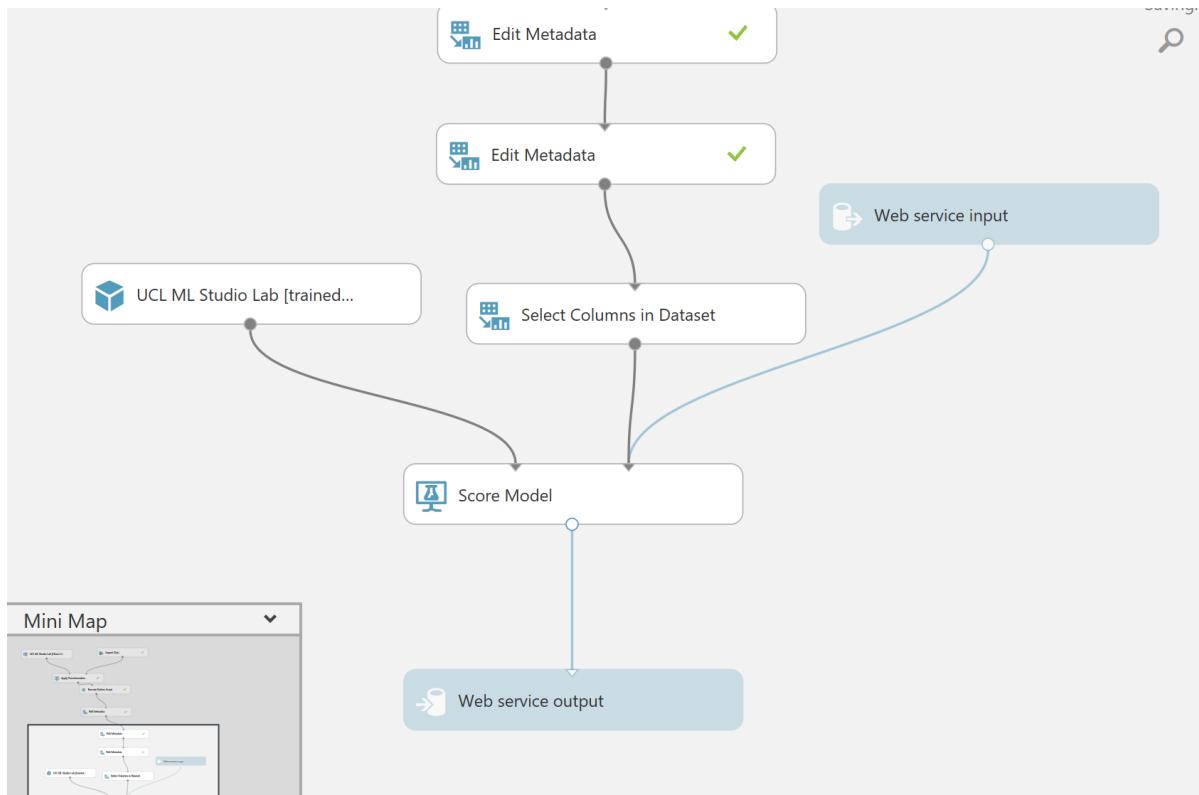
The data flowing through the grey path, from Edit Metadata to Score Model, in the above diagram, now contains the schema of the input to our web service. Run this experiment and then right click and visualise the output of the Edit Metadata module.



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.

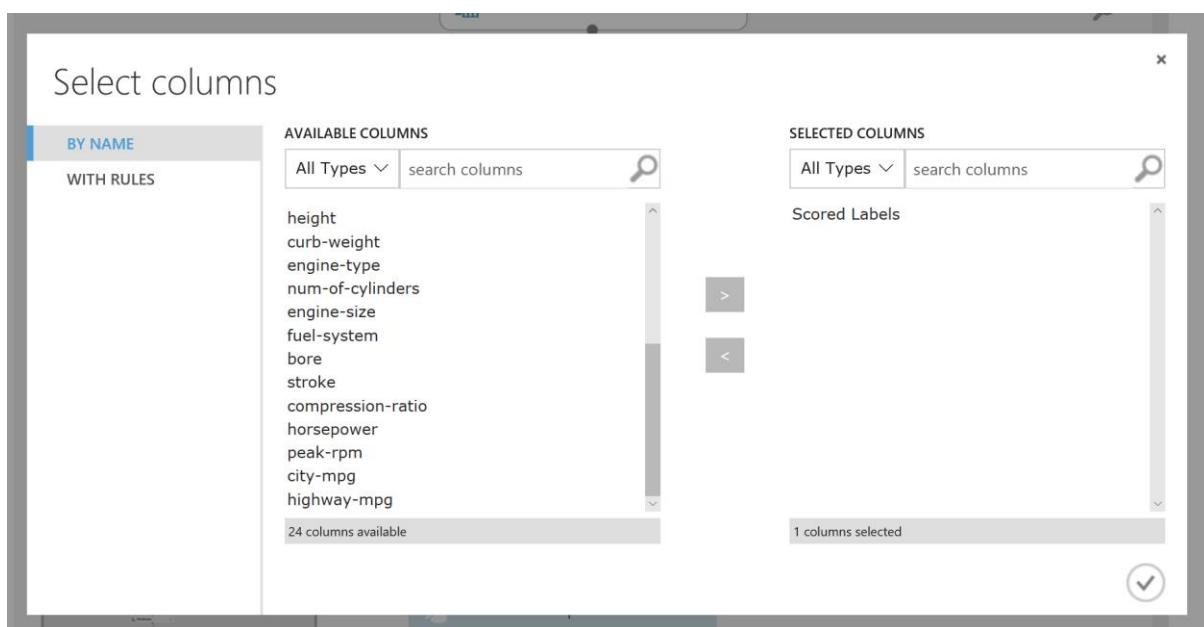
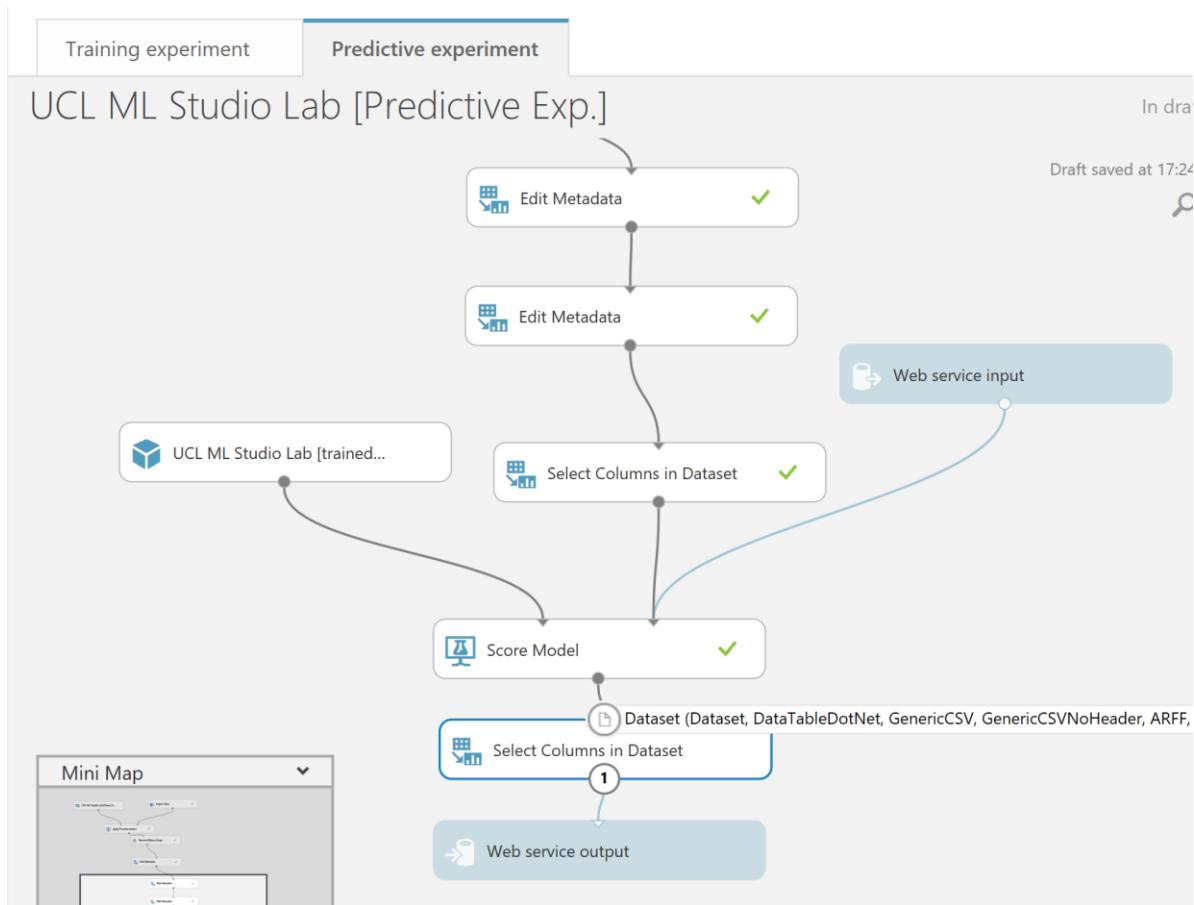
So we can eliminate this column by adding a 'Select columns in a Dataset' module as shown below that excludes all labels from the experiment:

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



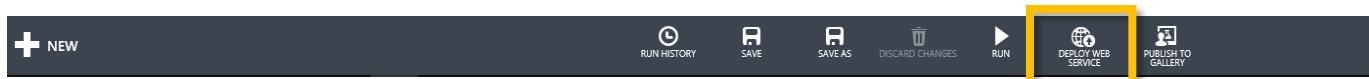
If we now think about what fields we want to return to the application or web site that will call our web service we are creating, then all we technically need is the scored label (the prediction) for the regression experiment. This makes an assumption that we keep hold of nay data needed on the front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

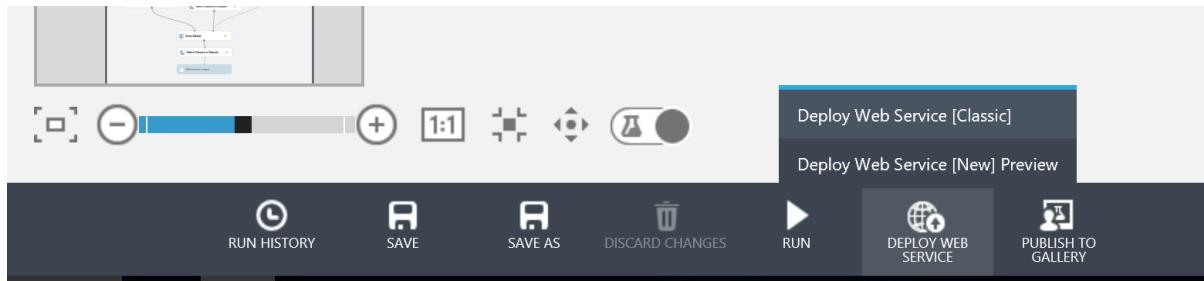


We must now **Run** this predictive experiment as this allows ML to validate the predictive experiment before we can publish it as web service, choose the **RUN** option from the bottom toolbar.

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.

The screenshot shows a deployment page for an experiment. The title is "Deploy 'UCL ML Studio Lab [Predictive Exp.]' experiment as a web service".

Fields filled in:

- Web Service Name: UCLMLStudioLab_webservice
- Price Plan: Free

Important note: "The plan tiers default to the plans in your default region and your web service will be deployed to that region. By clicking on "Deploy", you agree to pay the plan charges in accordance with the [Pricing Page](#)".

At the bottom left is a green "Deploy" button. At the bottom right are links for Microsoft, FAQ, Privacy and Cookies, Terms of Use, and © Microsoft.

Below the main form, there is a summary table:

Web Service Name	UCLMLStudioLab_webservice
Price Plan	Free

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***

The screenshot shows the Microsoft Azure Machine Learning Web Services interface. At the top, there's a navigation bar with links for Quickstart, Dashboard, Batch Request Log, Configure, Consume, Test, and Swagger API. Below the navigation bar, a breadcrumb trail shows 'Web Services' and the specific service name 'UCL ML Studio Lab [Predictive Exp.]'. The main content area is divided into three sections: 'BASICS' (with a cloud icon), 'MANAGE & MONITOR' (with a bar chart icon), and 'DEVELOP' (with a code icon). Under the 'BASICS' section, there are links for 'Test Web Service', 'Configure Web Service', 'Use Web Service', and 'Launch in Excel'. Under 'MANAGE & MONITOR', there's a link to 'View usage statistics'. Under 'DEVELOP', there's a link to 'Swagger Documentation' and a 'Tutorial: How to build apps'. At the bottom of the page, there's a Microsoft logo and links for FAQ, Privacy and Cookies, Terms of Use, and © Microsoft.

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.

The screenshot shows the 'Configure' page for the 'UCL ML Studio Lab [Predictive Exp.]' web service. The page has a header with links for Quickstart, Dashboard, Batch Request Log, Configure (which is highlighted in green), Consume, Test, and Swagger API. Below the header, there's a breadcrumb trail for 'Web Services' and the service name. The main form contains fields for 'Description' (a text area with placeholder text about predicting car prices), 'Title' (text input with value 'UCL ML Studio Lab [Predictive Exp.]'), 'Primary Key' (text input with placeholder hex string), 'Secondary Key' (text input with placeholder hex string), 'Storage Account Name' (checkbox labeled 'Update Key for uclabstorage'), 'Sample Data Enabled?' (radio buttons for 'Yes' and 'No' with 'Yes' selected), and 'Enable Logging' (checkboxes for 'None', 'Error', and 'All' with 'All' selected). At the bottom of the form are 'Cancel' and 'Save' buttons. At the very bottom of the page, there's a Microsoft logo and links for FAQ, Privacy and Cookies, Terms of Use, and © Microsoft.

You should receive a message stating the update was successful:

Web Service has been successfully updated.

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.

The screenshot shows the UCL ML Studio Lab [Predictive Exp.] web service interface. The top navigation bar includes links for Quickstart, Dashboard, Batch Request Log, Configure, Consume, **Test**, and Swagger API. Below the navigation is a breadcrumb trail: < Web Services and the main title UCL ML Studio Lab [Predictive Exp.]. A subtitle indicates it's a web service that predicts the price of a car given features such as type, fuel consumption, size and age. The main area is divided into two sections: 'input1' and 'output1'. Under 'input1', there are ten input fields with their corresponding values: normalized-losses (164), make (audi), new-fuel-type (petrol), aspiration (std), num-of-doors (4), body-style (sedan), drive-wheels (fwd), engine-location (front), wheel-base (99.8), and length (176.6). To the right of these inputs, under 'output1', is the 'Scored Labels' section with the value 15633.2186841452. At the bottom left of the input section, there are 'Request-Response' and 'Batch' buttons, with 'Request-Response' being the active tab.

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.

Quickstart Dashboard Batch Request Log Configure Consume **Test** Swagger API

[← Web Services](#)

UCL ML Studio Lab [Predictive Exp.]

A web service that predicts the price of a car given features such as type, fuel consumption, size and age.

Request-Response Batch

▽ input1

[Browse...](#)

▽ Test Batch Jobs

JOB ID	STATUS	RUN START	RUN END	DURATION	Result
0009a2c34cc542cfbe63c0d92c1c21f3	Finished	11/29/2016 18:05 PM M	11/29/2016 18:05 PM	7s	output1

1 / 1

[Logging Help ↗](#)

Note: We will enable CORS on your storage account to upload this file

[Test](#)

Microsoft
[FAQ](#) [Privacy and Cookies](#) [Terms of Use](#) © Microsoft

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:

Quickstart **Dashboard** Batch Request Log Configure Consume Test Swagger API

[← Web Services](#)

UCL ML Studio Lab [Predictive Exp.]

A web service that predicts the price of a car given features such as type, fuel consumption, size and age.

REQUESTS OVER TIME

Period: Last Week ▾

Batch Request-Response

REQUEST-RESPONSE REQUESTS

Total 0
Failed 0

AVERAGE REQUEST-RESPONSE COMPUTE TIME

0.00 milliseconds

BATCH REQUESTS

Total 1
Failed 0

AVERAGE JOB LATENCY

7.60 seconds

ERRORS

0

PLAN USAGE

Transactions 1
Compute Hours 0.002

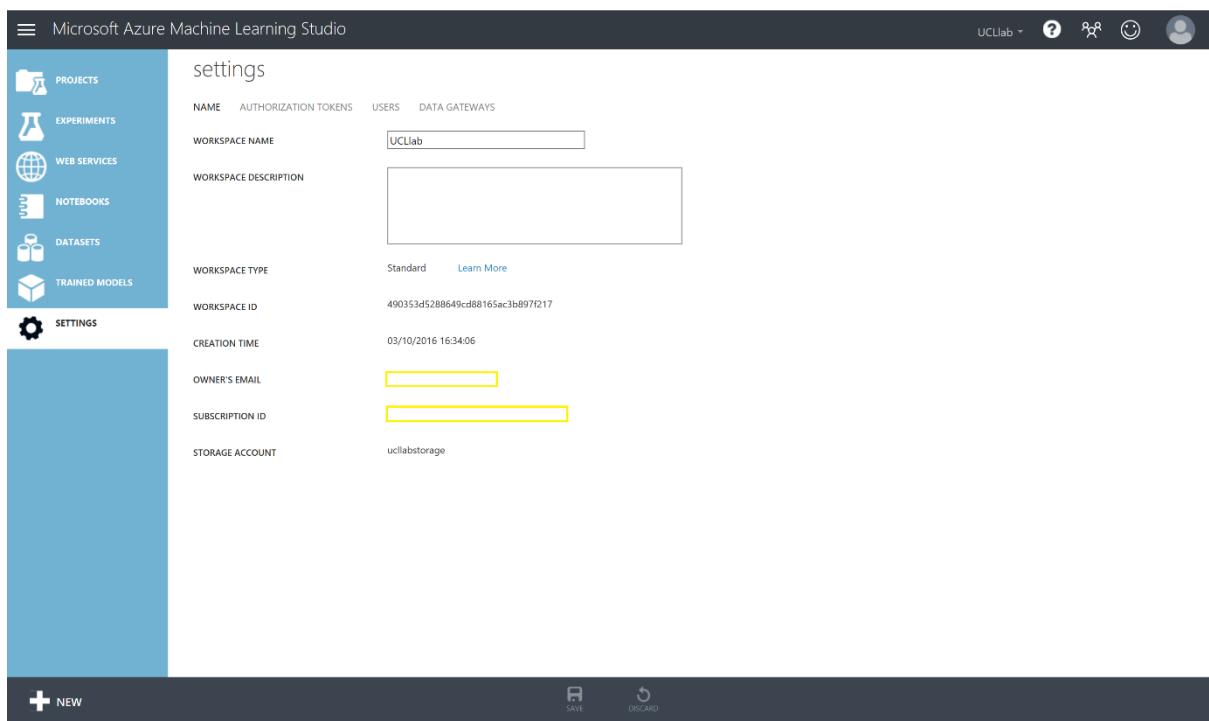
Finally, the 'Consume' tab provides your web service API keys, excel spreadsheets with your model preloaded and also sample code in C#, Python, Python 3+ and R to implement into your applications.

The screenshot shows the 'Consume' tab of the UCL ML Studio Lab [Predictive Exp.] interface. At the top, there are tabs for Quickstart, Dashboard, Batch Request Log, Configure, Consume (which is highlighted in green), Test, and Swagger API. Below the tabs, there's a back arrow labeled 'Web Services' and the title 'UCL ML Studio Lab [Predictive Exp.]'. A subtitle states: 'A web service that predicts the price of a car given features such as type, fuel consumption, size and age.' Under the heading 'Web service consumption options', there are two icons: one for 'Excel 2013 or later' showing an Excel logo with a chart, and another for 'Excel 2010 or earlier' showing a similar icon. Below these are sections for 'Basic consumption info' and 'Request-Response' and 'Batch Requests' endpoints. Each endpoint section includes a 'Primary Key' and 'Secondary Key' input field, a 'Documentation' link, and a copy icon. The 'Request-Response' endpoint URL is partially redacted: `https://europewest.services.azureml.net/subscriptions/` [REDACTED] `/services/` [REDACTED] `58/execute?api-version=2.0&format=swagger`. The 'Batch Requests' endpoint URL is partially redacted: `https://europewest.services.azureml.net/subscriptions/` [REDACTED] `/services/` [REDACTED] `58/jobs?api-version=2.0`.

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



The screenshot shows the 'settings' tab in the Microsoft Azure Machine Learning Studio. The left sidebar has icons for PROJECTS, EXPERIMENTS, WEB SERVICES (selected), NOTEBOOKS, DATASETS, TRAINED MODELS, and SETTINGS. The main area displays workspace details:

NAME	AUTHORIZATION TOKENS	USERS	DATA GATEWAYS
WORKSPACE NAME	UCLLab		
WORKSPACE DESCRIPTION			
WORKSPACE TYPE	Standard	Learn More	
WORKSPACE ID	490353d5288649cd88165ac3b897f217		
CREATION TIME	03/10/2016 16:34:06		
OWNER'S EMAIL			
SUBSCRIPTION ID			
STORAGE ACCOUNT	ucllabstorage		

At the bottom, there are 'SAVE' and 'DISCARD' buttons.

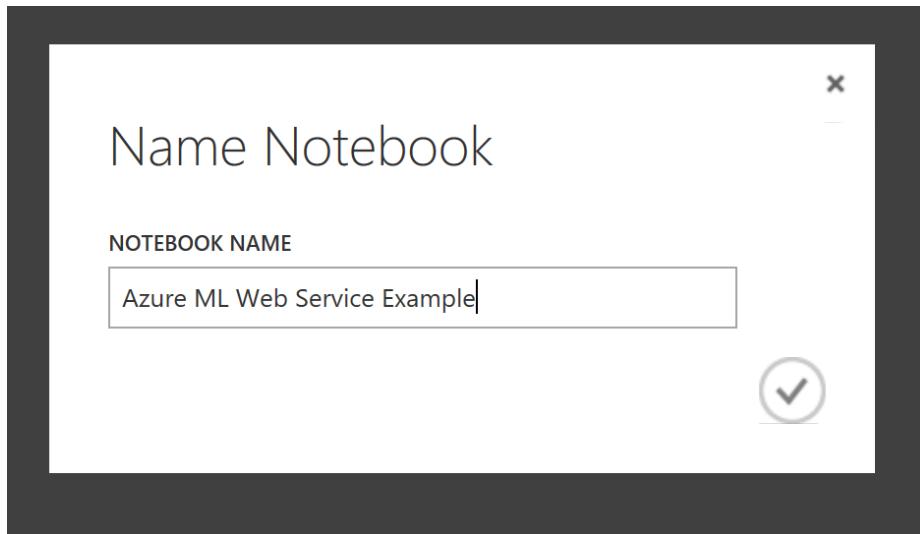
Take a note of your workspace ID from here and then enter the Authorisation Token header and take note of your Primary Authorisation token. You will need these later on in the lab.

The screenshot shows the Microsoft Azure Machine Learning Studio interface. On the left, there's a sidebar with icons for Projects, Experiments, Web Services, Notebooks, Datasets, and Trained Models. Below that is a 'SETTINGS' section with a gear icon. At the bottom of the sidebar is a 'NEW' button. The main area is titled 'settings' and contains sections for NAME, AUTHORIZATION TOKENS, USERS, and DATA GATEWAYS. It shows two authorization tokens, both of which are yellowed out. There are 'Regenerate' buttons for each token.

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

The screenshot shows the 'notebooks' page in Microsoft Azure Machine Learning Studio. The left sidebar has a 'NEW' section with options like Dataset, Module, Project Preview, Experiment, and Notebook Preview, with 'Notebook Preview' currently selected. The main area shows a search bar and a list of notebooks. It includes a 'FROM LOCAL FILE' upload option and a 'Tutorial on Azure Machine Learning Notebook' sample. Below are four blank notebook templates: Python 3, Python 2, R, and a general one. At the bottom, there are links for 'Access Azure ML Experiment Data', 'Variable Selection in Azure ML Jupyter Notebook', 'GBM in Azure ML Jupyter Notebook', and 'Evaluating Multiple Models'.

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 azureml)
Requirement already satisfied: pytz>=2011k in /home/nbcommon/anaconda2_20/lib/python2.7/site-packages (from pandas->azureml)
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-dateutil->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).

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 azureml)
Requirement already satisfied: pytz>=2011k in /home/nbcommon/anaconda2_20/lib/python2.7/site-packages (from pandas->azureml)
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-dateutil->azureml)

In [8]:

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

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
```

In [17]:

```
myfirstservice.service.url
```

Out[17]:

```
u'https://europewest.services.azureml.net/workspaces/.../services/.../execute?api-version=2.0'
```

- Web service help page
- ```
Myfirstservice.service.help_url
```

In [18]:

```
myfirstservice.service.help_url
```

Out[18]:

```
u'https://europewest.studio.azureml.net/apihelp/workspaces/.../webservices/.../endpoints/.../score'
```

To call your 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 [19]:

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

Out[19]:

```
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 your 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:

The screenshot shows the Microsoft Azure Machine Learning Web Services consumer interface. At the top, there are tabs for Quickstart, Dashboard, Batch Request Log, Configure, **Consume**, Test, and Swagger API. The Consume tab is selected. Below it, there's a link to 'Web Services' and a status message 'Loading data...'. A note says 'A web service that predicts the price of a car given features such as type, fuel consumption, size and age.' Under 'Web service consumption options', there are two icons: one for 'Excel 2013 or later' and one for 'Excel 2010 or earlier'. The 'Excel 2013 or later' icon is highlighted. Below these are sections for 'Basic consumption info', 'Request-Response', and 'Batch Requests', each with a URL and a 'Documentation' link. At the bottom, there's a 'Sample Code' section with tabs for 'Request-Response' (selected) and 'Batch', and buttons for 'C#' (disabled), 'Python', 'Python 3+', and 'R'. A code snippet in C# is shown:

```
https://services.azureml.net/
// This code requires the NuGet package Microsoft.AspNet.WebApi.Client to be installed.
// Instructions for doing this in Visual Studio: https://go.microsoft.com/fwlink/?LinkId=393789
// Instructions for doing this in Power BI Desktop: https://go.microsoft.com/fwlink/?LinkId=393790
```

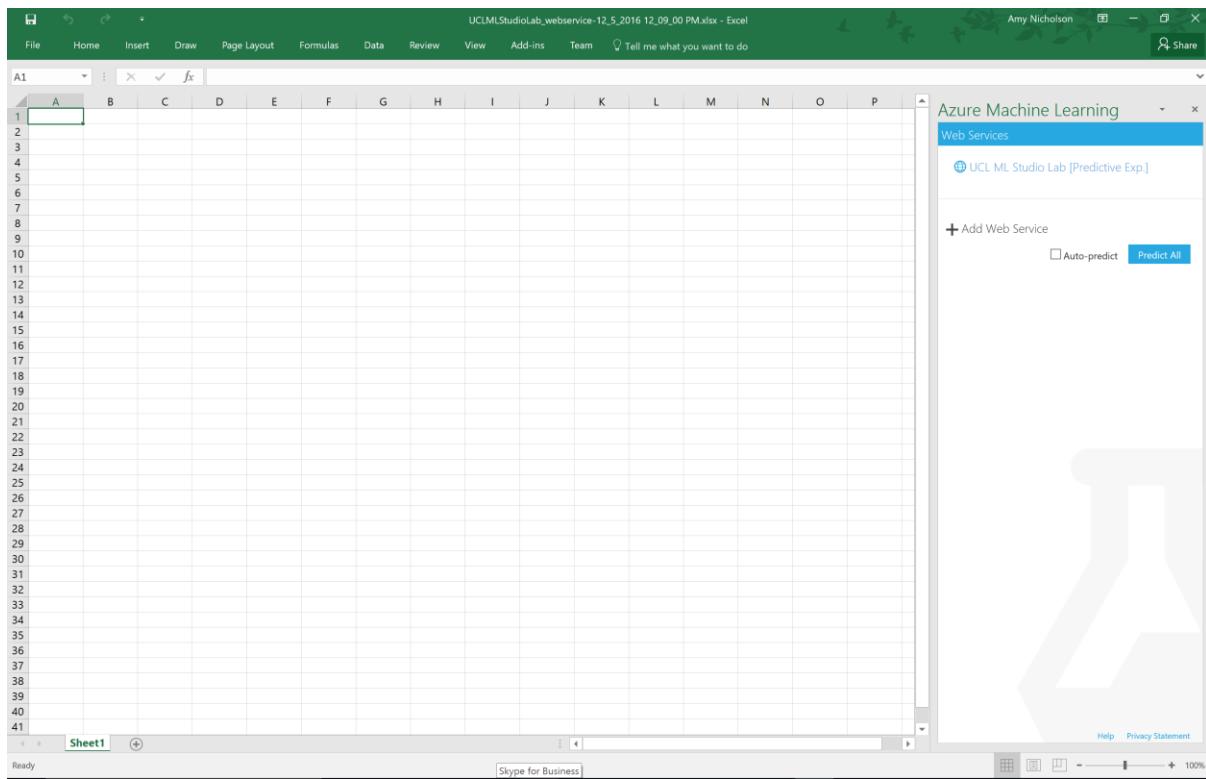
Select the excel version you have available (this tutorial will select the Excel 2013 or later option) and save the download file

The screenshot shows the Microsoft Azure Machine Learning Web Services interface. At the top, there are tabs: Quickstart, Dashboard, Batch Request Log, Configure, **Consume**, Test, and Swagger API. The **Consume** tab is selected. Below it, under 'Web Services', is the title 'UCL ML Studio Lab [Predictive Exp.]'. A sub-header says 'A web service that predicts the price of a car given features such as type, fuel consumption, size and age.' There are two icons for consumption: 'Excel 2013 or later' (with a green checkmark) and 'Excel 2010 or earlier'. Below these are sections for 'Basic consumption info' and 'Web service consumption options'. Under 'Basic consumption info', there are four fields: Primary Key (containing a long URL), Secondary Key (containing another long URL), Request-Response (containing a URL), and Batch Requests (containing a URL). Each field has a 'Documentation' link below it. Under 'Web service consumption options', there are two more icons: 'Excel 2013 or later' and 'Excel 2010 or earlier'. At the bottom, there is a 'Sample Code' section with a 'Request-Response' tab selected, showing a dialog box with the message 'What do you want to do with UCLMLStudioLab\_webservice-12\_5\_2016 12\_09\_00 PM.xlsx (7.08 KB)? From: officeapps.azureml.net'. The dialog has 'Save', 'Save as', and 'Cancel' buttons.

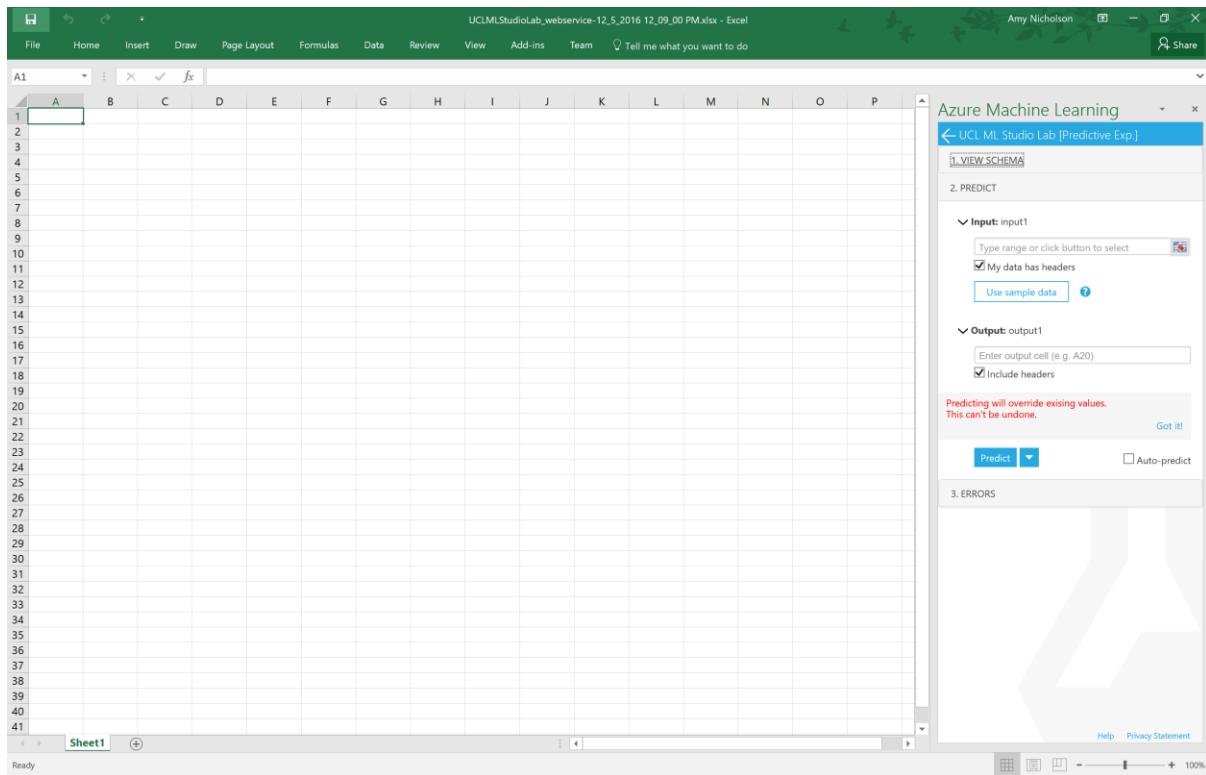
Once Excel has opened, choose enable editing in the top yellow bar

The screenshot shows a Microsoft Excel window titled 'UCLMLStudioLab\_webservice-12\_5\_2016 12\_09\_00 PM.xlsx [Protected View] - Excel'. The top ribbon has tabs: File, Home, Insert, Draw, Page Layout, Formulas, Data, Review, View, Add-ins, Team, and a 'Tell me what you want to do' search bar. A yellow 'PROTECTED VIEW' bar at the top says 'Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.' with a 'Enable Editing' button. The main area is a blank spreadsheet with columns labeled A through U and rows numbered 1 to 39. The status bar at the bottom right shows 'Ready' and '100%'. The bottom navigation bar includes 'Sheet1' and other sheet icons.

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.

at you want to do

Share

The screenshot shows a browser-based interface for Azure Machine Learning. On the left, there is a large, empty spreadsheet grid with columns labeled M, N, O, and P. On the right, a sidebar titled "Azure Machine Learning" displays the schema for a dataset named "UCL ML Studio Lab [Predictive Exp.]". The sidebar has a header "1. VIEW SCHEMA" and a section titled "Inputs" which lists various features with their types:

- input1
  - normalized-losses (Integer)
  - make (String)
  - new fuel type (String)
  - aspiration (String)
  - num-of-doors (Integer)
  - body-style (String)
  - drive-wheels (String)
  - engine-location (String)
  - wheel-base (Number)
  - length (Number)
  - width (Number)
  - height (Number)
  - curb-weight (Integer)
  - engine-type (String)
  - num-of-cylinders (Integer)
  - engine-size (Integer)
  - fuel-system (String)
  - bore (Number)

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.

The screenshot shows a Microsoft Excel spreadsheet titled "UCLMLStudioLab\_webservice-12\_5\_2016 12\_09\_00 PM.xlsx - Excel". The data is in a table with columns labeled A through N. The first few rows contain data points, such as "164 audi petrol std 4 sedan fwd front 99.8 176.6 66.2 54.3 2337 ohc". To the right of the Excel window is the "Azure Machine Learning" interface, specifically the "Predictive Exp." section. Under "Input: input1", there is a dropdown menu set to "Sheet1!A1:X6" and a checkbox "My data has headers" which is checked. Under "Output: output1", there is a dropdown menu set to "Sheet1!Y1" and a checkbox "Include headers" which is checked. A message at the bottom says "Predicting will override existing values. This can't be undone." with a "Got it!" button.

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

The screenshot shows the same Microsoft Excel spreadsheet and Azure Machine Learning interface as the previous one. However, the data in the table has been shifted down to rows 2 through 7, and the prediction cell Y1 now contains the value "2337 ohc". The "Input: input1" dropdown is still set to "Sheet1!A1:X6" and "My data has headers" is checked. The "Output: output1" dropdown is set to "Sheet1!Y1" and "Include headers" is checked. The message at the bottom remains the same: "Predicting will override existing values. This can't be undone." with a "Got it!" button.

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

The screenshot shows a Microsoft Excel spreadsheet on the left and the Azure Machine Learning interface on the right.

**Data Grid:**

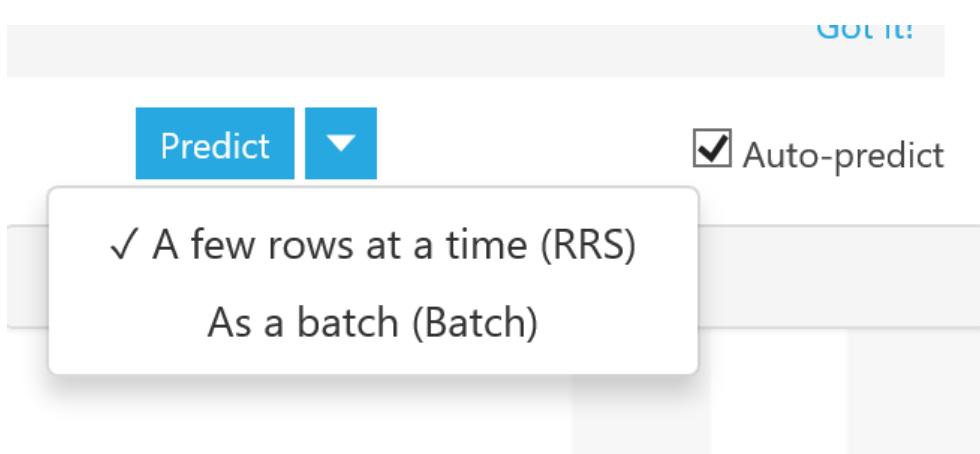
|    | V    | W  | X  | Y        | Z |
|----|------|----|----|----------|---|
| 12 | 5500 | 24 | 30 | 15633.22 |   |
| 15 | 5500 | 18 | 22 | 22423.83 |   |
| 10 | 5500 | 19 | 25 | 22360.36 |   |
| 10 | 5500 | 17 | 20 | 23868.41 |   |
| 11 | 5800 | 23 | 29 | 16773.16 |   |

**Azure Machine Learning Interface:**

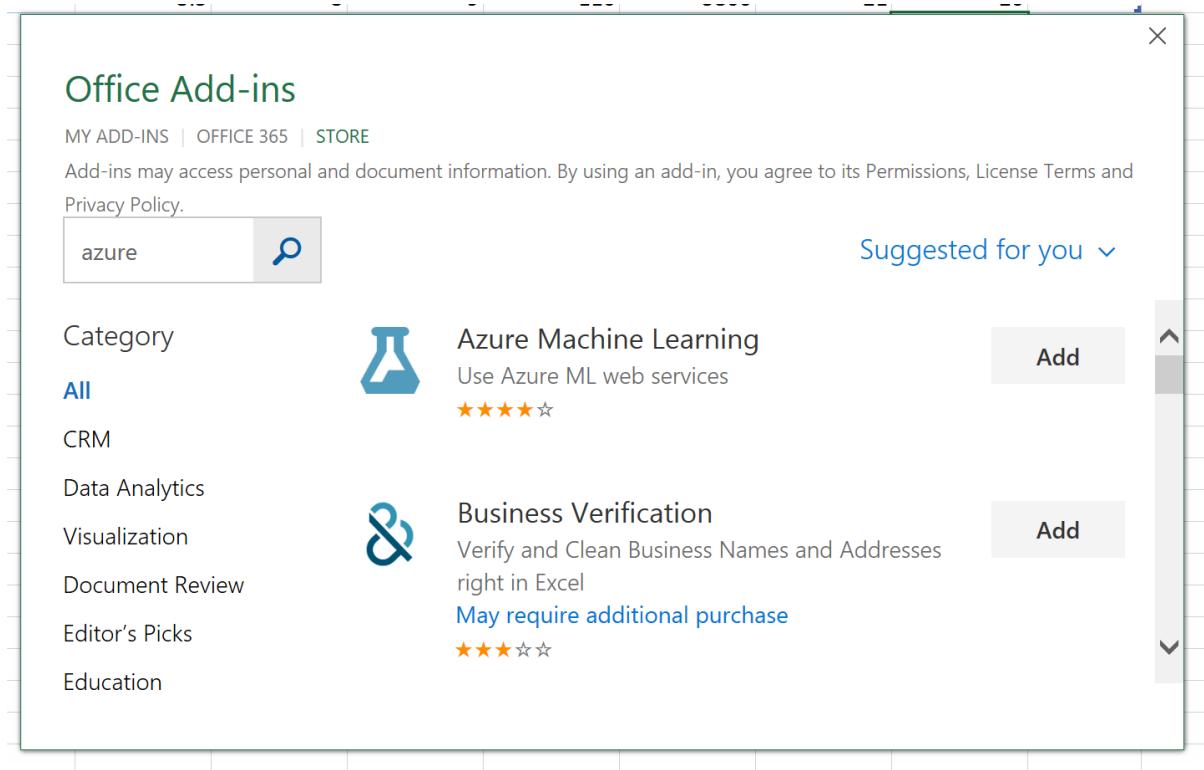
- 1. VIEW SCHEMA:** Shows columns V, W, X, Y, Z with dropdown menus for peak-rpm, city-mpg, highway-mpg, and Scored I.
- 2. PREDICT:**
  - Input:** Set to Sheet1!A1:Y6, with "My data has headers" checked. Includes "Use sample data" and a question mark icon.
  - Output:** Set to Sheet1!Y1, with "Include headers" checked.
  - A warning message: "Predicting will override existing values. This can't be undone." with a "Got it!" button.
  - Predict** button and **Auto-predict** checkbox.
- 3. ERRORS:** Placeholder for error messages.

Other features to note:

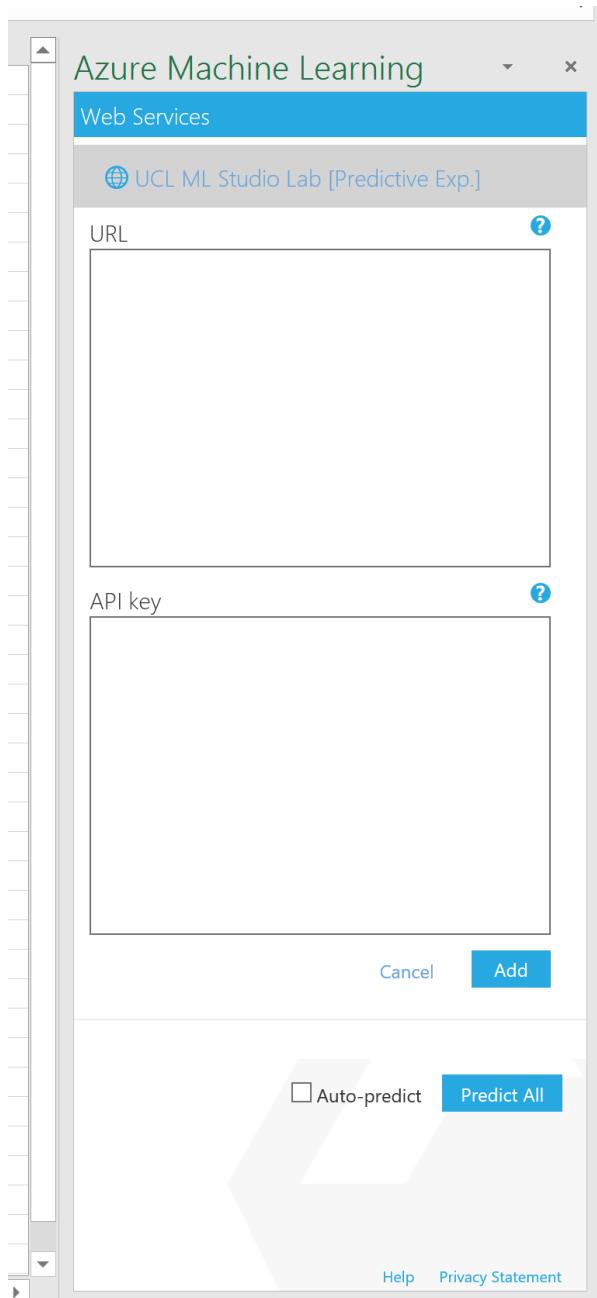
- You can choose the type of prediction you use
  - o Request Response – each row of data is processed separately, more real time option
  - o Batch – all the rows of data are sent as a batch and returned results as a batch.
- You can check the Auto-predict box and populate new data in the excel and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



# 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.

The screenshot shows the Microsoft Azure marketplace interface. At the top, there's a navigation bar with links for Sales, My Account, Portal, and Search, along with a 'FREE ACCOUNT' button. Below the navigation, a breadcrumb trail shows 'Marketplace > Web Applications'. The main heading is 'Web App Marketplace', with a subtext: 'Quickly deploy dynamic blogs, CMS platforms, e-commerce sites, and more with ready-to-use Azure web apps and templates—including hundreds of popular open-source solutions.' A search bar at the bottom contains the query 'azure ml'. Two service cards are visible: 'Azure ML Request-Response' (Microsoft) and 'Azure ML Batch Execution Service' (Microsoft). The Azure ML Request-Response card features a blue icon with a flask and the letters 'RRS'.

---

Choose create web app

The screenshot shows the Microsoft Azure Marketplace page for the "Azure ML Request-Response Service Web App". The top navigation bar includes links for "SALES 0800 098 8435", "MY ACCOUNT", "PORTAL", "Search", and a "FREE ACCOUNT" button. The main content area features a blue header with the text "Azure ML Request-Response Service Web App" and "by Microsoft". Below the header is a green "Create Web App >" button. To the left is a blue icon of a flask labeled "RRS". To the right is a graphic of a globe with network connections. The central text explains the template is an ASP.NET Web App template that auto-generates a web app calling your web service API. It includes a "Phone and tablet friendly" note and a "Responsive Design" mention.

This is an ASP.NET Web App template. Auto-generated web app: After publishing a web service in Azure ML, use this template in Visual Studio to auto-generate a web app that calls your web service API. The template uses the API Post URL and the API Key to get the web service schema and generate a web app. Configurable UI: The setting.aspx page allows you to configure the parameter name, default values and ranges, and include or exclude output parameters in the response. Phone and tablet friendly: the generated web app uses Responsive Design to auto-adjust to smaller browsers.

Useful Links      Microsoft  
Learn More

Find more web apps

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

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with a "New" button and a search bar. Below it are sections for "MARKETPLACE", "Compute", "Networking", "Storage", "Web + Mobile", "Databases", "Intelligence + analytics", "Internet of Things", "Enterprise Integration", "Security + Identity", "Developer tools", "Monitoring + management", "Add-ons", and "Containers". Under "RECENT", there are entries for "Function App", "Bot Service (preview)", "Data Science Virtual Machine", "Data Lake Analytics", and "Power BI Embedded". The main panel is titled "Azure ML Request-Response Service Web App" and contains fields for "App name" (set to "ucllabml"), "Subscription" (selected), "Resource Group" (set to "Academic"), and "App Service plan/Location" (set to "amkyatenichoPlan(North Europe)"). There are also "Application Insights" settings ("On") and a "Create" button at the bottom. A "Pin to dashboard" checkbox is checked.

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

The screenshot shows the Microsoft Azure portal interface for an App Service named 'ucllabml'. The left sidebar contains a navigation menu with various icons and links such as Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quickstart, Deployment credentials, Deployment slots, Deployment options, Continuous Delivery (Preview), Application settings, Authentication / Authorization, Backups, Custom domains, and SSL certificates. The main content area is titled 'Essentials' and displays the following information:

- Resource group: Academic
- Status: Running
- Location: North Europe
- Subscription name: Andrew Fryer cross charged DX subscription
- Subscription ID: b701cd95-2320-40ed-951d-c048a77f2cc2
- URL: <http://ucllabml.azurewebsites.net>
- App Service plan/pricing tier: amykatenichoPlan (Free)
- External Repository Project: <https://github.com/raymondlaghaeian/Azu...>

The 'Monitoring' section shows a chart titled 'Requests and errors' with a Y-axis from 0 to 100 and an X-axis from 11:45 AM to 12:30 PM. The chart shows 0 errors and 0 requests. Below the chart, there are two large red and blue numbers '0'.

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.

Quickstart    Dashboard    Batch Request Log    Configure    **Consume**    Test    Swagger API

[← Web Services](#)

## UCL ML Studio Lab [Predictive Exp.]

A web service that predicts the price of a car given features such as type, fuel consumption, size and age.

Web service consumption options




Excel 2013 or later    Excel 2010 or earlier

Basic consumption info

Want to see how to consume this information? [Check out this easy tutorial.](#)

|                  |                                                                                                                                                                                                          |                                                                                     |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| Primary Key      | <input type="text" value="https://europewest.services.azureml.net/subscriptions/b701cd95232040ed951dc048a77f2cc2/services/959fd395ef704ea49cfb7f203c853858/execute?api-version=2.0&amp;format=swagger"/> |  |
| Secondary Key    | <input type="text" value="https://europewest.services.azureml.net/subscriptions/b701cd95232040ed951dc048a77f2cc2/services/959fd395ef704ea49cfb7f203c853858/execute?api-version=2.0&amp;format=swagger"/> |  |
| Request-Response | <input type="text" value="https://europewest.services.azureml.net/subscriptions/b701cd95232040ed951dc048a77f2cc2/services/959fd395ef704ea49cfb7f203c853858/jobs?api-version=2.0"/>                       |  |
| Batch Requests   | <input type="text" value="https://europewest.services.azureml.net/subscriptions/b701cd95232040ed951dc048a77f2cc2/services/959fd395ef704ea49cfb7f203c853858/jobs?api-version=2.0"/>                       |  |

Documentation    Documentation

Paste them into the web app configuration as below


Web App Configuration

### Web Service Info ▾

**API Post URL:** [?](#)

**API Key:** [?](#)

Press Submit to load input parameters
**Submit**

Powered by **Azure Machine Learning**

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

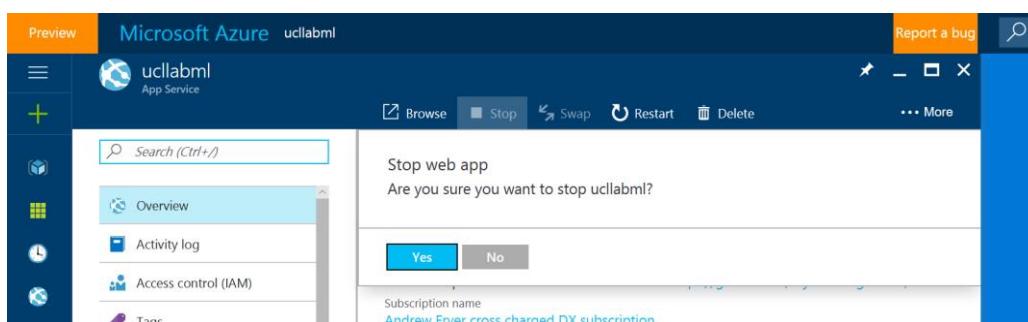


## UCL ML Studio Lab [Predictive Exp.]

### Input1 Parameters

|                                                                                                                                                         |                                                                                                                                                         |
|---------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Normalized-Losses</b><br><input style="width: 100%; height: 15px; border: 1px solid #ccc; margin-bottom: 5px;" type="range" value="50"/> 0 ————— 100 | <b>Engine-Type</b><br><input style="width: 100%; height: 30px; border: 1px solid #ccc; margin-bottom: 10px;" type="text"/>                              |
| <b>Make</b><br><input style="width: 100%; height: 30px; border: 1px solid #ccc; margin-bottom: 10px;" type="text"/>                                     | <b>Num-Of-Cylinders</b><br><input style="width: 100%; height: 15px; border: 1px solid #ccc; margin-bottom: 5px;" type="range" value="50"/> 0 ————— 100  |
| <b>New Fuel Type</b><br><input style="width: 100%; height: 30px; border: 1px solid #ccc; margin-bottom: 10px;" type="text"/>                            | <b>Engine-Size</b><br><input style="width: 100%; height: 15px; border: 1px solid #ccc; margin-bottom: 5px;" type="range" value="50"/> 0 ————— 100       |
| <b>Aspiration</b><br><input style="width: 100%; height: 30px; border: 1px solid #ccc; margin-bottom: 10px;" type="text"/>                               | <b>Fuel-System</b><br><input style="width: 100%; height: 30px; border: 1px solid #ccc; margin-bottom: 10px;" type="text"/>                              |
| <b>Num-Of-Doors</b><br><input style="width: 100%; height: 15px; border: 1px solid #ccc; margin-bottom: 5px;" type="range" value="50"/> 0 ————— 100      | <b>Bore</b><br><input style="width: 100%; height: 15px; border: 1px solid #ccc; margin-bottom: 5px;" type="range" value="50"/> 0 ————— 100              |
| <b>Body-Style</b><br><input style="width: 100%; height: 30px; border: 1px solid #ccc; margin-bottom: 10px;" type="text"/>                               | <b>Stroke</b><br><input style="width: 100%; height: 15px; border: 1px solid #ccc; margin-bottom: 5px;" type="range" value="50"/> 0 ————— 100            |
| <b>Drive-Wheels</b><br><input style="width: 100%; height: 30px; border: 1px solid #ccc; margin-bottom: 10px;" type="text"/>                             | <b>Compression-Ratio</b><br><input style="width: 100%; height: 15px; border: 1px solid #ccc; margin-bottom: 5px;" type="range" value="50"/> 0 ————— 100 |
| <b>Engine-Location</b><br><input style="width: 100%; height: 30px; border: 1px solid #ccc; margin-bottom: 10px;" type="text"/>                          | <b>Horsepower</b><br><input style="width: 100%; height: 15px; border: 1px solid #ccc; margin-bottom: 5px;" type="range" value="50"/> 0 ————— 100        |
| <b>Wheel-Base</b><br><input style="width: 100%; height: 15px; border: 1px solid #ccc; margin-bottom: 5px;" type="range" value="50"/> 0 ————— 100        | <b>Peak-Rpm</b><br><input style="width: 100%; height: 15px; border: 1px solid #ccc; margin-bottom: 5px;" type="range" value="50"/> 0 ————— 100          |

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.