

Machine Learning

What is Azure Machine Learning?

Contents

The Problem Domain

First Time Setup

The Data

Data Pre-Processing

Training the Model

Optimising Classifiers

Creating a Scoring Experiment

Publish as a Web Service

Testing the Web Service

Publish to the Gallery

Conclusion

Introduction

First Time Setup

The Data

Data Processing

Training the Model

Comparing Classifiers

Publish as a Web Service

Visualise Results

Conclusion

Extra Resources

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).

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 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 in line in experiments 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 application such as a website, 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 flight delays in the USA and see how Python scripts and Jupyter Notebooks can be used to aid in the exploration and evaluation of the data. We’ll see how to publish the model by reviewing the sample code for a published experiment and testing the output from an Excel add-in.

The Problem Domain

During this lab, we will create and publish a model that predicts if a flight will be delayed depending on a range of flight details and weather data from freely available Federal Aviation Authority data in the USA:

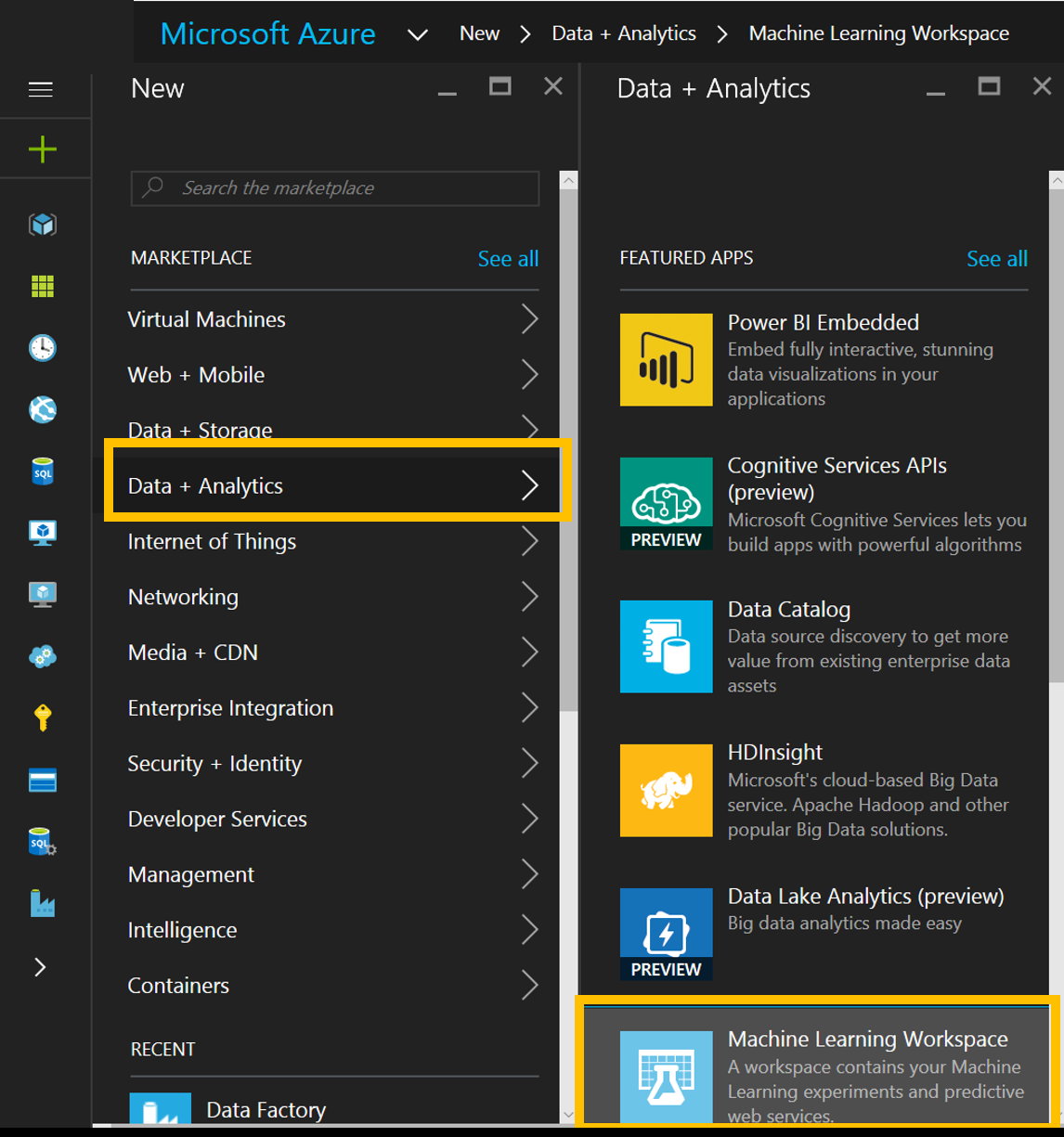
* The flight related columns include the date and time of a flight, the carrier ID (airline), origin and destination ID’s as well as delay time information.
* The weather data shows temperature wind speed visibility etc. at the departing airport when the flight departed.

The model we’ll create is a form of supervised learning so we will use historical flight and weather data to predict if a future flight is delayed. This model will use a binary (aka two class) classification technique by *classifying* the flights into two classes or groups - those that are delayed by more than 15 minutes and those that are on time. This information is in the sample data in the ArrDel15 (arrival delayed by more than 15 minutes) column where a 1 indicates a delayed flight and 0 represents an on time flight.

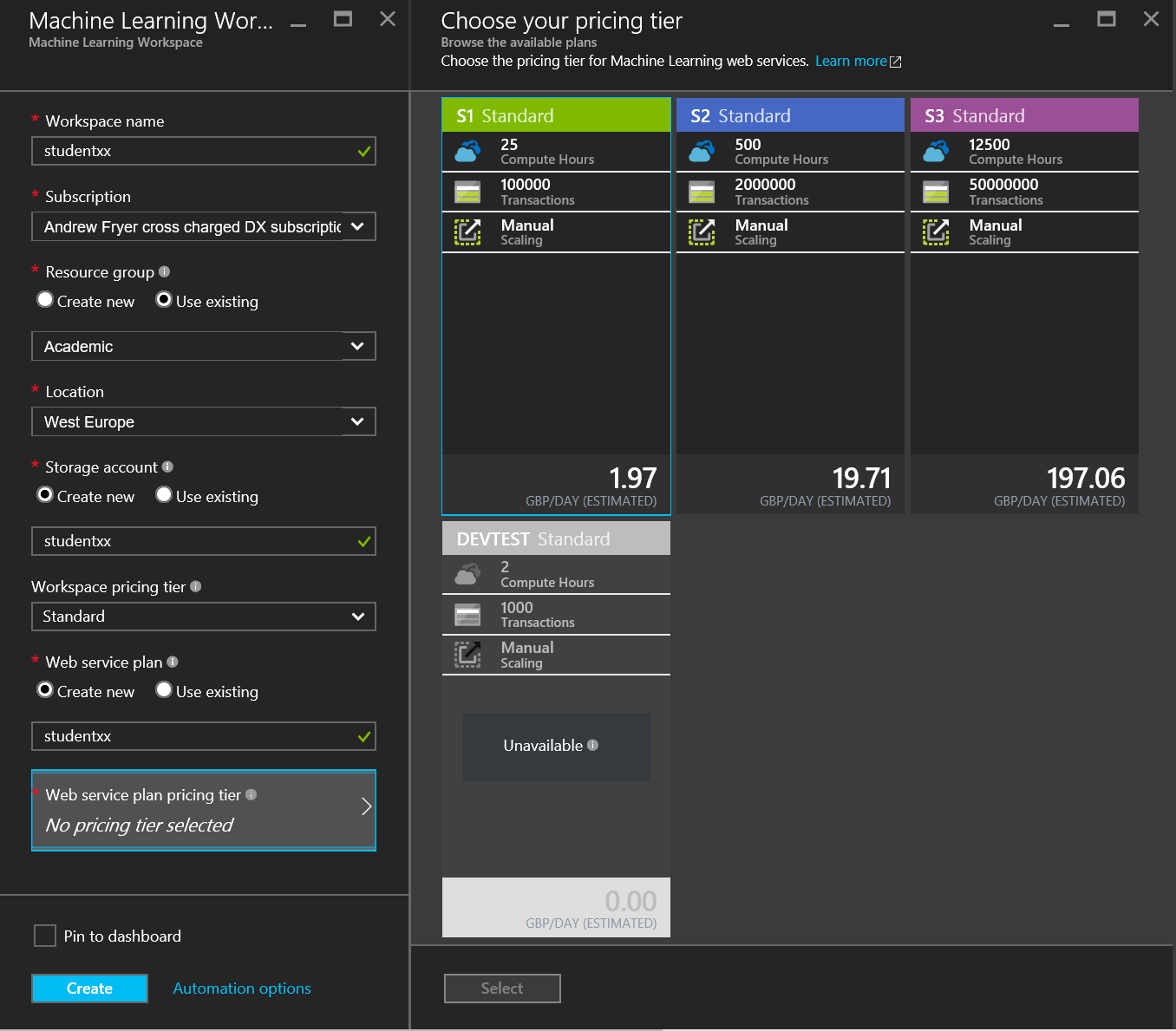
First Time Setup

Microsoft Azure Machine Learning offers a free-tier service and a standard tier for which you need an Azure subscription. In order to access the free workspace, go to <https://studio.azureml.net/> , sign in with a Microsoft account (@hotmail.com, @outlook.com etc.) and a free workspace is automatically created for you.

Each student will create a machine workspace to store their work in against a faculty run Azure Resource Group. Go to portal.azure.com which is the browser based management console for all services in Azure. To create your workspace click on the plus sign and look for data analytics-> Azure Machine Learning.



Fill out the form (called a blade in Azure) using the supplied name for the resource group and give your workspace an easily identifiable name and use the same name for the storage account and the web service plan. For the rest of the fields use the values and setting below:



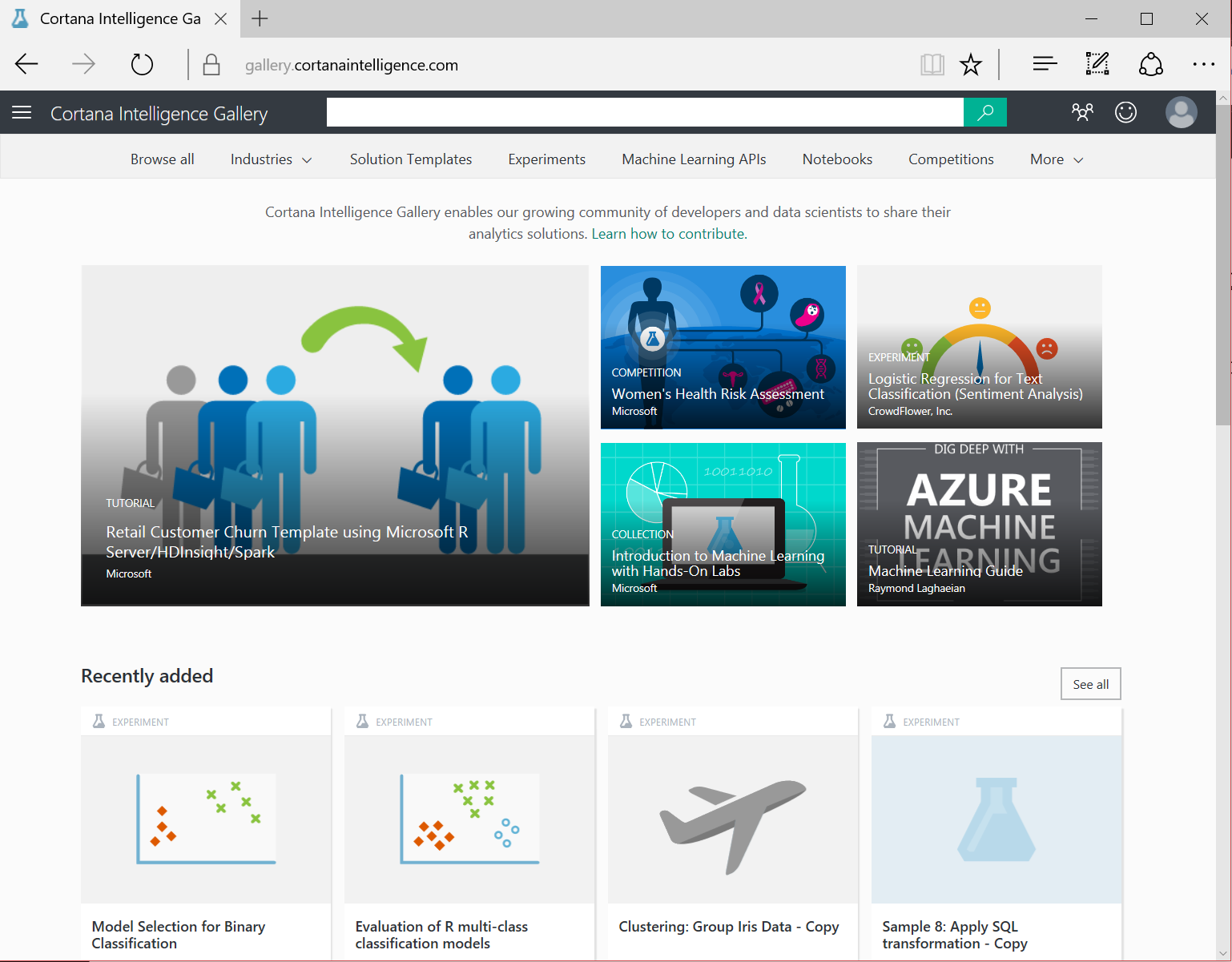
So workplace pricing tier is standard and use S1 standard for the pricing tier.

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:

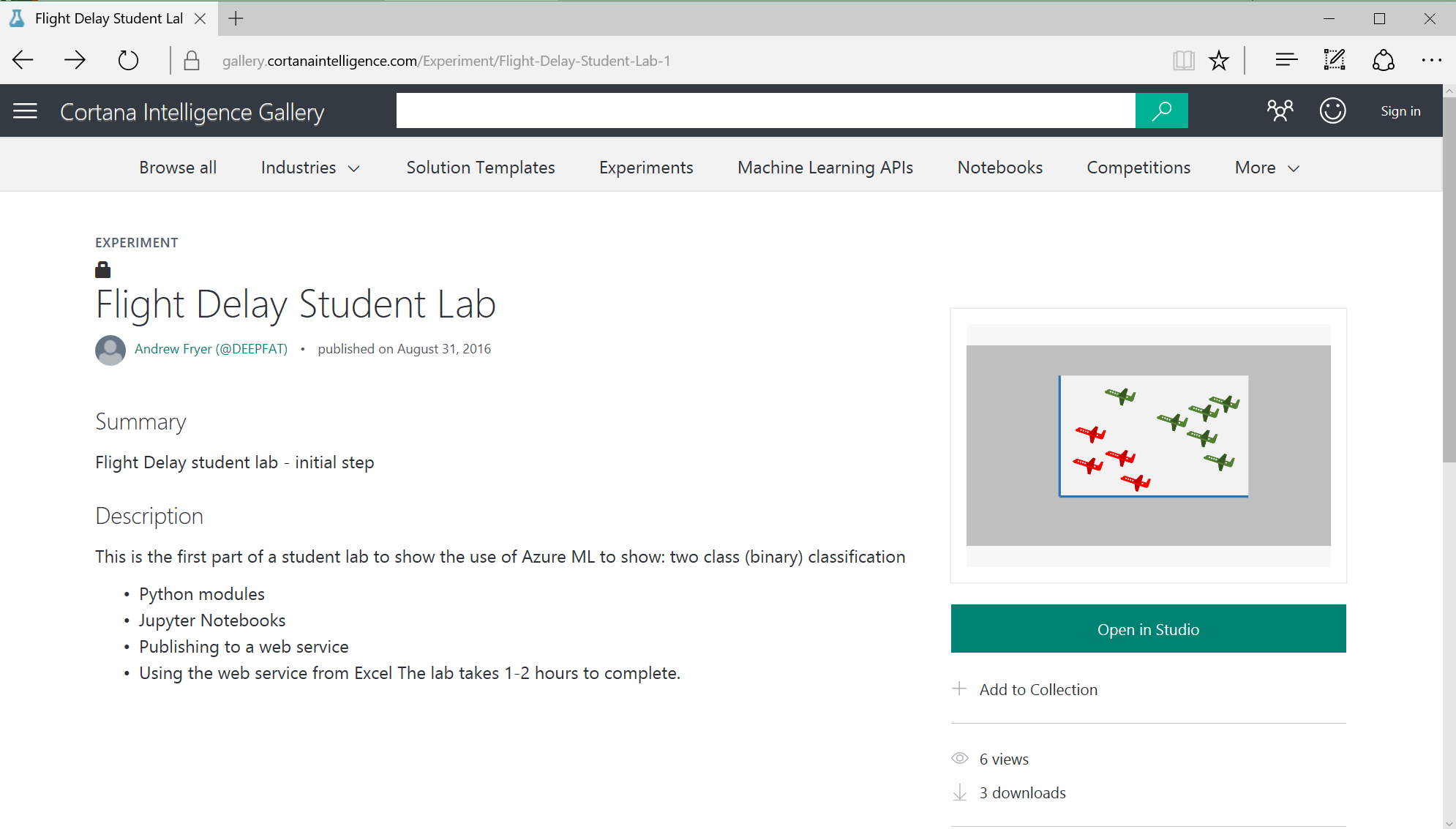
* Azure ML accepts many formats including CSV, TSV, Plain Text and Zip files
* Alternatively, you can use the ‘Import Data’ Module to access your data from sources such as SQL Azure, Blob storage, HD insight and industry standard OData feeds.

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

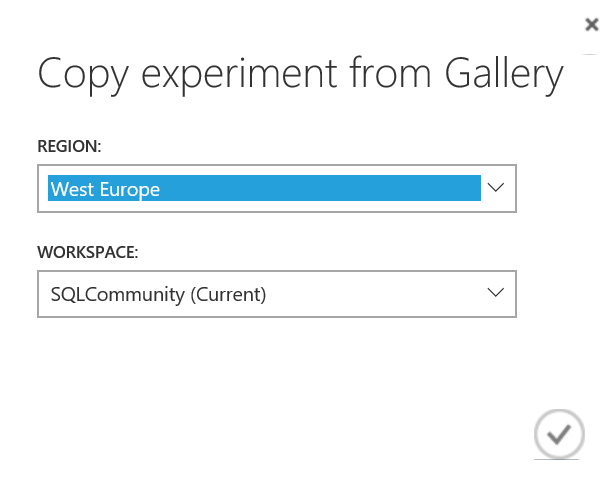


For this lab we will also use an existing sample which has the data in it already by going directly to

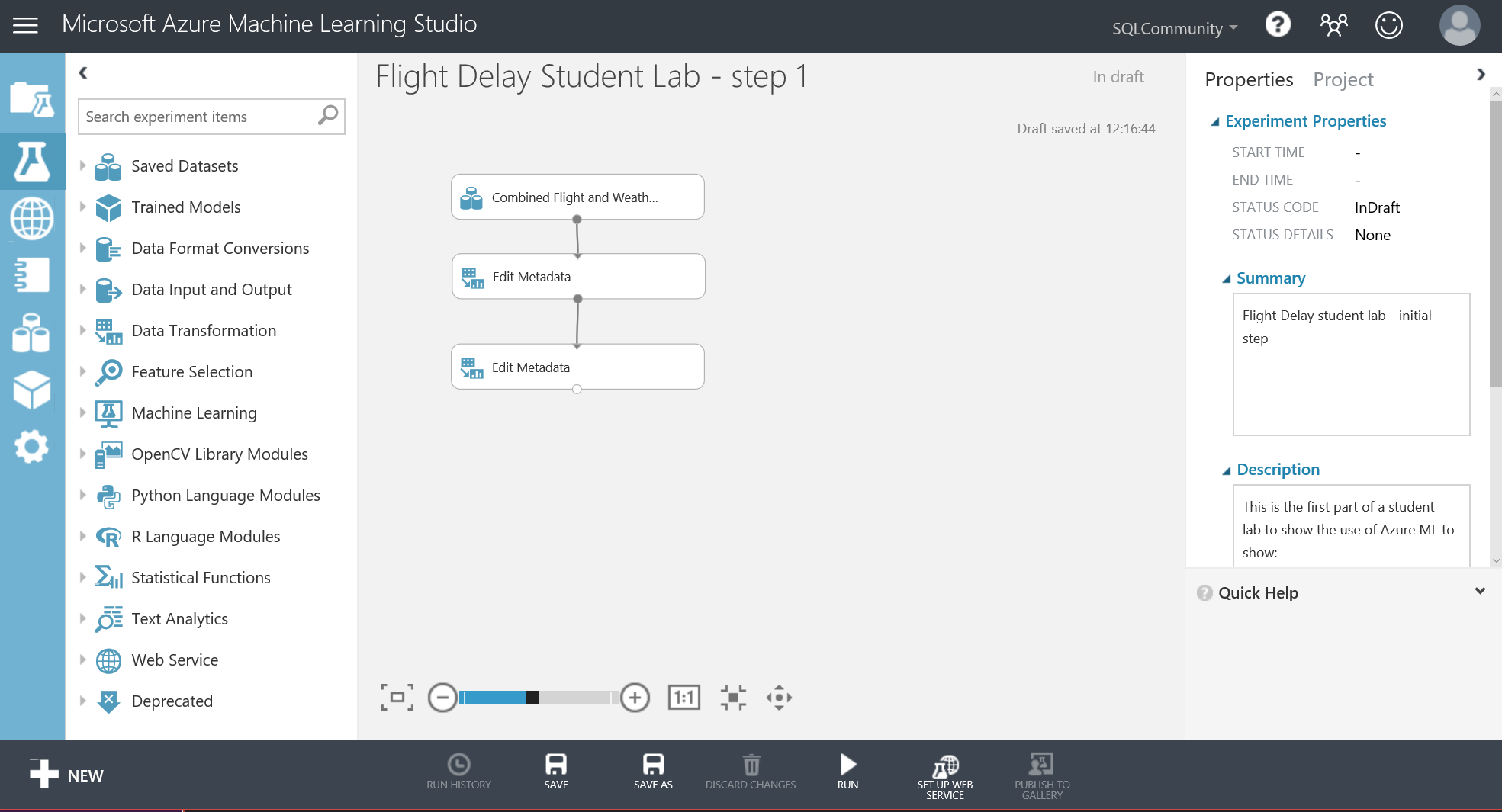
<https://gallery.cortanaintelligence.com/Experiment/Flight-Delay-Student-Lab-1>



Click on Open in Studio and confirm that a copy of the experiment will be saved in your workspace:



We can now see the sample experiment in ML studio as shown below, and from here we can review the tools we’ll use later:

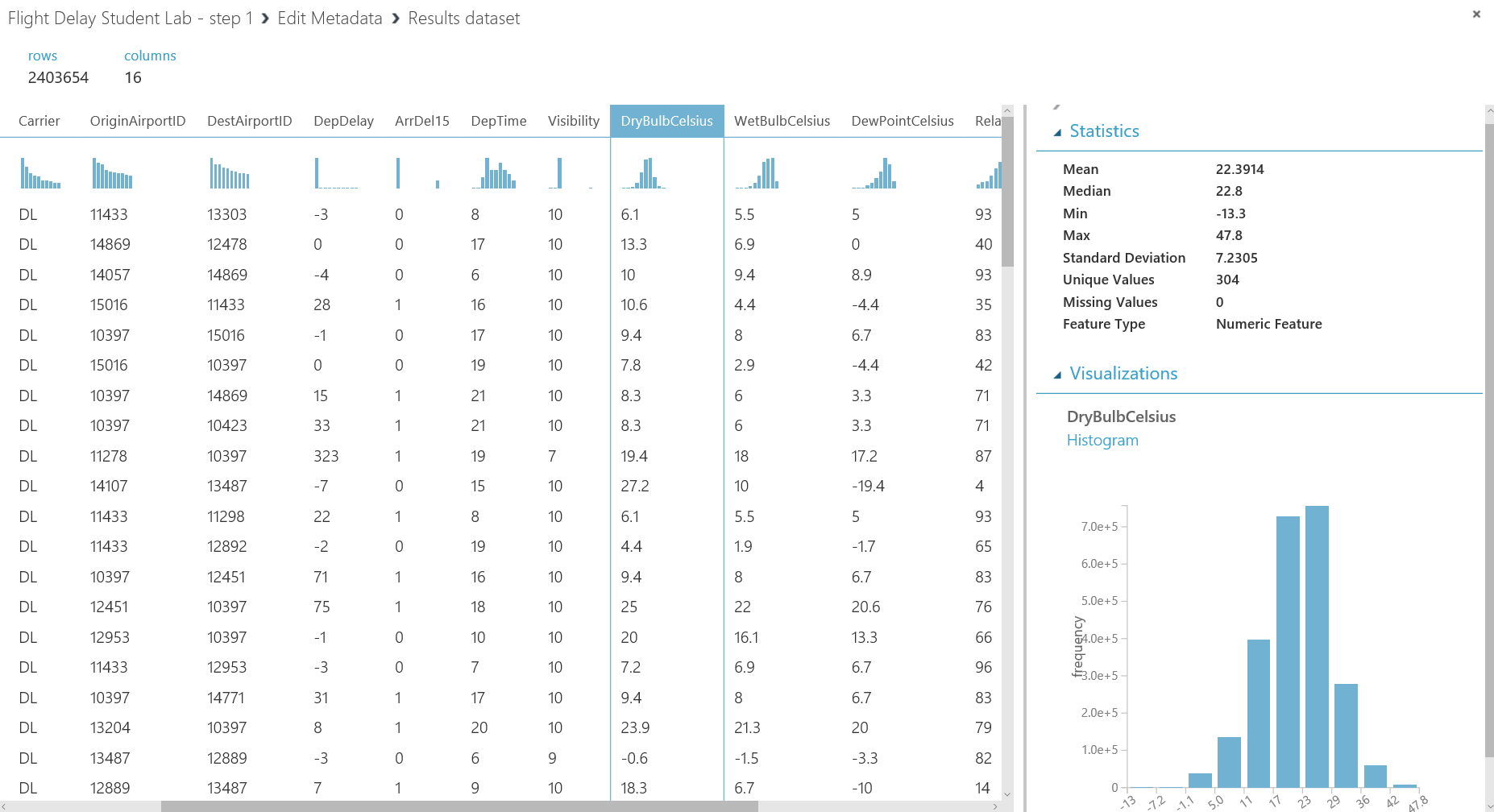


On the left in blue with white icons are the different objects we use in ML studio such as projects, Experiments, Web services, Jupyter Notebooks, Data Sets, Trained Models and Settings. Next on the left is a nested list of all the modules we can drag on to the design surface in the middle of the screen. On the right are the properties of the module that has focused or the whole experiment if no module is selected.

There are also functions we can perform in the bottom dark grey toolbar and the first thing we need to do is to click on Run to run the experiment as is.

The sample experiment has three modules, the top one represents the data we are working on and the grey lines connecting it to the two below represent the data flowing through the experiment. The other two Edit Metadata modules change the characteristics of the data; the top one declares the column ArrDel15 as the label – the thing we want to make predictions against and the second declares all the other columns as features, the columns to be used to make the prediction.

If we right click on the small circle at the bottom of the Combined Flight and Weather Dataset module and choose ‘Visualise’ we can quickly see the data we are working with. This view shows you the first few rows of the dataset and various statistics about each column in the dataset, and here we can see the statistics for the column Dry Bulb Celsius:



This data has already been pre-processed to make it suitable for machine learning:

* There are no missing values or duplicate rows in the data
* The data is correctly typed (as strings numeric etc.)
* There are no duplicates
* We have identified some features which should enable us to make good predictions

This is all part of feature engineering and cleaning the data and selecting the right features can be the biggest part of any machine learning project. There are good tools in Azure ML for this as well as the ability to use Python or R modules, but what we’ll do first is look at the ML process itself.

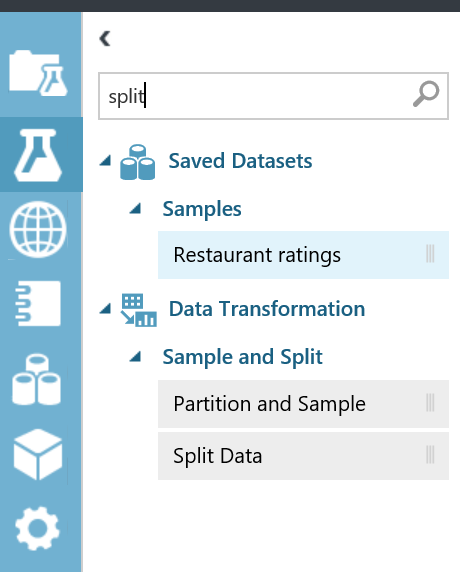
Training the Model

At this point in the experimentation process we are assuming we have a clean set of data that can be used to train and evaluate a model. We are moving from the business knowledge domain to the machine learning domain. We have identified that we are trying to predict the label (arrDel15) which can be 1 or 0 (two class classification) against a set of flight and weather data.

We will train a model against our data using one of the built in algorithms in Azure ML. Once the model has been trained it can be used to make predictions about whether a flight is late against flight that the model has not seen. In order to check how good the model is, we must evaluate the predicted answer against the original value (in the Arrdel15 column). This is just like any good science experiment - we want a control group to check the accuracy of the model.

To do this we split the data into two random sets and we’ll want to use most of the data (typically 80%) for training and keep back 20% for scoring. We’ll also want to stratify the split over the label to ensure that each of these separate groups of data have the same ration of values in the label i.e. the train score and evaluate sets of data will have the same ratio of on time to delayed flights.

In Azure ML we do this by using the built in split module. The easiest way to find this is to type split into the search box at the top of the list modules:

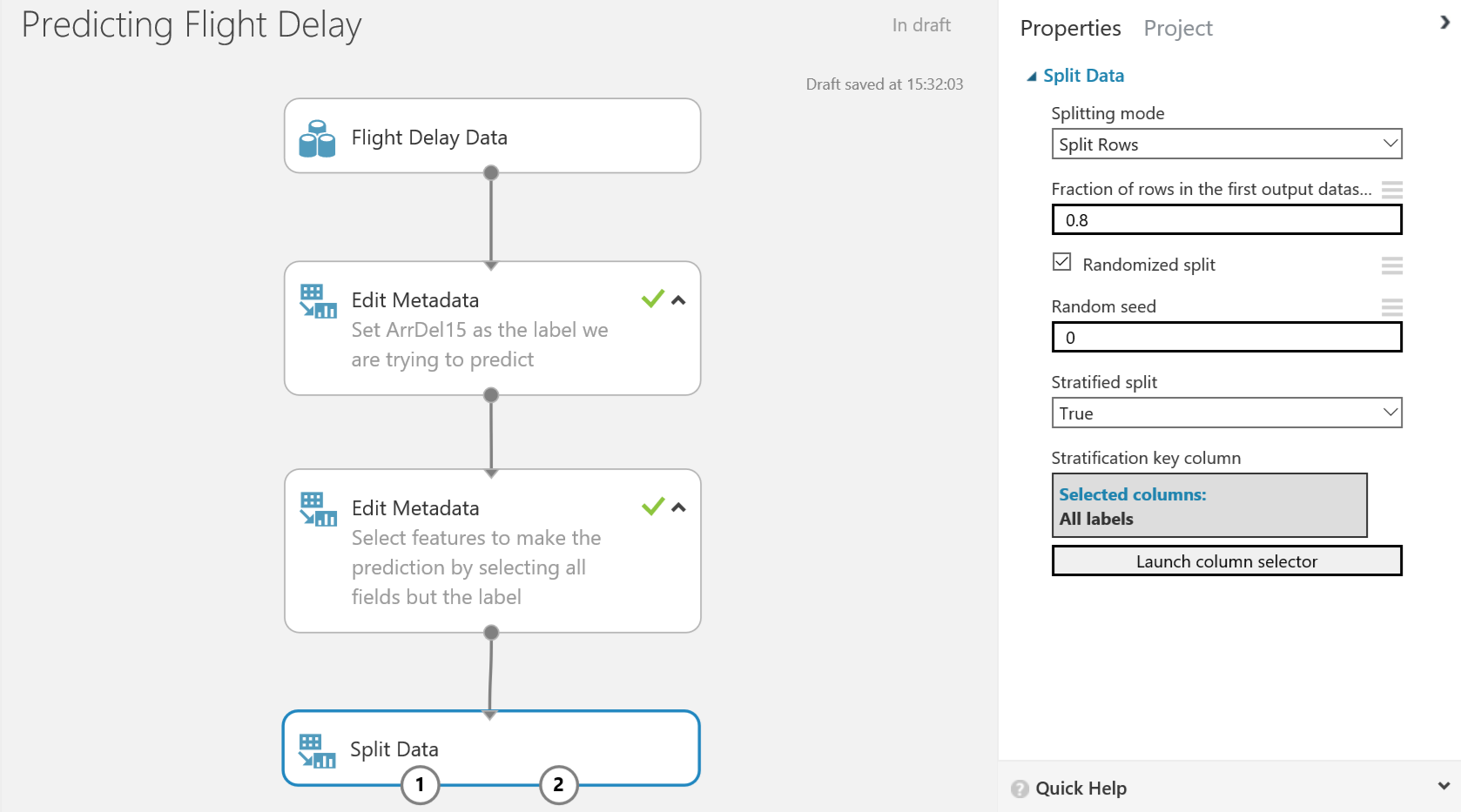


Now drag and drop the first split module onto the workspace. Connect this to the project columns module and set the ‘fraction of rows in the first output dataset’ field to 0.8 (80%).

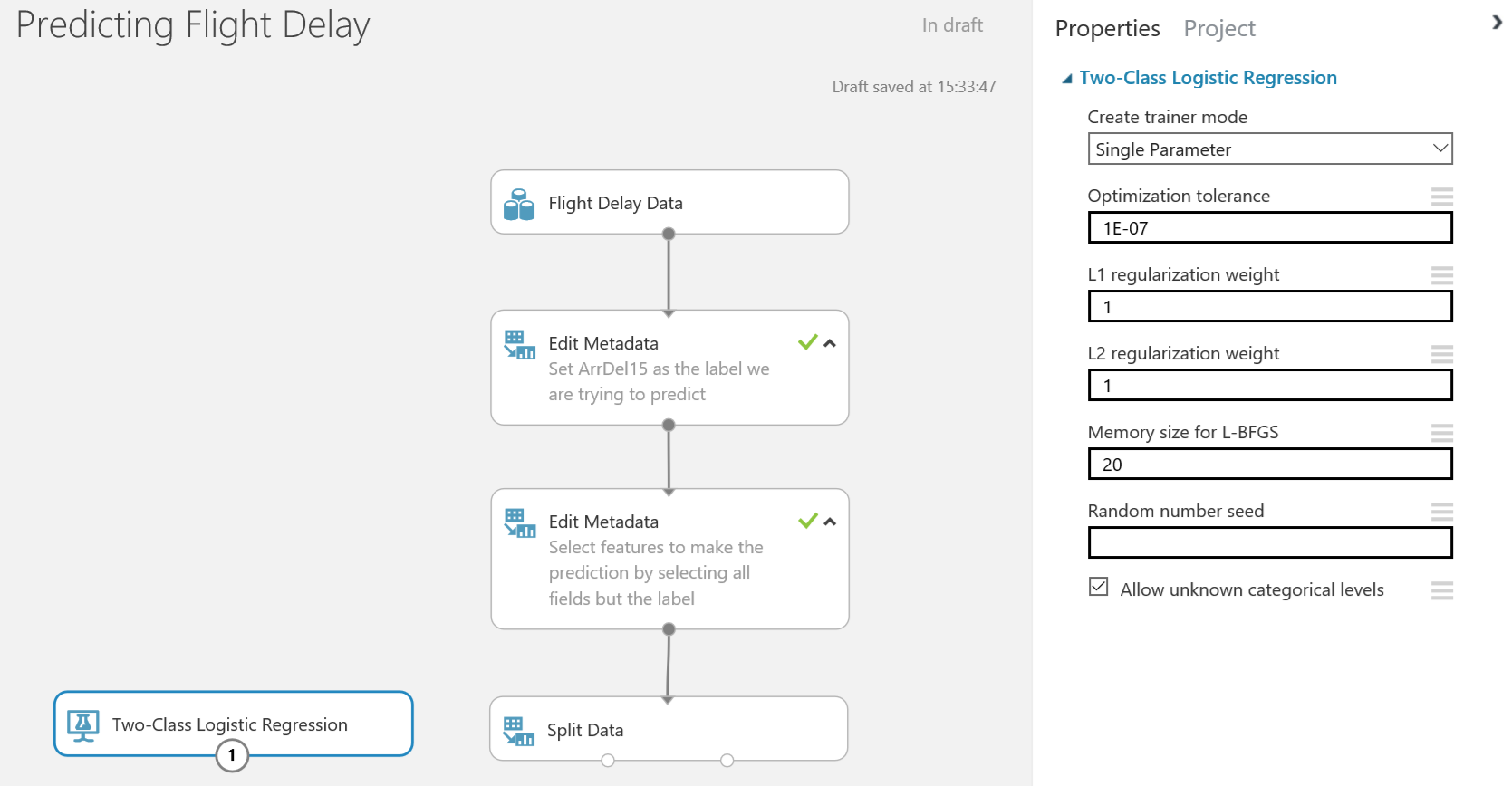
Set the “Stratified Split” option to True and set the stratification key column to all labels in this dialog box:



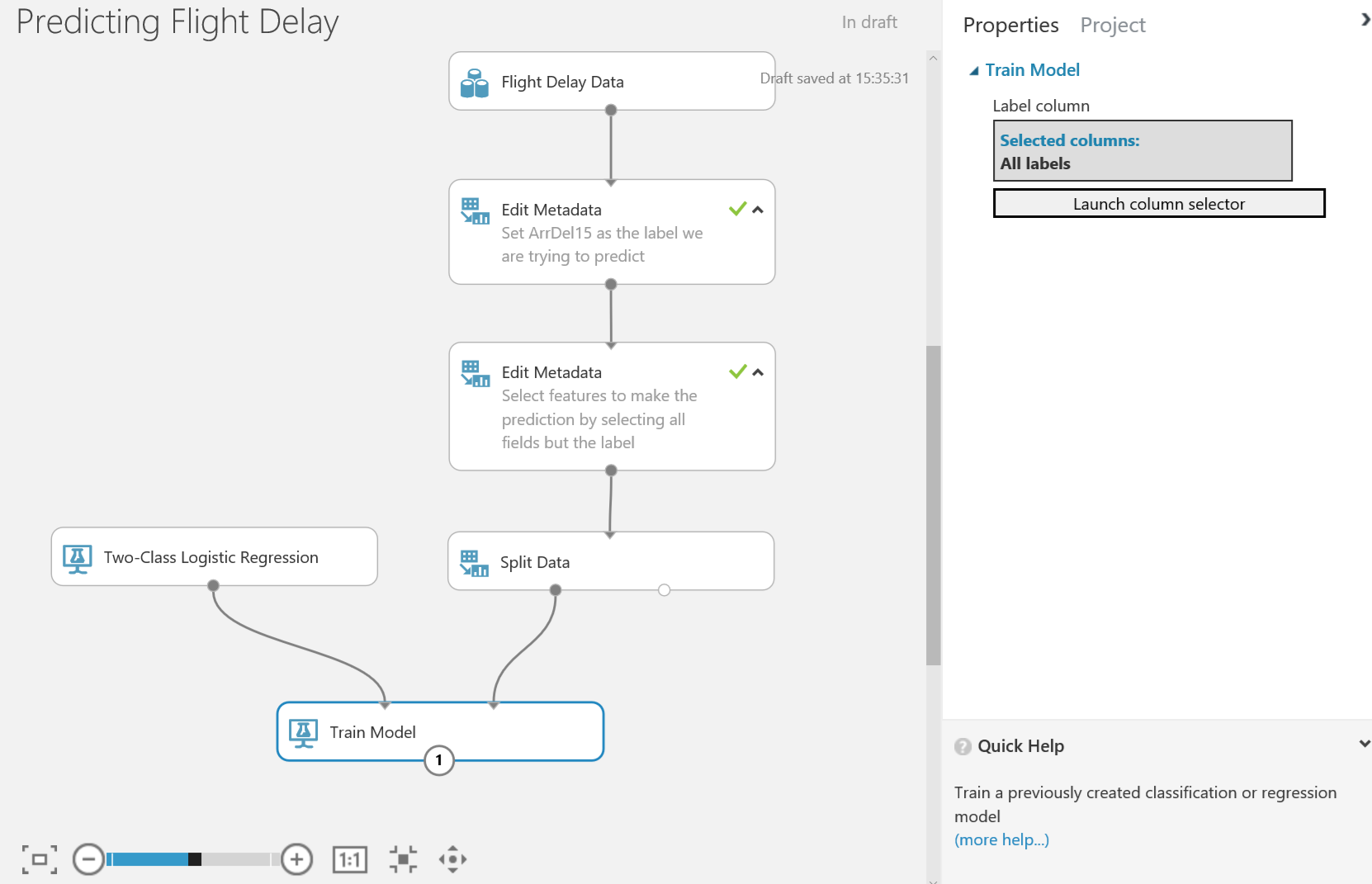
Select with rules, and then set the dropdown boxes to include and all labels. Click the checkbox to close the dialog to save these changes. The properties of the stratified split should now look like this:



After splitting the data, choose a classifier from the module list in the left hand menu. There are many classifiers depending on your prediction problem and the type of learning you want to use. In the accompanying deck with this lab there is a cheat sheet on which algorithm to use. For this lab we will use the ‘Two-Class Logistic Regression’ classifier module. Drag and drop this onto the workspace.

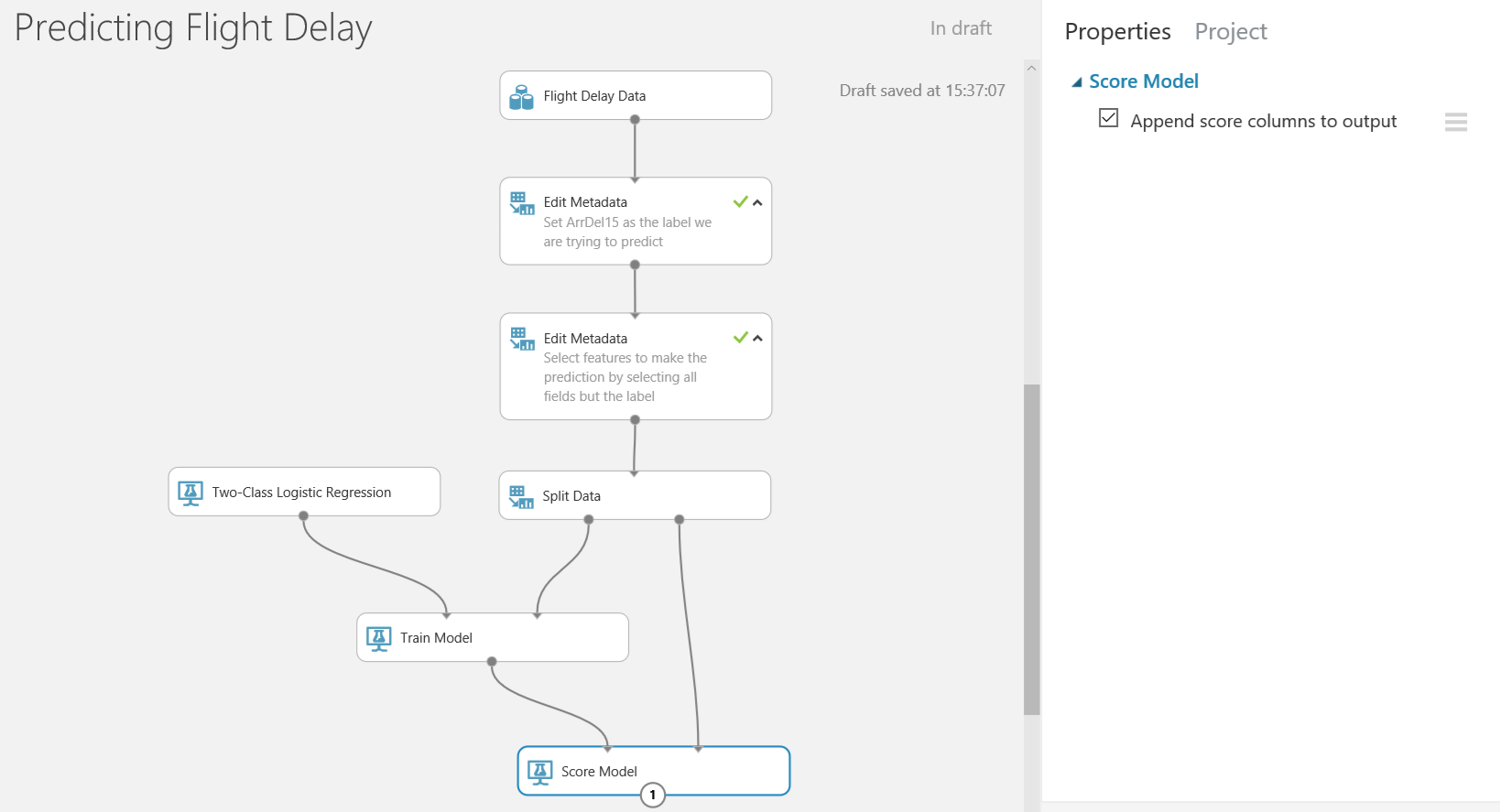


Next, drag and drop a ‘Train Model’ module onto the workspace. Connect the classifier to the left input port and connect the training data to the right input port.

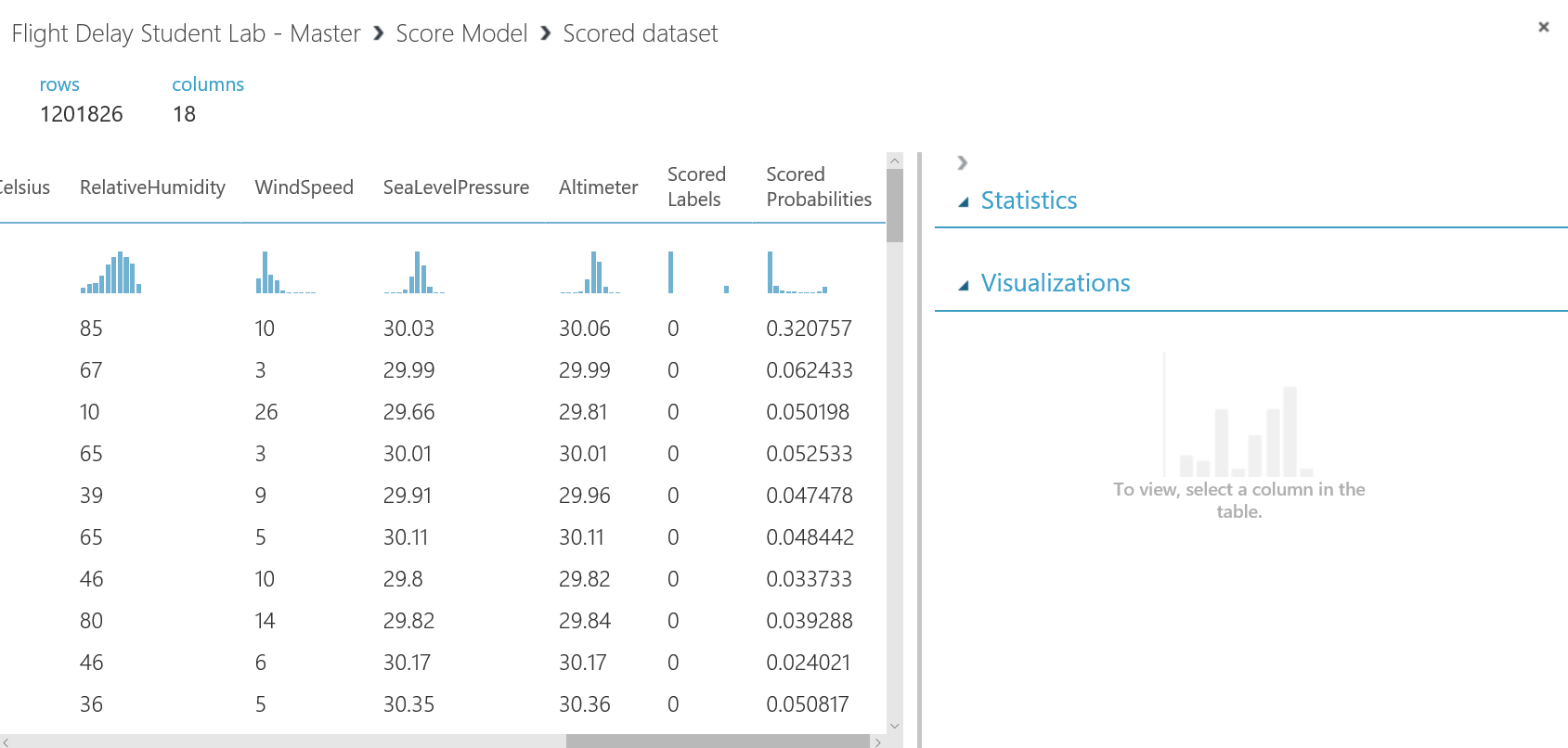


Select the Train module and look at the properties menu. Click on the ‘Launch Column Selector’ module and you can choose the column you want to learn (our label) by selecting ‘Include’ and the next drop down ‘All Labels’

Next use a ‘Score Model’ module to score the trained classifier against the test data. There are no parameters to set for the score module.

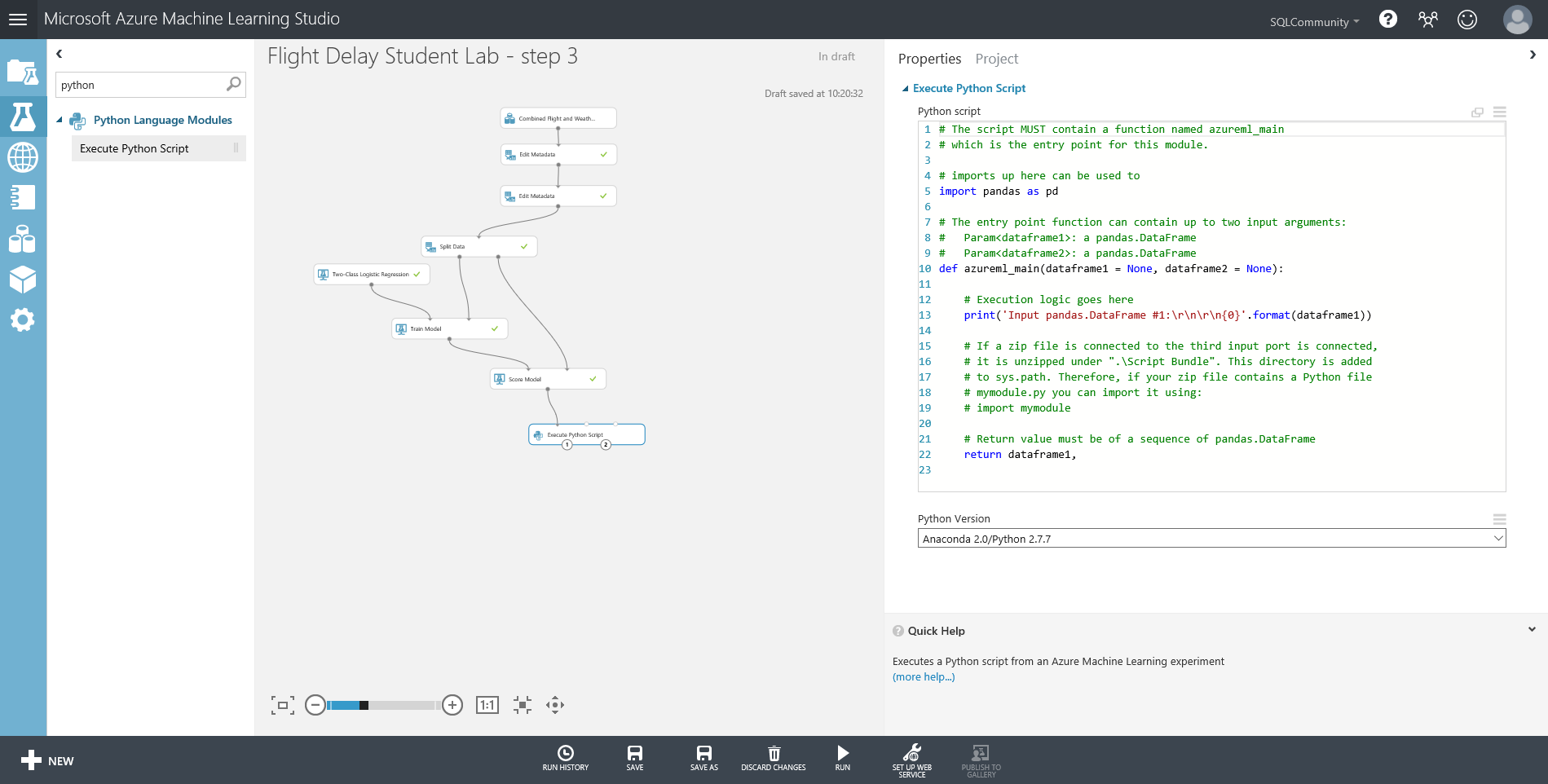


If we now run the experiment again by clicking on the run icon is on the toolbar at the bottom of ML studio we can see how well the model does by clicking on the small circle at the bottom of the Score Model once the experiment has run:



We can see the score experiment has added two new columns, Scored Labels and Scored Probabilities to the data set, the Scored Label is the prediction of the Arrdel15 column, and the Scored Probability is how confident the model is that the label is 1. If we scroll across we could compare the Scored Label with ArrDel15 for each of the rows but what we really need is way to evaluate the scores overall. Azure Ml does provide a module for the (Evaluate Module) but we could also add our own version by writing a Python script.

First we need to drag a Python module onto the design surface:



The Python module already has stub code in it load the data we are working on into dataframe1. We are going to plot a curve the [Receiver Operator Characteristic Curve](https://en.wikipedia.org/wiki/Receiver_operating_characteristic) (RoC)to get an idea of how our model is performing by replacing the sample code in the module with this script:

import matplotlib

matplotlib.use("TkAgg")

def azureml\_main(dataframe1 = None):

import sklearn.metrics as m

import matplotlib.pyplot as plt

dataframe1 = dataframe1.dropna()

# pick the label out of the dataframe and the positive label

r1 = m.roc\_curve(dataframe1["ArrDel15"], dataframe1["Scored Probabilities"], pos\_label= 1)

plt.plot(r1[0], r1[1], 'r-', label="Boosted Trees");

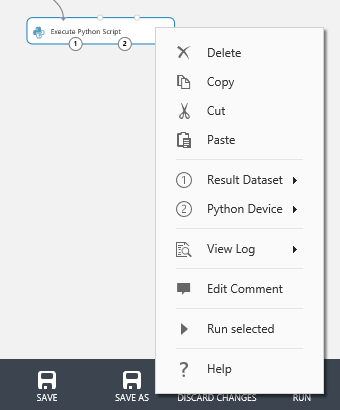
plt.grid("on");

plt.legend(loc="best")

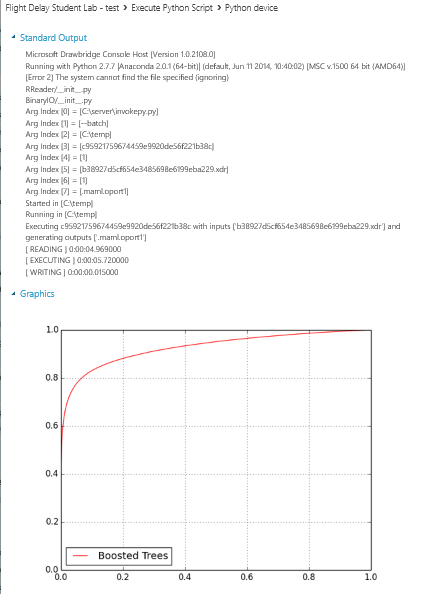
plt.savefig("roc.png")

return dataframe1,

To check this is working we can simply right click on the Python module and click run selected.



Once it has completed we can see the RoC curve by right clicking on output 2 (the one on the right) of the module: and click visualize



However, while the shape of this curve and the area under it are good indicators of accuracy if we want a numerical analysis of how well the model is working then we need a confusion matrix to show the

True positives, True negatives etc. and the standard scores that accompany these statistics:

* **Accuracy** = true Positives + True negatives over the overall total = 0.922
* **Recall** = true positives divided by true positives + false negatives = 0.684
* **Precision** = true positives divided by true positives + false positives = 0.913
* **F1 score** = .2 \* (precision\* recall)/ (precision + recall) = 0.782

We can add some more code to our python module to derive this information using the [scikit-learn](http://scikit-learn.org/stable/) library in Python. If we substitute all of this code in the Python module we’ll retain the plot we just created as well as these new measures:

import matplotlib

matplotlib.use("TkAgg")

import sklearn.metrics as m

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.metrics import confusion\_matrix as cm

def azureml\_main(dataframe1 = None):

dataframe1 = dataframe1.dropna()

# pick the label out of the dataframe and the positive label and plot the RoC curve

r1 = m.roc\_curve(dataframe1["ArrDel15"], dataframe1["Scored Probabilities"], pos\_label= 1)

plt.plot(r1[0], r1[1], 'r-', label="Boosted Trees");

plt.grid("on");

plt.legend(loc="best")

plt.savefig("roc.png")

#Derive test statistics to show the accuracy of the model and output on POrt1

cmarray = cm(dataframe1['ArrDel15'], dataframe1['Scored Labels'])

TrueNeg, FalsePos = cmarray[0]

FalseNeg, TruePos = cmarray[1]

TruePos = float(TruePos)

FalseNeg = float(FalseNeg)

FalsePos= float(FalsePos)

TrueNeg= float(TrueNeg)

Accuracy = (TruePos + TrueNeg) /(TruePos+ TrueNeg + FalsePos + FalseNeg)

Recall = TruePos /(TruePos + FalseNeg)

Precision = TruePos /(TruePos+ FalsePos)

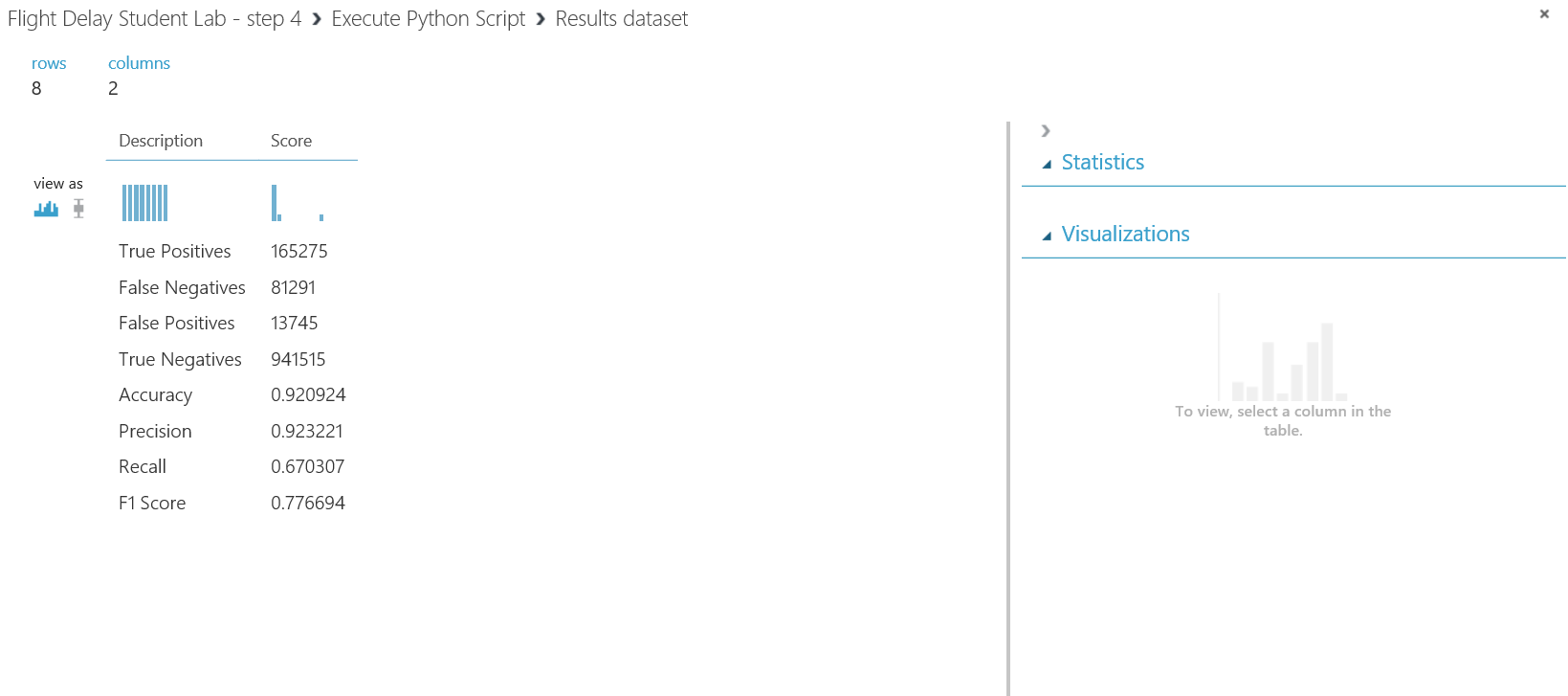
F1Score = 2 \* (Precision \* Recall)/(Precision + Recall)

data = {'Description': ['True Positives','False Negatives','False Positives','True Negatives','Accuracy','Precision','Recall','F1 Score'],'Score': [TruePos,FalseNeg,FalsePos,TrueNeg,Accuracy,Precision,Recall,F1Score]}

dataframe1 =pd.DataFrame(data,columns=['Description','Score'])

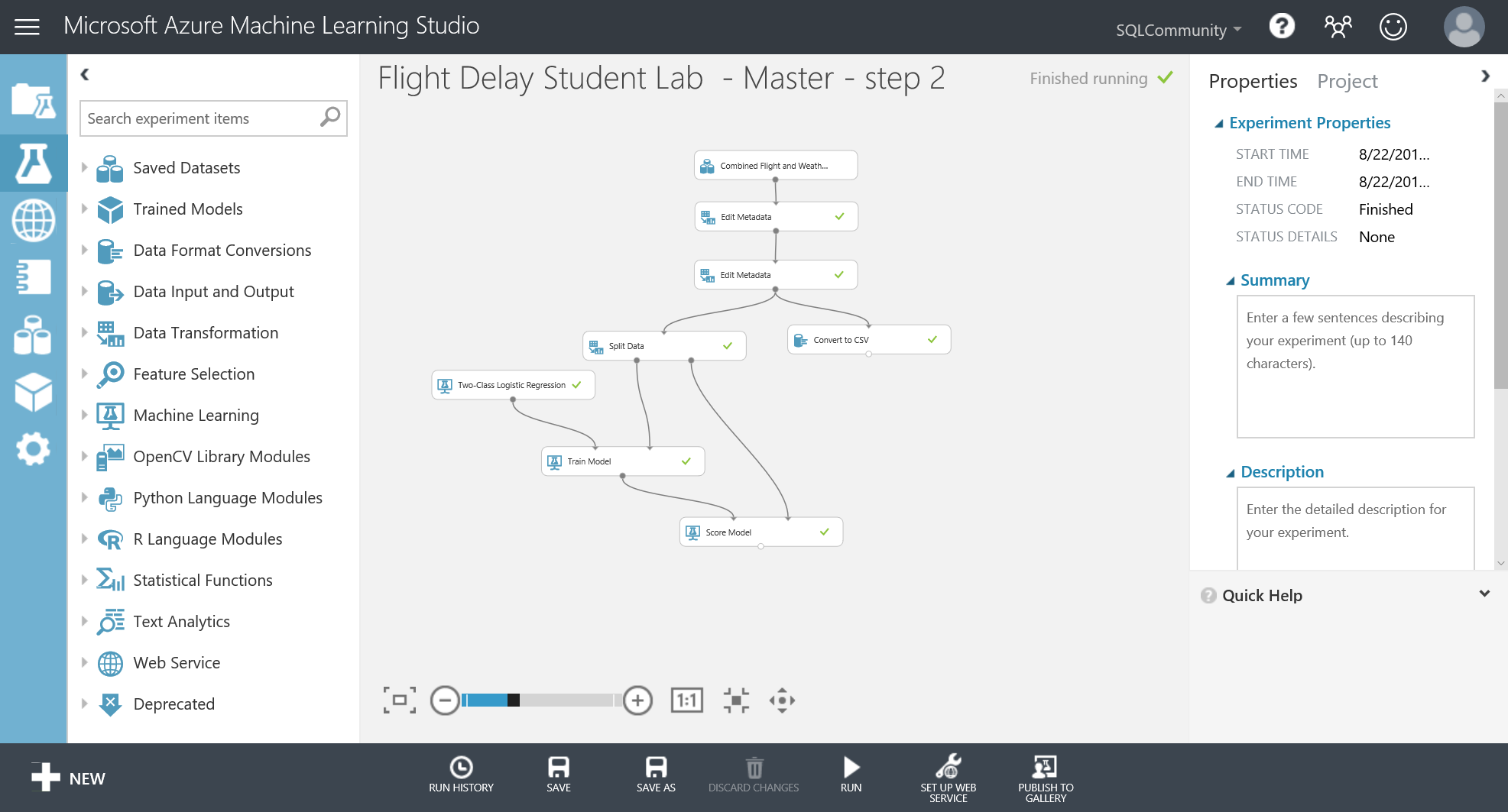
return dataframe1,

If we just rerun this module again we can see these new statistics by right clicking on output 1 (the left one) and selecting visualise again:



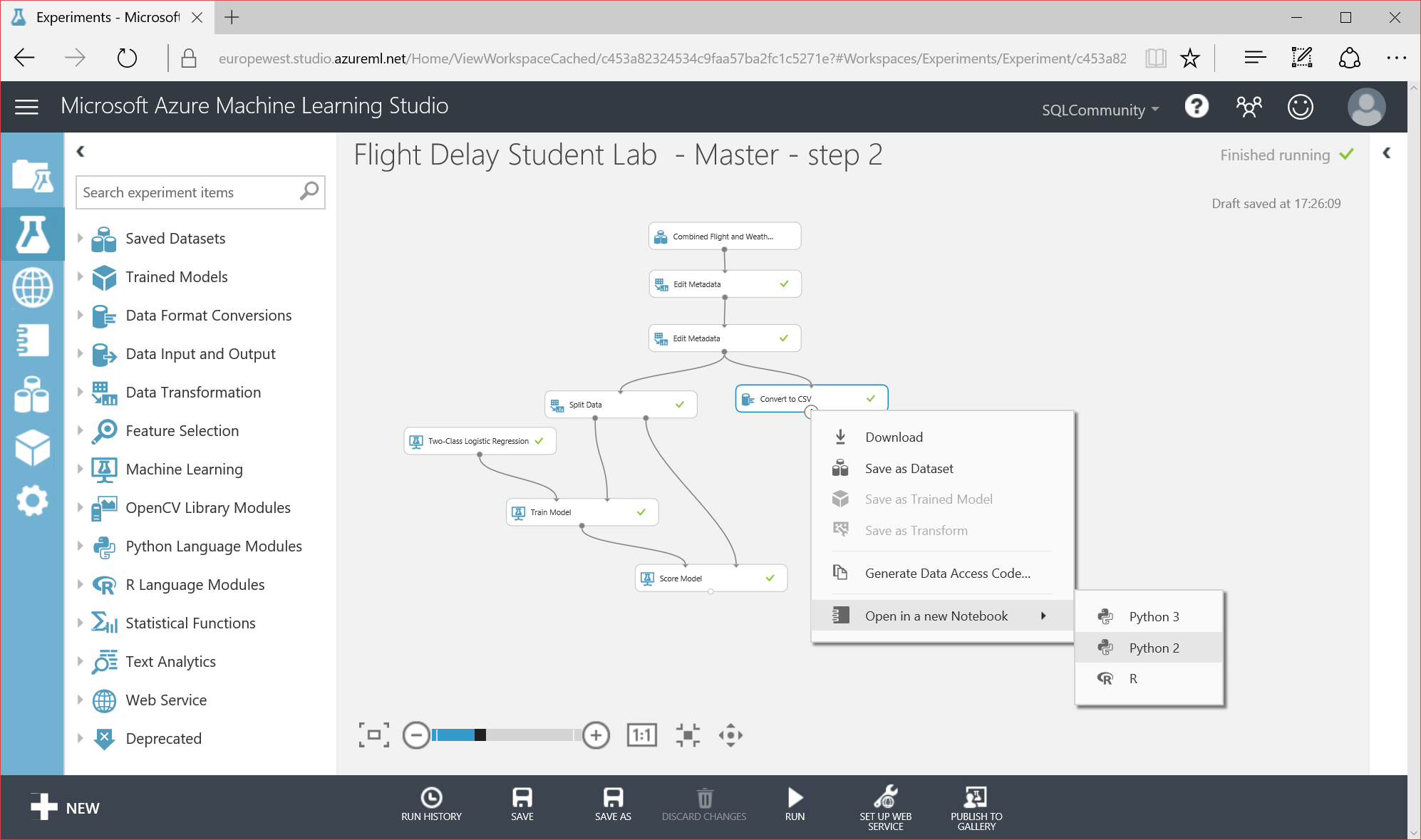
Jupyter Notebooks

Our experiment seems to be quite accurate, but we should be clear on why this so we are clear that the correlations used by the model make sense in business. We saw at the start of this lab that we can get basic statistics just by visualising the data and there are statistical modules to do this. However, we might wish to bring our own tools to bear and a great way for those with Python or R skills to do this is with Jupyter notebooks. These are ad hoc scripts which we can attach to our experiments or use on their own.

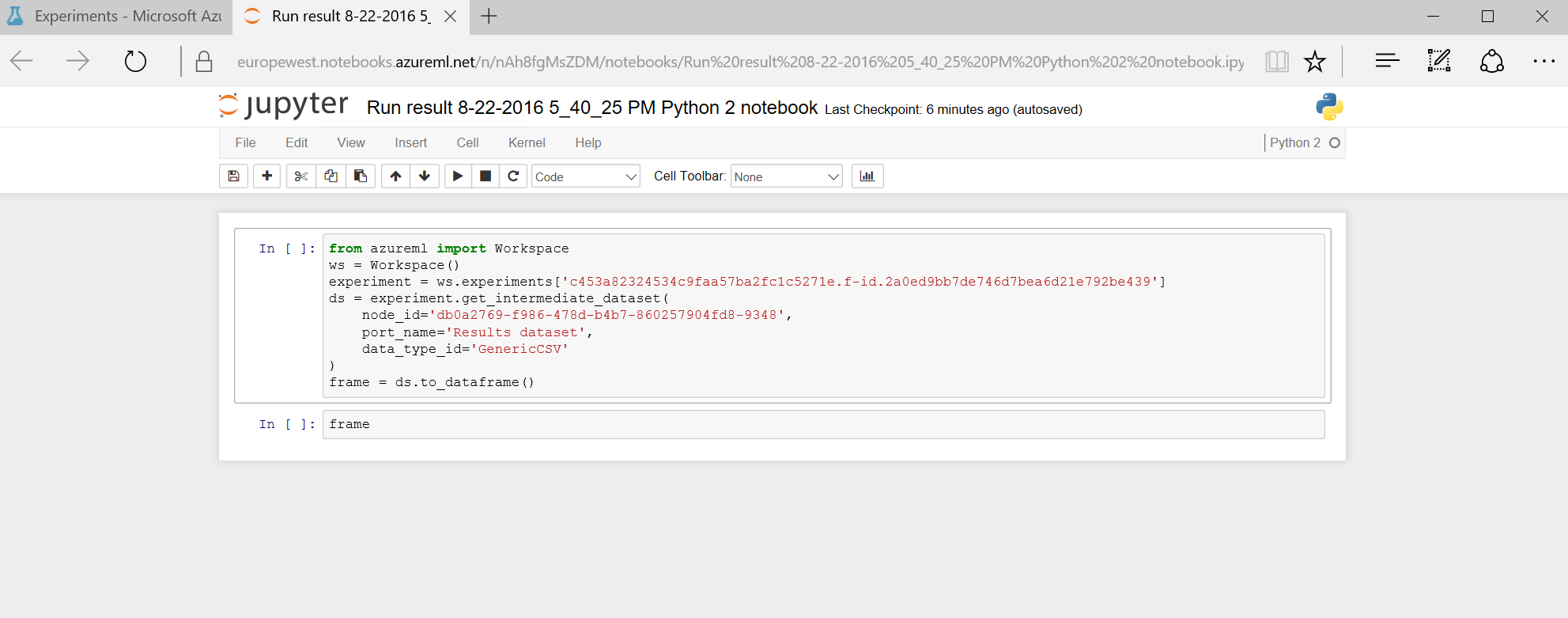
To use these notebooks, we need to extract the data to a format (.csv) that the notebooks can use and this can quickly be done with the export to CSV module which we’ll connect to the output of the second metadata module:

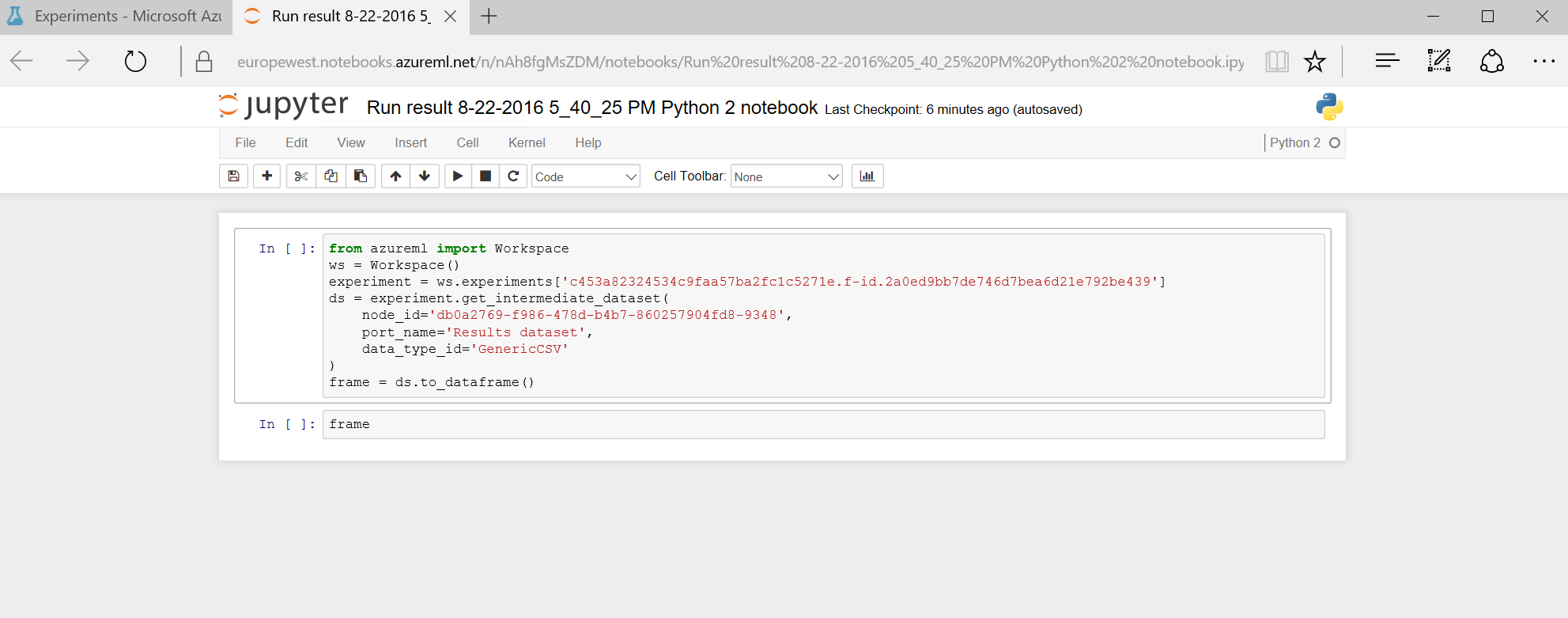
Right click on the Export to CSV module and click Run Selected to just run that module.

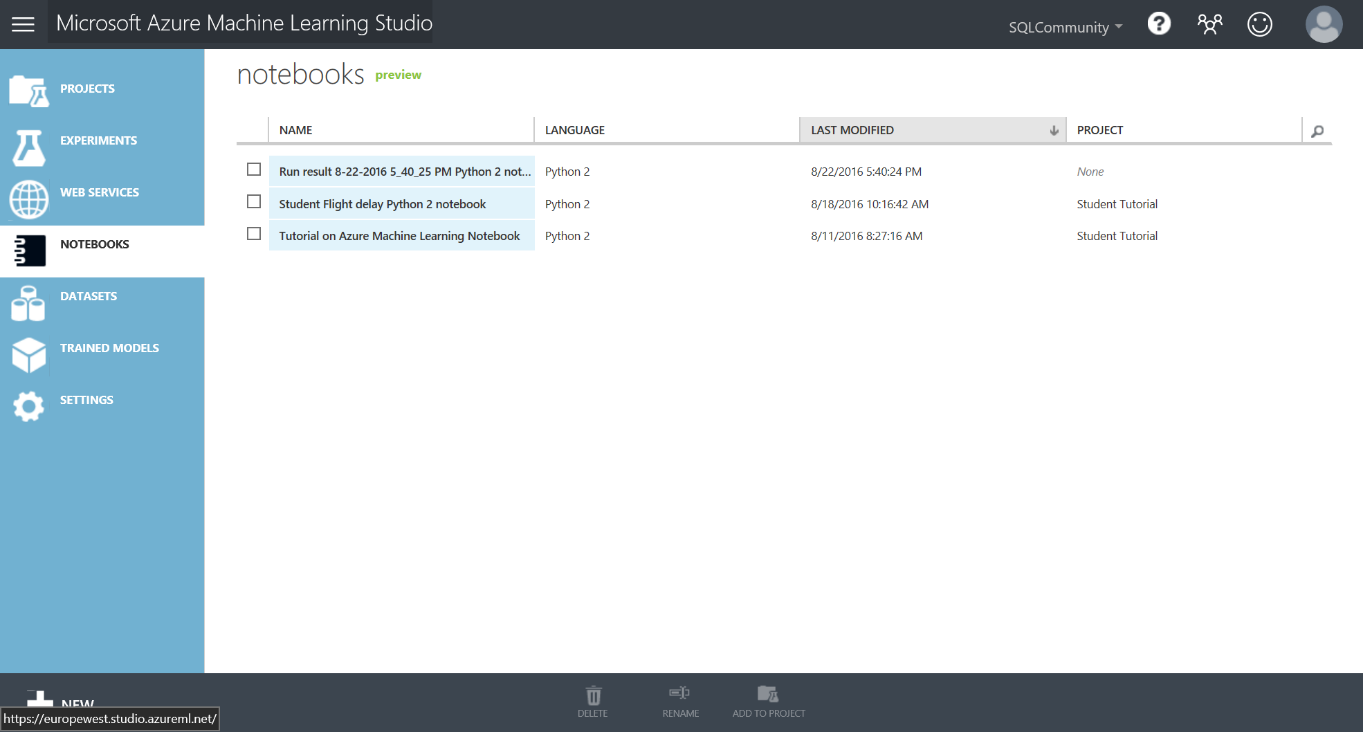
When it’s finished right click again and select Open in a New Notebook -> Python 2:

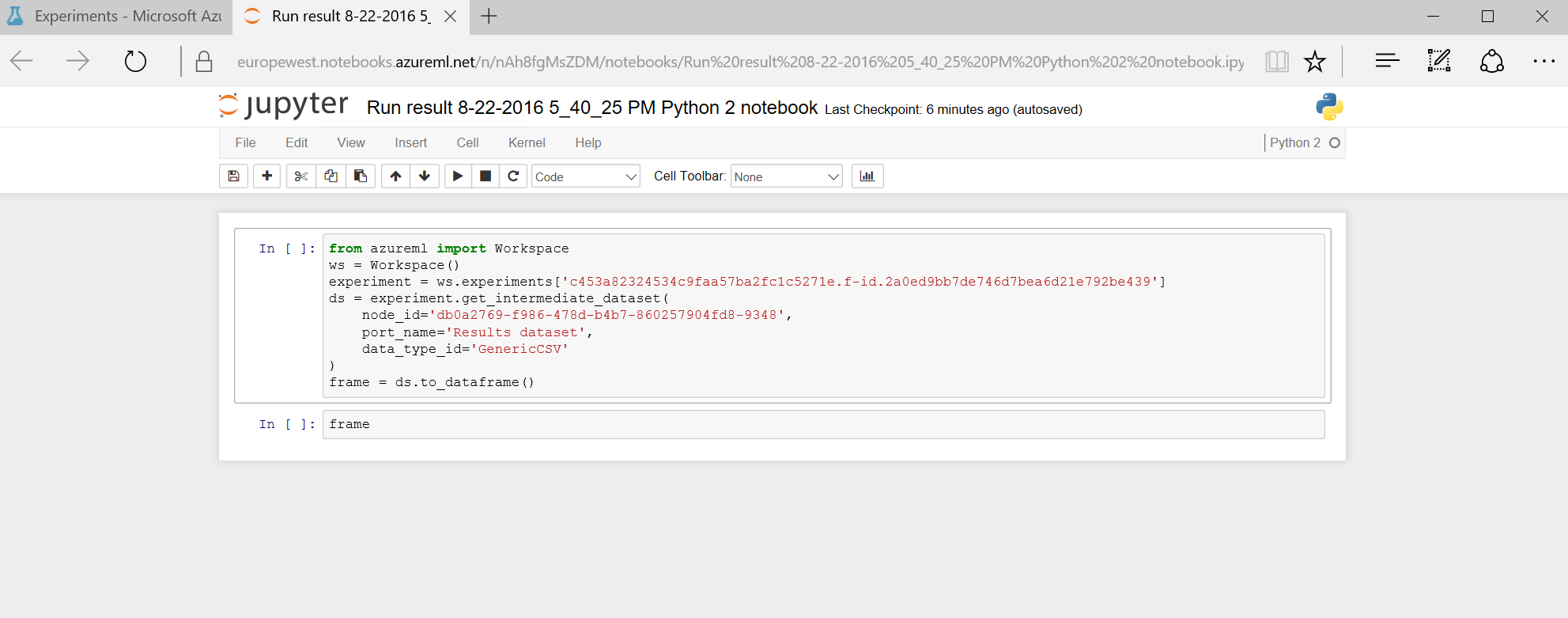


We’ll now be taken to a new browser window with our Notebook in:



Notice we already have a stub script in it which takes our data and loads it into a dataframe called frame. The first thing we should do is rename our notebook to something meaningful as this notebook will persist in our ML workspace. So rename it click on save and confirm this by going back to the browser tab for ML studio and clicking on the Notebook icon on the left (it’s not yet renamed in the screenshot below):

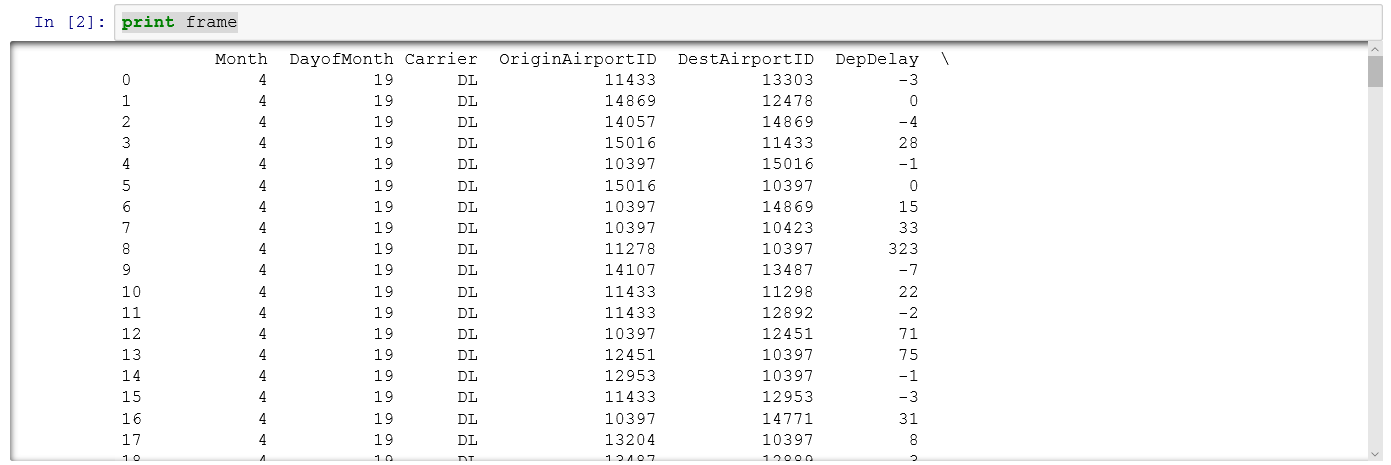


We can now go back to the Notebook tab and add some of our own code. Firstly, we need to run the cell with the existing code in to load the dataframe with data from our experiment which we do by setting focus on the cell and clicking on the run icon in the top toolbar.

In the second cell there is just the word frame and if we want to see a sample of our data we should change this to

print frame

and click the run icon to get back a sample of our data:



What we can do now is use any of the many libraries and function in Python to analyse this data without any further effort. In this case let’s see how the various features correlate with each other. This plot only works against numerical data to which statistics can be applied so we’ll need to exclude the carrier and use a separate data frame for this.

Create a new cell in the notebook and enter this code

correlationFrame = frame.drop("Carrier", axis=1)

print correlationFrame

and run this.

We can then use this to do our correlation plot but before we do we’ll need to bring in some libraries, so add a new cell and enter this:

import pip

import pandas

import numpy as np

import pandas as pd

import matplotlib

matplotlib.use("agg")

import matplotlib.pyplot as plt

#Now we have these libraries we can create the plot itself:

cm=correlationFrame.corr()

fig=plt.figure()

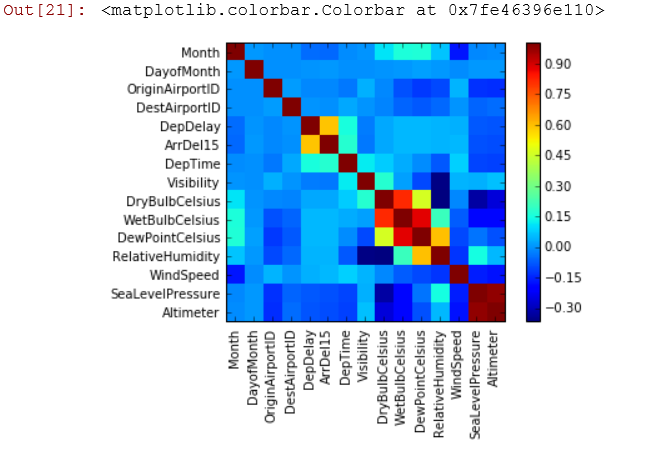
plt.imshow(cm,interpolation='nearest')

plt.xticks(list(range(0,len(cm.columns))),list(cm.columns.values), rotation=90)

plt.yticks(list(range(0,len(cm.columns))),list(cm.columns.values))

plt.colorbar()

If we run this now we’ll get a plot like this:



What this is showing us is how the various columns in our dataset are correlated. Some of this is obvious:

* The correlation on the leading diagonal is always one because month is perfectly correlated to month
* There is little correlation between month and the various temperature readings.

What we are interested is how each of these columns correlates with ArrDel 15 (the yellow box) and here we can see there is a strong correlation (about 0.65) with DepDelay. This makes sense - if a plane takes off late there is more chance it will land late. There is also some negative correlation with Altimete and see level pressure which may require further investigation.

This analysis can then be used to refine our experiment for example

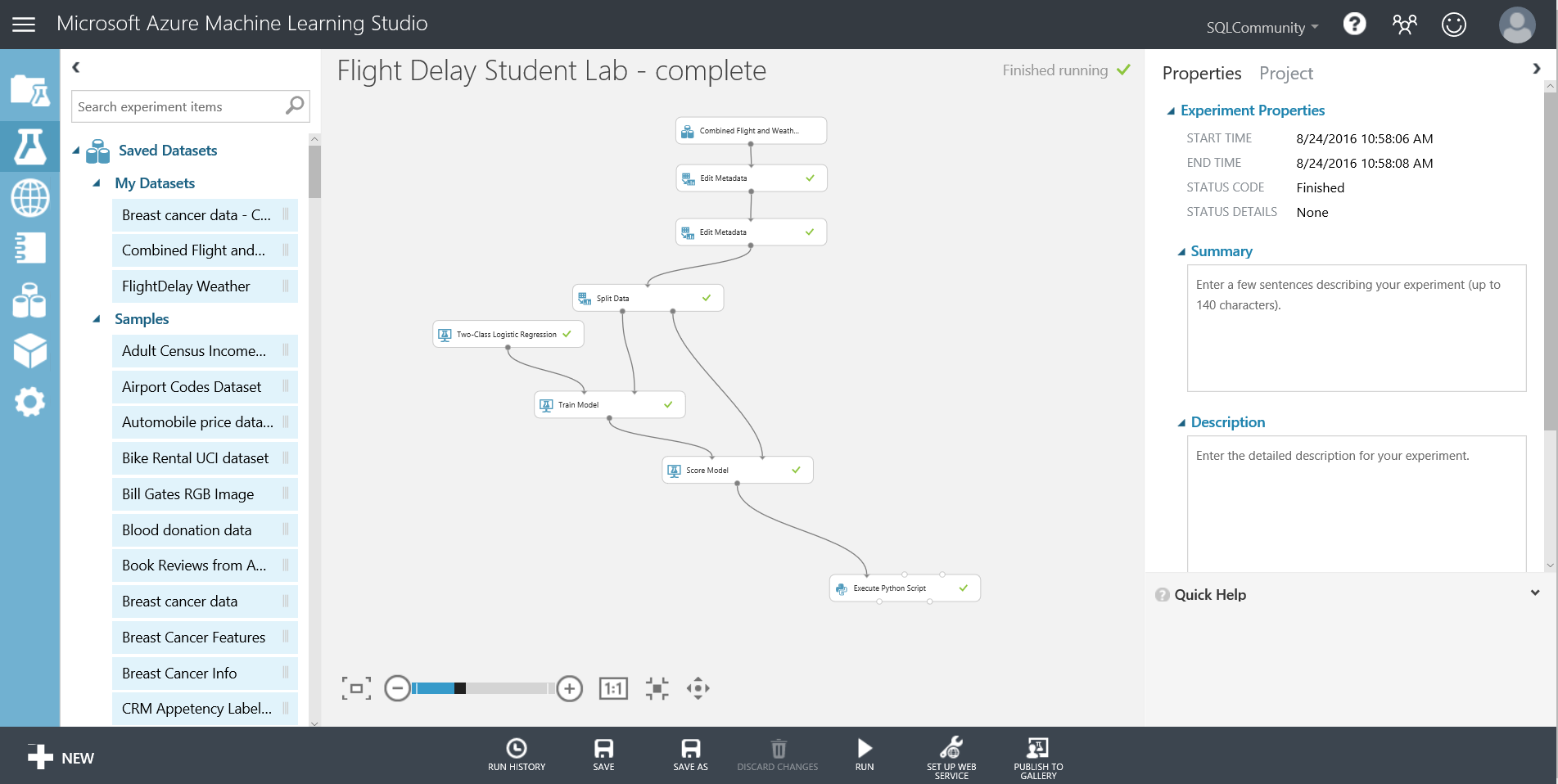
* What does the RoC curve look like if we remove the DepDelay feature from our experiment?
* Can we predict Dep Delay and then feed that answer into this experiment?

There is no right answer to these so in machine learning we constantly iterate to success.

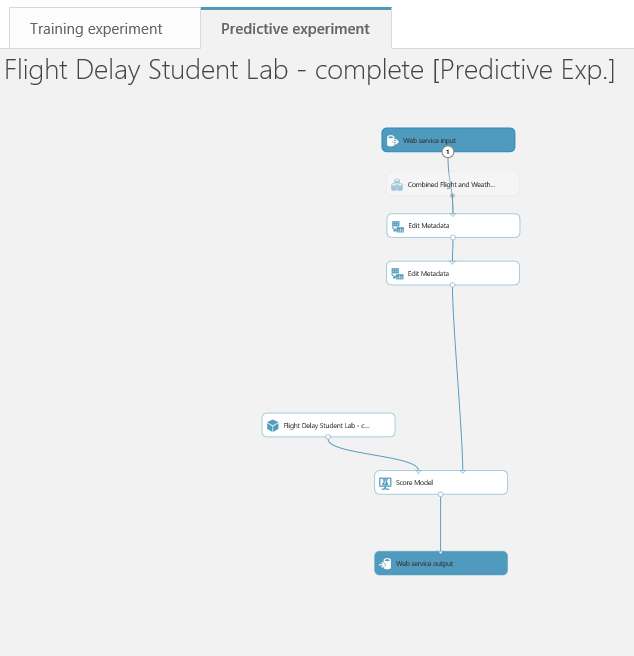
Publishing a Web Service

We have trained a model to classify flight data and checked to see how it is working. Now we want to use one of Azure ML’s key features – operationalising the classifier, by publishing an api to expose the model we have created. In order to do this, we first need to create a Training Experiment. This 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 - there are green ticks by each module you will see a button on the toolbar at the bottom of the screen become active:

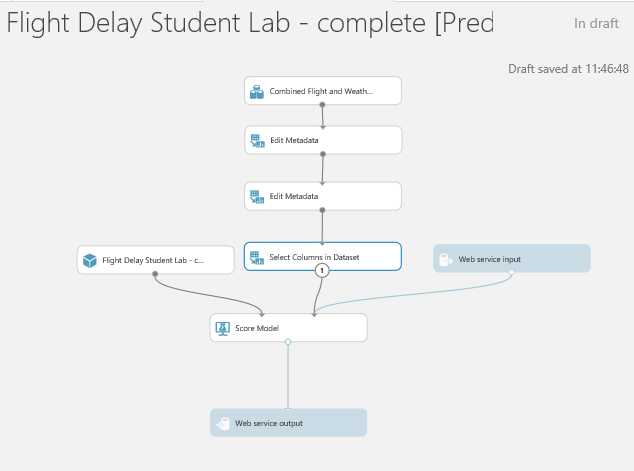


After clicking the ‘Predictive Web Service (Recommended)’ option our experiment appears to get redrawn and consolidated. What has really happened, is that ML has created a new Predictive experiment tab at the top of the screen.



and this is what we are looking at. Our original experiment is still there as it was on the Training experiment tab. Note there are two new modules in blue, “Web service input” and “Web service output” which represent the data format that will flow into and out of the web service we are creating. The Web service input will use the same fields as the input to the module it is connected to in this case what flows into the top metadata module.

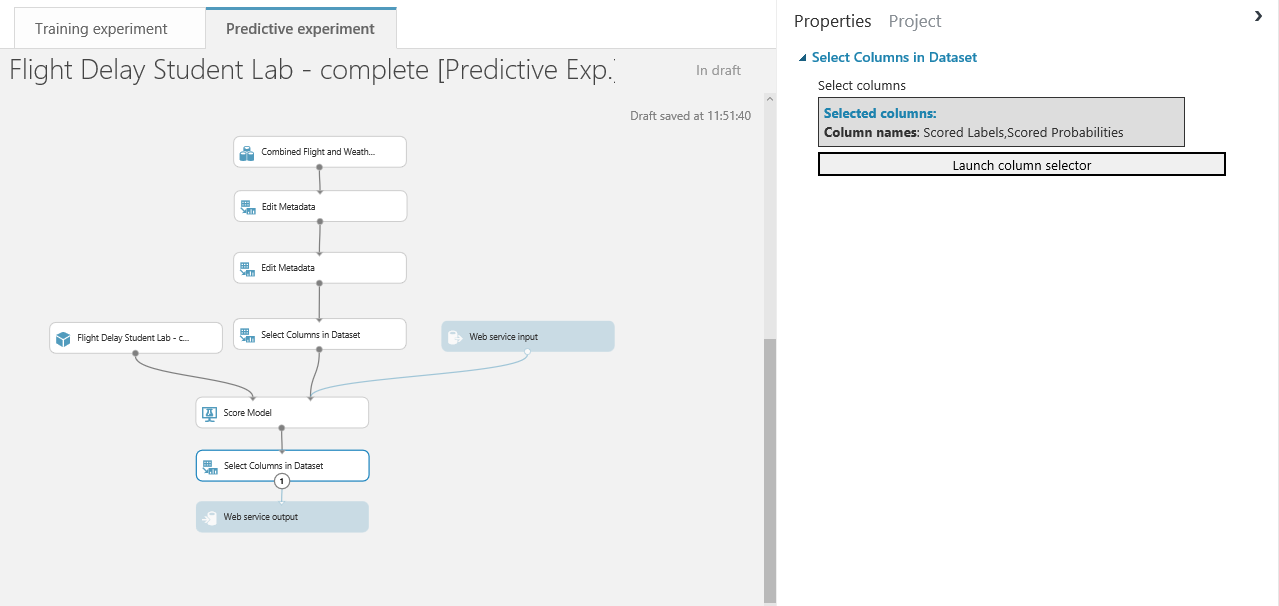
While the wizard has done a basic job of placing where the input and outputs on our predictive experiment it is not perfect. The data flowing through feed 1 in the above diagram contains our label (ArrDel15) which is what we are trying to predict. While this is valid for training we shouldn’t have it here so we can eliminate it by moving the Web service input to connect with the score model module and adding a ‘Select columns in a Dataset’ module as shown:



and set this module to exclude all labels:



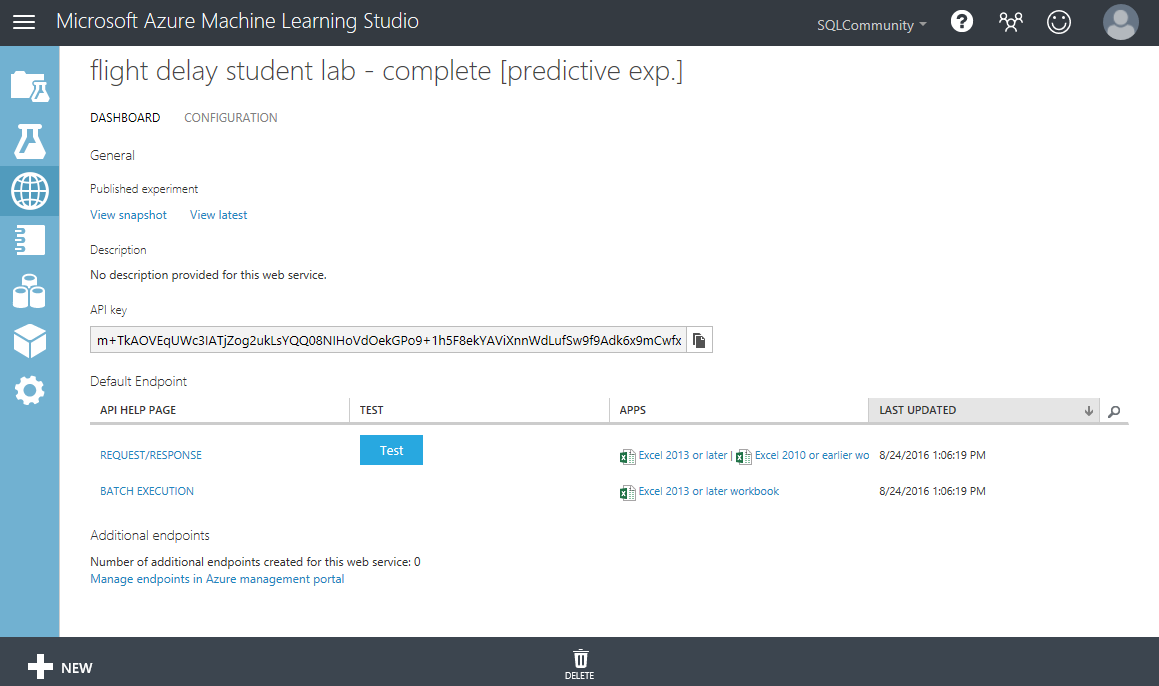
If we now think about what fields we want to return to the application or web site that will call our web service we are creating, then all we need are the scored labels (the prediction) and scored probabilities (the probability of the prediction being true). So we should add another ‘Select columns in a Dataset’ module as shown to just return those fields:



We must now run this predictive experiment as this allows ML to validate the predictive experiment before we can publish it as web service. 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 (classic). After a few seconds we’ll see we are taken to the web service section of ML studio and our new web service is displayed:



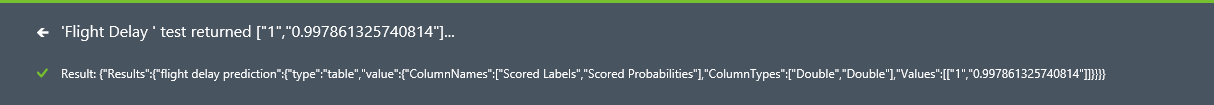
We can see some general information about the WEB API like the API key which is will be used for authorization purposes. There are also hyperlinks to help pages for the two endpoints of the service, a Request/Response endpoint and a Batch Execution endpoint. Notice we are also given some Excel spreadsheets where we can test the service as well.

On the Dashboard tab click on the button alongside the REQUEST/RESPONSE API as a quick sense check to see if our web service is working as expected.

Enter the following values in the Enter data to predict dialog:

* **DepDelay =** 33
* **DepTime =** 8
* **Month =** 4
* **Altimeter =** 29.6
* **Carrier =** DL
* **SeaLevelPressure** 30
* **DewPointCelcius =** 1.7

The result you receive back after processing returns a JSON like output.

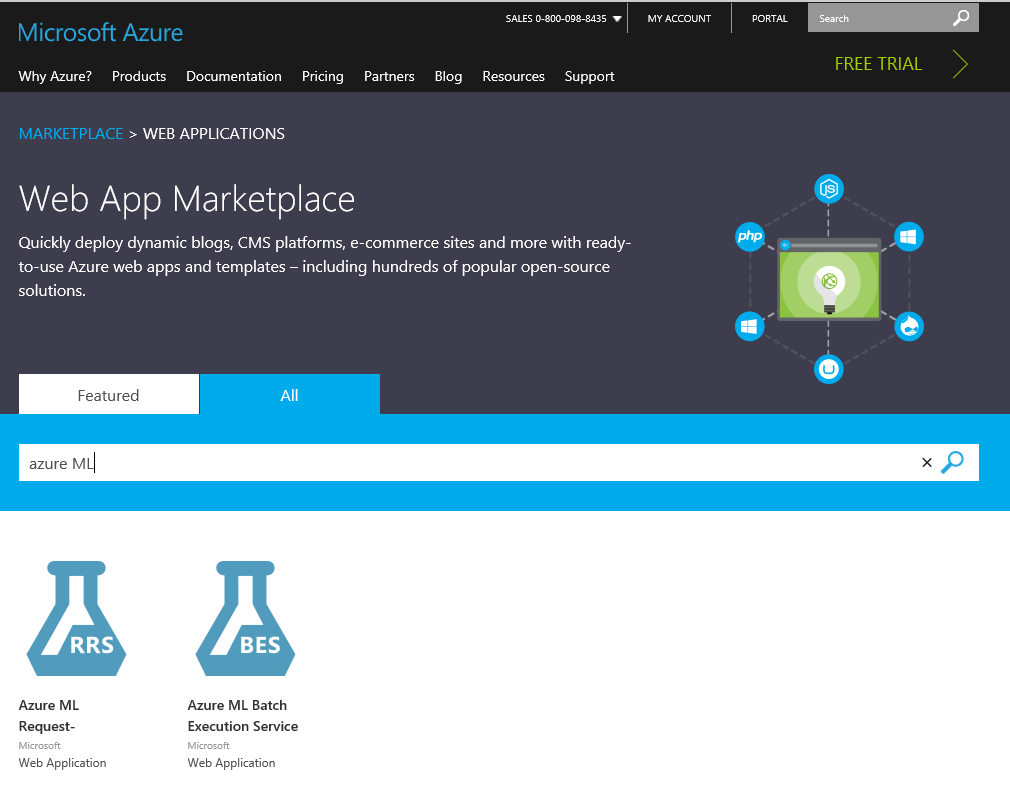


Notice we just get back the Scored Label and the Scored Probabilities. So in the case below we can say that given the parameters above the classifier has predicted the flight will be more than 15 mins delayed (1) and it is very confident in its decision (0.997861325740814).

Optional Exercise – create a Web Site using the api

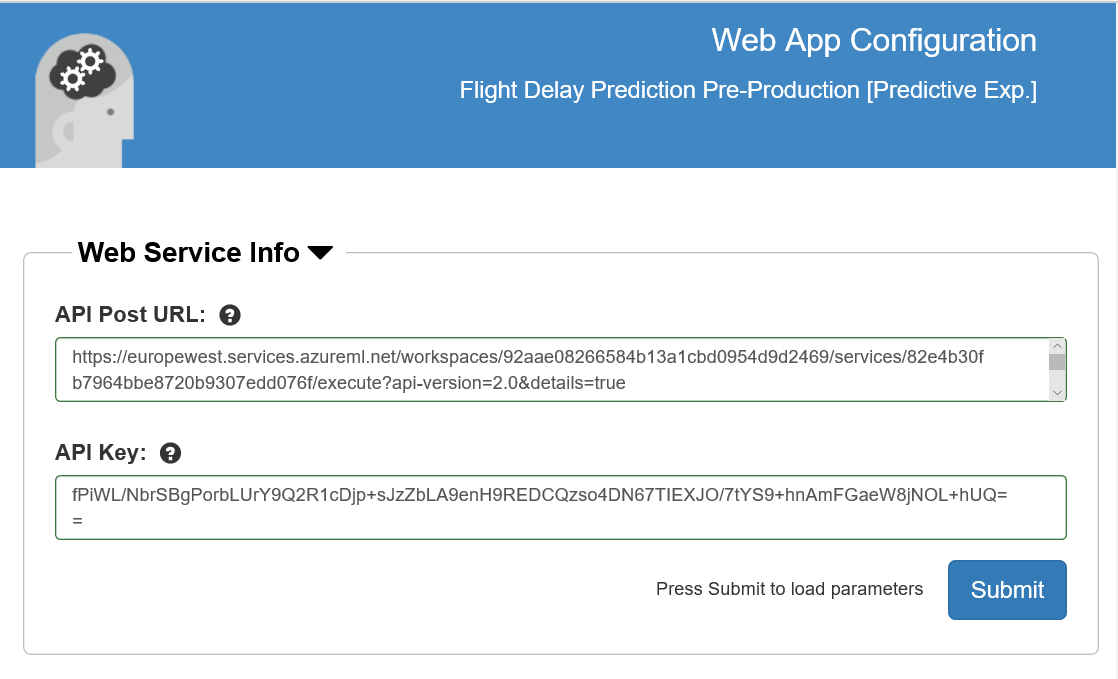
We can also test our new api using a partially configured web site in the [Azure Web App Marketplace](https://azure.microsoft.com/marketplace/web-applications/all/). Simply go to the site and look for Azure ML.

This will create a web app in our azure subscription.

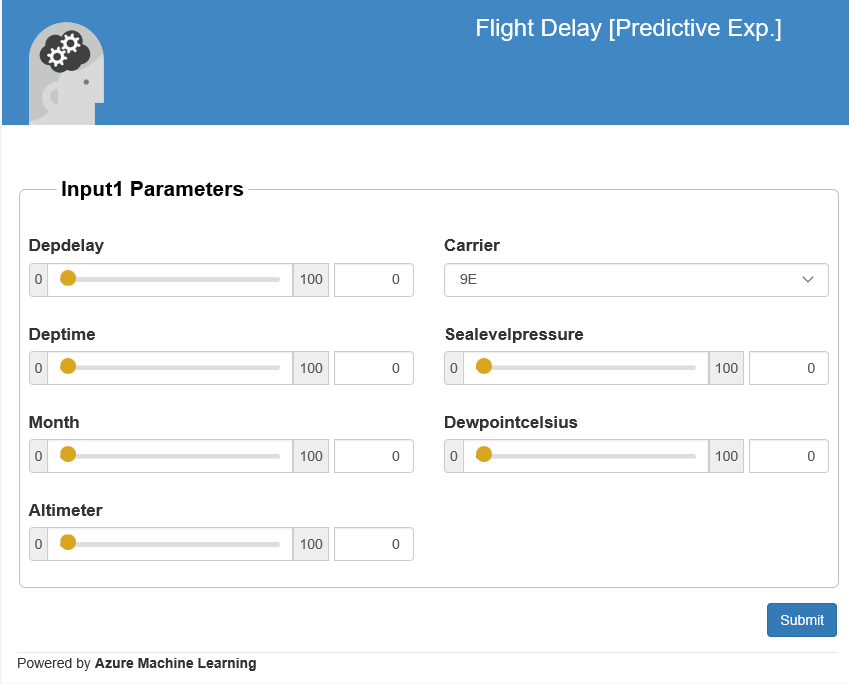




If we then click on the URL of the new site we’ll be presented with a simple page where we can enter our API URL and API key ..



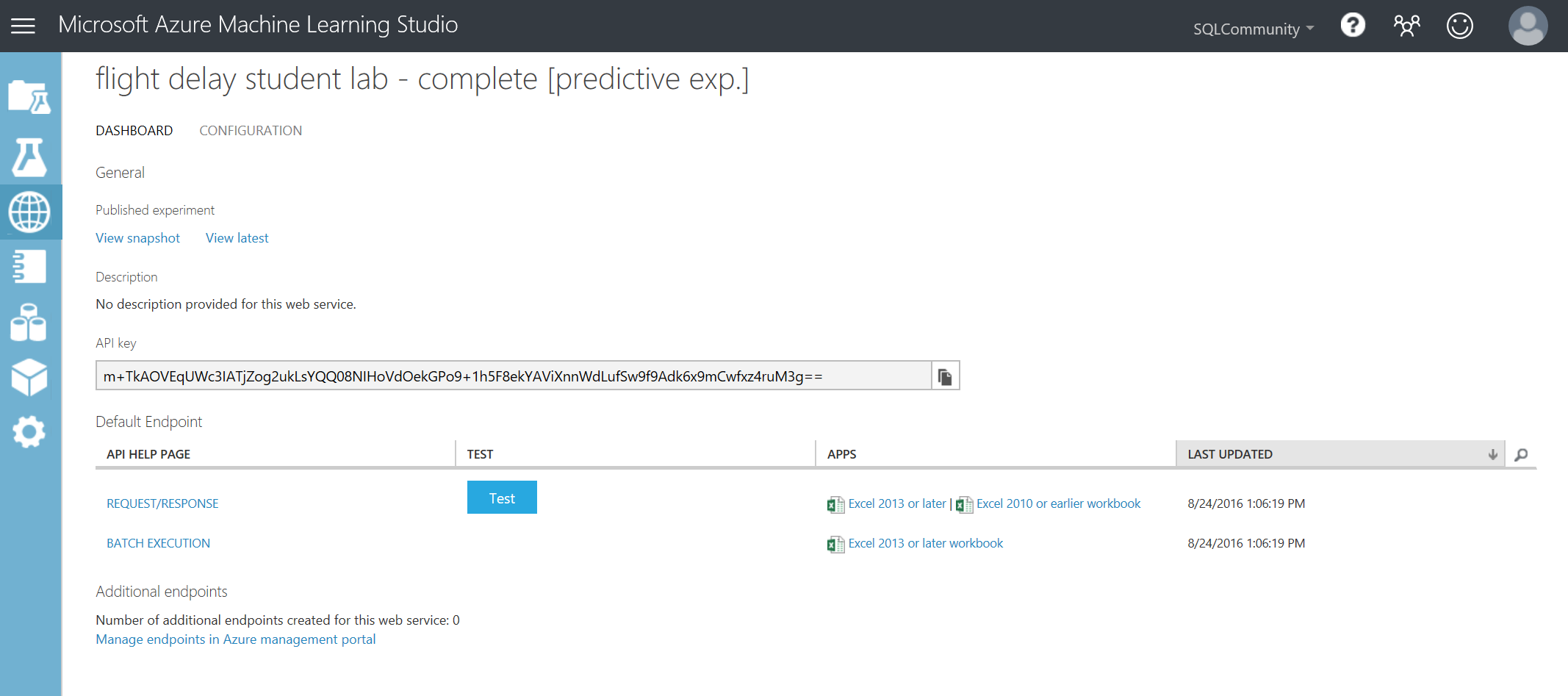
The key is on the API dashboard and the URL is at the top of the Request response page. Click Submit and close the page. Go back to the Azure Portal and open the site again and enter some trial values..



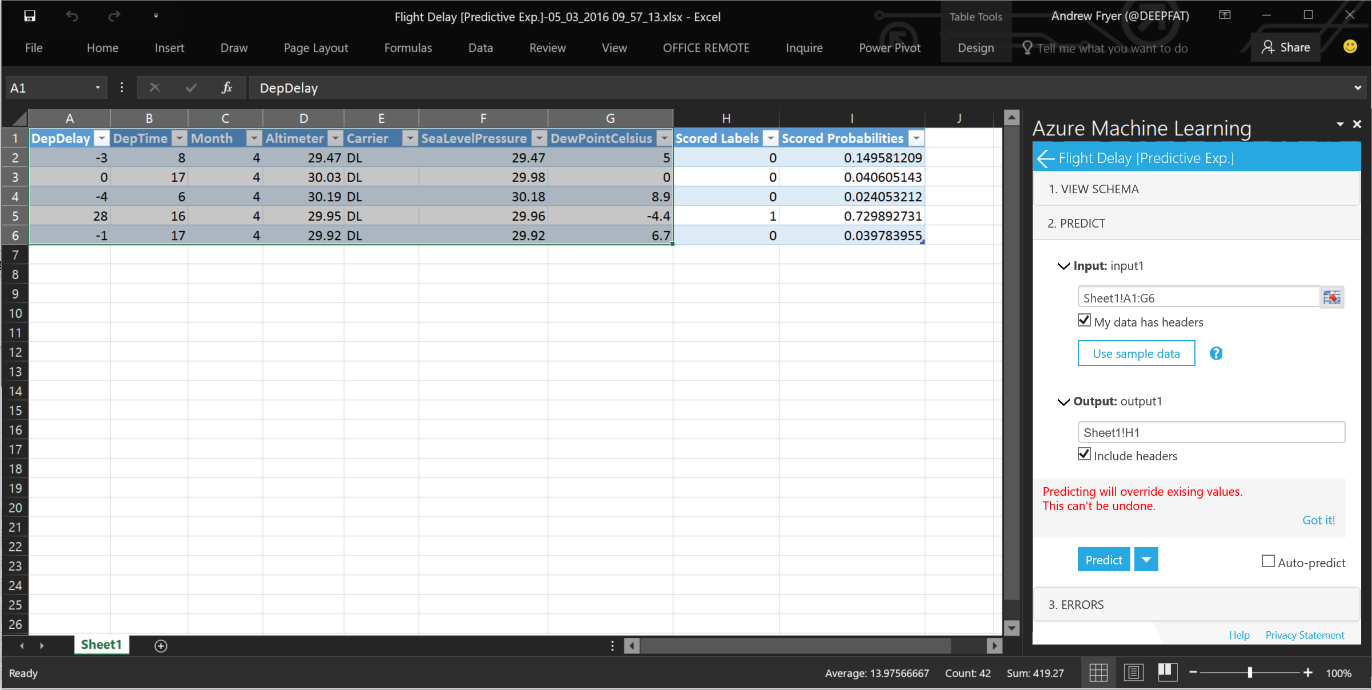
Optional Exercise – using Excel to call the api

We can also see how we can interact witht the new api form Excel, if you have Excel on your machine. Below is what the Excel 2013 version looks like which uses the new Excel add-ins to automatically setup a connection to the API we have and also allows us to use sample data to test.

From the web services page click on the web service and select the right Excel version for your laptop



Open the spreadsheet once it’s downloaded:



Click on sample data in the Azure Learning pane on the right. Select the sample data as Input rows and H1 as the Output and click Predict. You should see the new columns for Scored Labels and Scored Probabilities.

Conclusion

This lab was intended to introduce you to the basic concepts of Machine Learning such as binary classification, feature selection, training and testing a model and using Azure Machine Learning. A web service was created to operationalize and deploy the model for production.

**Next Steps:**

* Check out the UK Data Developer page on MSDN here:
* Check out the Azure ML Gallery and download and edit/run other types of machine learning experiments: <https://gallery.azureml.net/>
* Also check out other Azure Services that can be used with Machine Learning such as HDInsight, Data Factory and Stream Analytics: <http://azure.microsoft.com/en-us/services/>