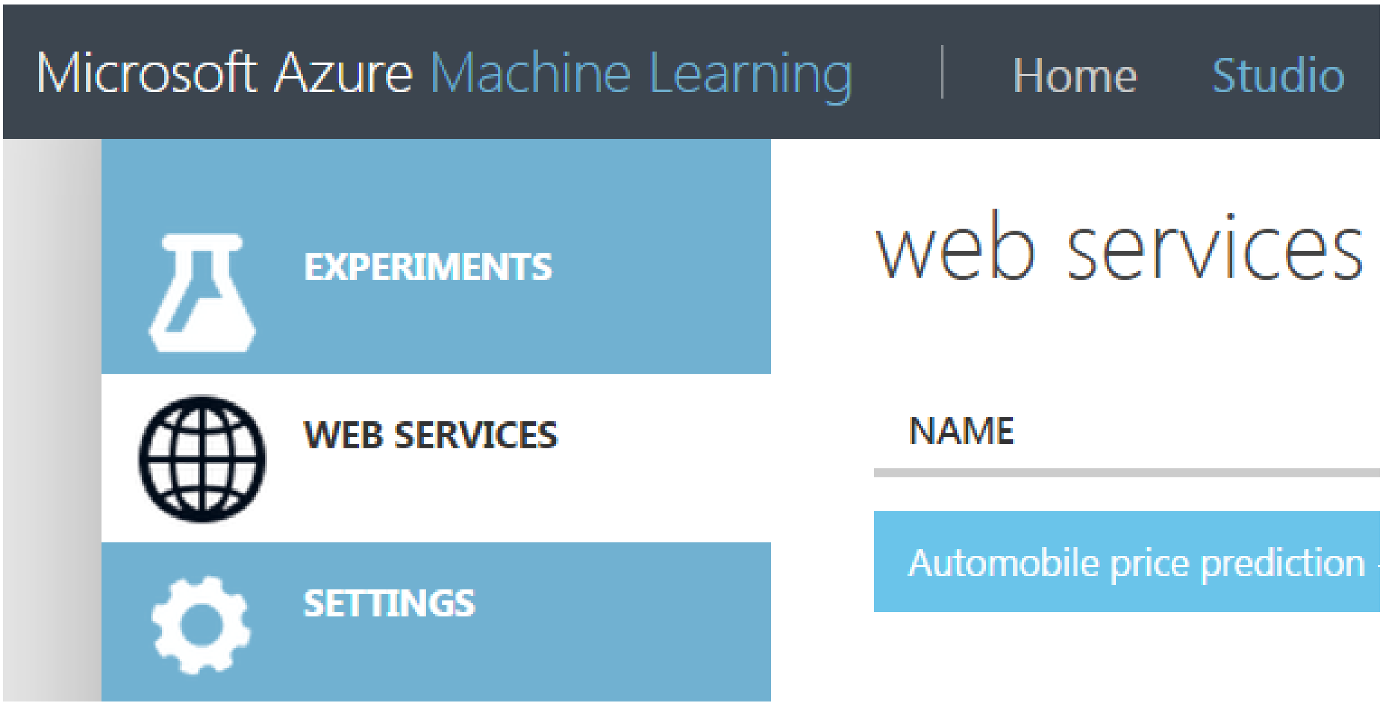
Deploy Serialised Python Models as Web Services using Azure Machine Learning Studio.



By Theo van Kraay, Data and AI Solution Architect at Microsoft

Azure Machine Learning Workbench is an integrated, end-to-end advanced analytics solution for professional data scientists. Data scientists can use it to prepare data, develop experiments, and deploy models at cloud scale. Go here for a full end-to-end tutorial on how to prepare (part 1) build (part 2), and deploy/operationalise your models as web services using Dockers (part 3) with Azure Machine Learning Workbench.

In this article, instead we explore an alternative method for deploying externally generated machine learning models as web services, using Microsoft's graphical tool for Data Science; Azure Machine Learning Studio. This product was originally designed to make Data Science more accessible for a wider group of potential users, by providing easy to use modules and a drag and drop experience for various Machine Learning related tasks.

The purpose of this article is to take you through how to deploy an externally trained and serialised sklearn Python machine learning model as a web service using the Studio features. Bear in mind that the Studio development environment is normally used for developing and training models using the drag and drop tools. However, it does also provide a convenient way of deploying robust machine learning web services in a serverless environment.

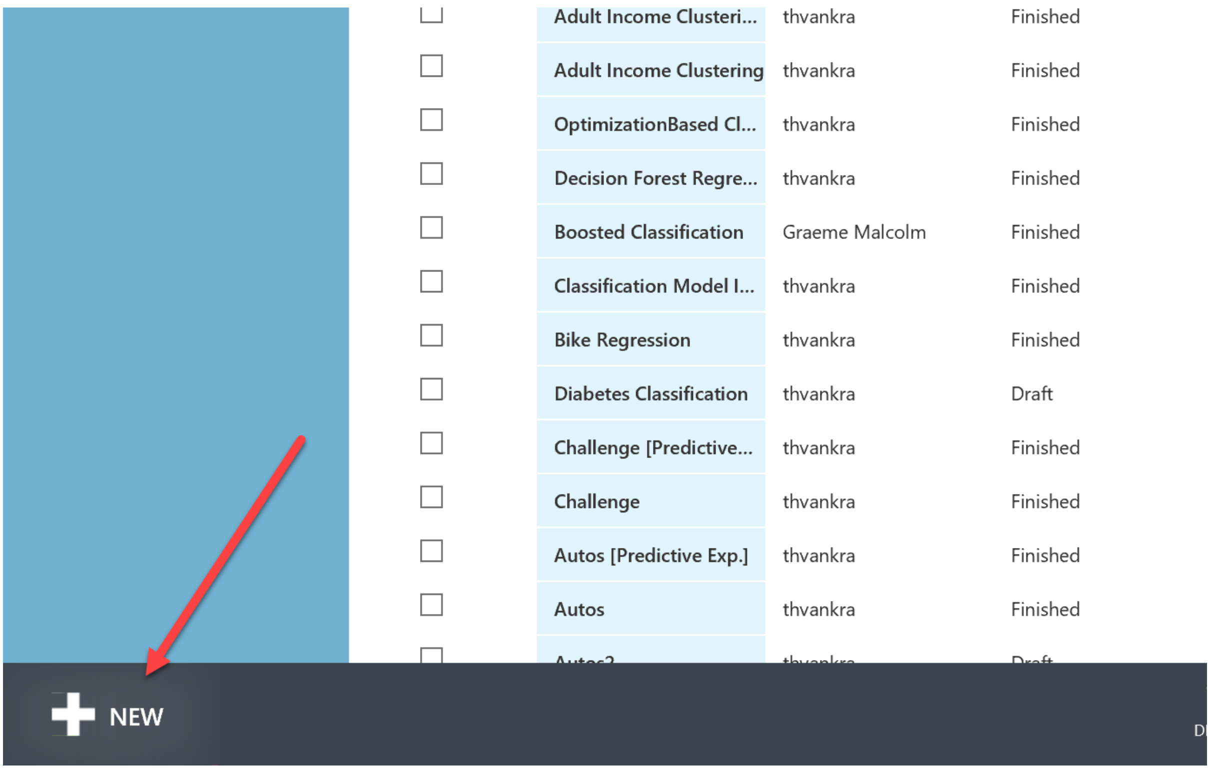
First, we generate a simple model in Python using the pickle module, training the model using a csv file that contains a sample from iris data set:

```
import pickle
import sys
import os
import pandas
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_curve

# create the outputs folder
os.makedirs('./outputs', exist_ok=True)

# load Iris dataset from a DataPrep package as a pandas DataFrame
iris = pandas.read_csv('iris.csv')
print ('Iris dataset shape: {}'.format(iris.shape))

# load features and labels
X, Y = iris[['Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width']].values,
iris['Species'].values

# split data 65%-35% into training set and test set
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.35, random_state=0)

# change regularization rate and you will likely get a different accuracy.
reg = 0.01

# load regularization rate from argument if present
if len(sys.argv) > 1:
    reg = float(sys.argv[1])

print("Regularization rate is {}".format(reg))

# train a logistic regression model on the training set
clf1 = LogisticRegression(C=1/reg).fit(X_train, Y_train)
print (clf1)

# evaluate the test set
accuracy = clf1.score(X_test, Y_test)
print ("Accuracy is {}".format(accuracy))

# serialize the model on disk in the special 'outputs' folder
print ("Export the model to model.pkl")
f = open('./outputs/model.pkl', 'wb')
pickle.dump(clf1, f)
f.close()
```

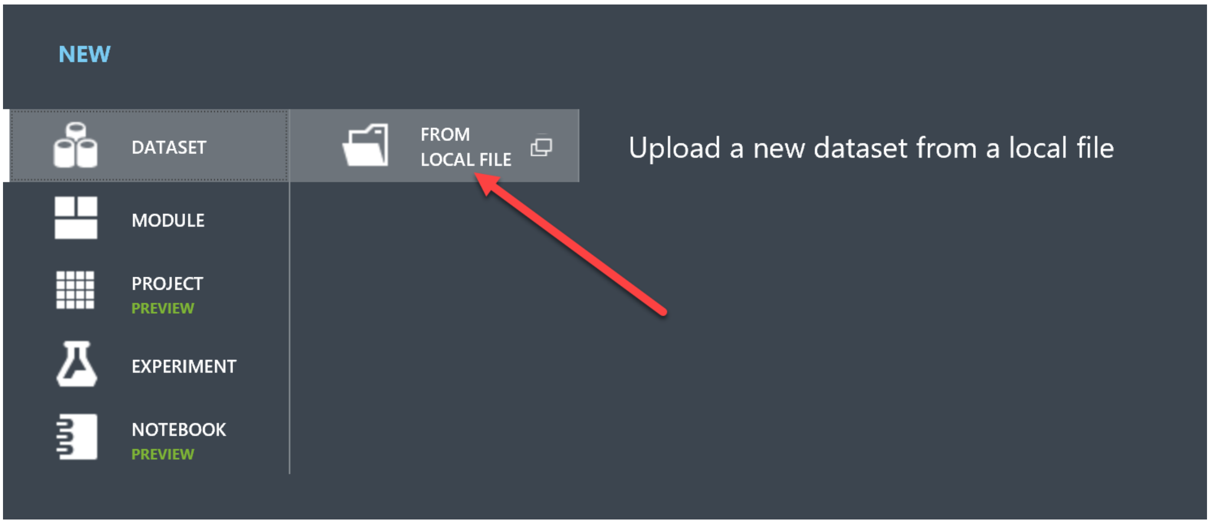
In the above example, the csv file just contains a sample of the iris data set. The “species” column is the classification that our Logistic Regression model is going to predict base on the 4 features of Sepal and Petal length and width:

	A	B	C	D	E
1	Sepal Lengt	Sepal Widtl	Petal Lengt	Petal Widtl	Species
2	5.1	3.5	1.4	0.2	Iris-setosa
3	4.9	3	1.4	0.2	Iris-setosa
4	4.7	3.2	1.3	0.2	Iris-setosa
5	4.6	3.1	1.5	0.2	Iris-setosa
6	5	3.6	1.4	0.2	Iris-setosa
7	5.4	3.9	1.7	0.4	Iris-setosa
8	4.6	3.4	1.4	0.3	Iris-setosa
9	5	3.4	1.5	0.2	Iris-setosa
10	4.4	2.9	1.4	0.2	Iris-setosa

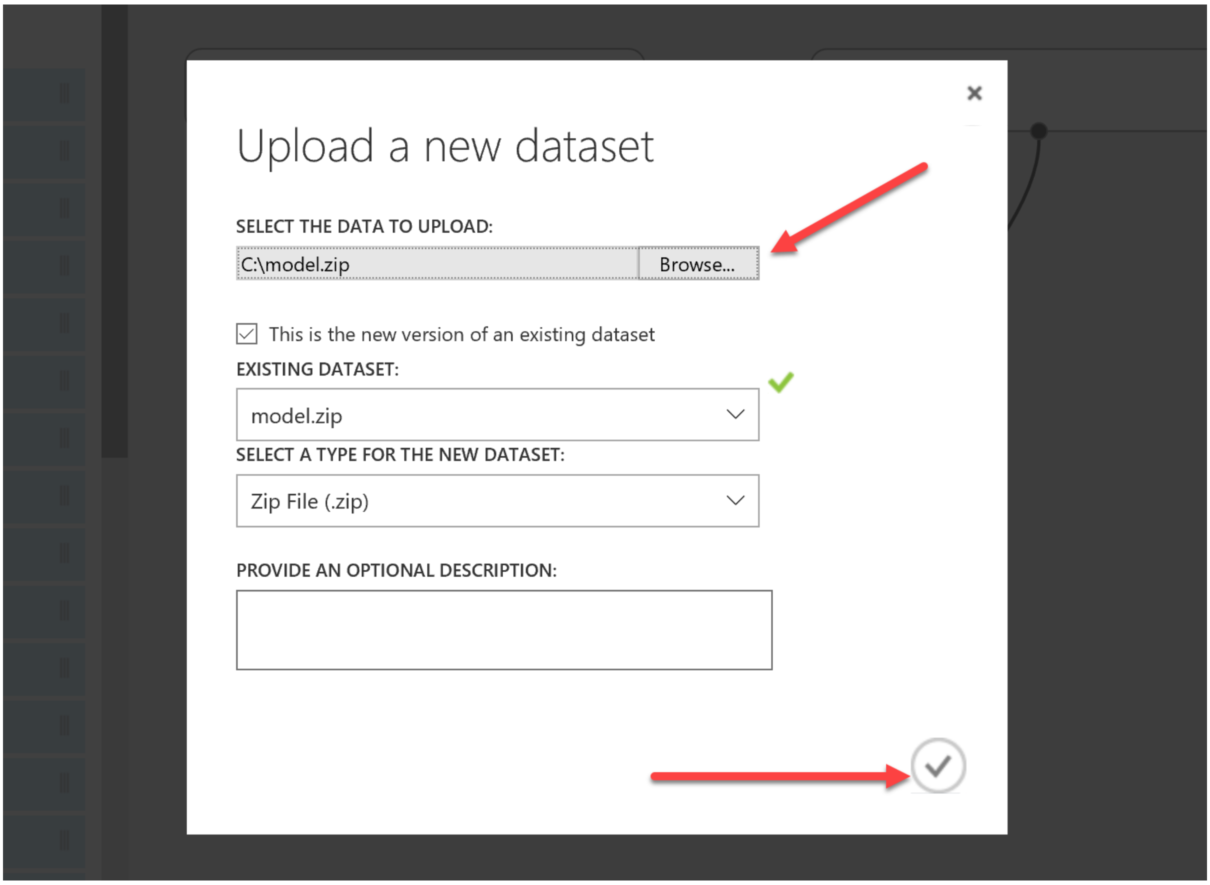
The above code will save the serialised model into the outputs folder. We take the model.pkl file, zip it, and upload it into the only Azure Machine Learning Studio (sign up [here]() if you have not already done so). Click the “New” icon in the bottom left:



In the pane that comes up, click on dataset, and then “From Local File”:



Select the zip file where you stored your serialised sklearn model and click the tick:

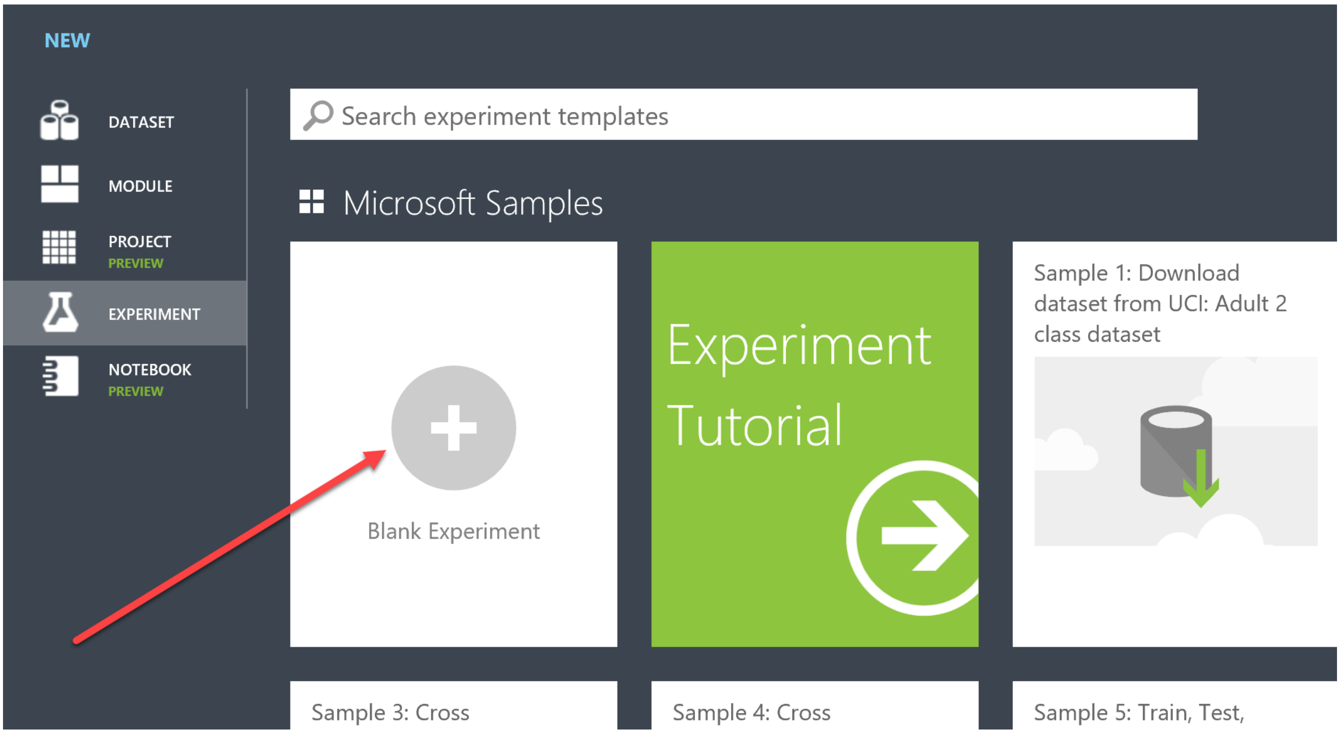


We are also going to create an **iris_input.csv** file that will be used to model the request input to the web service (note that this will not have the "species" column, as this is score label):

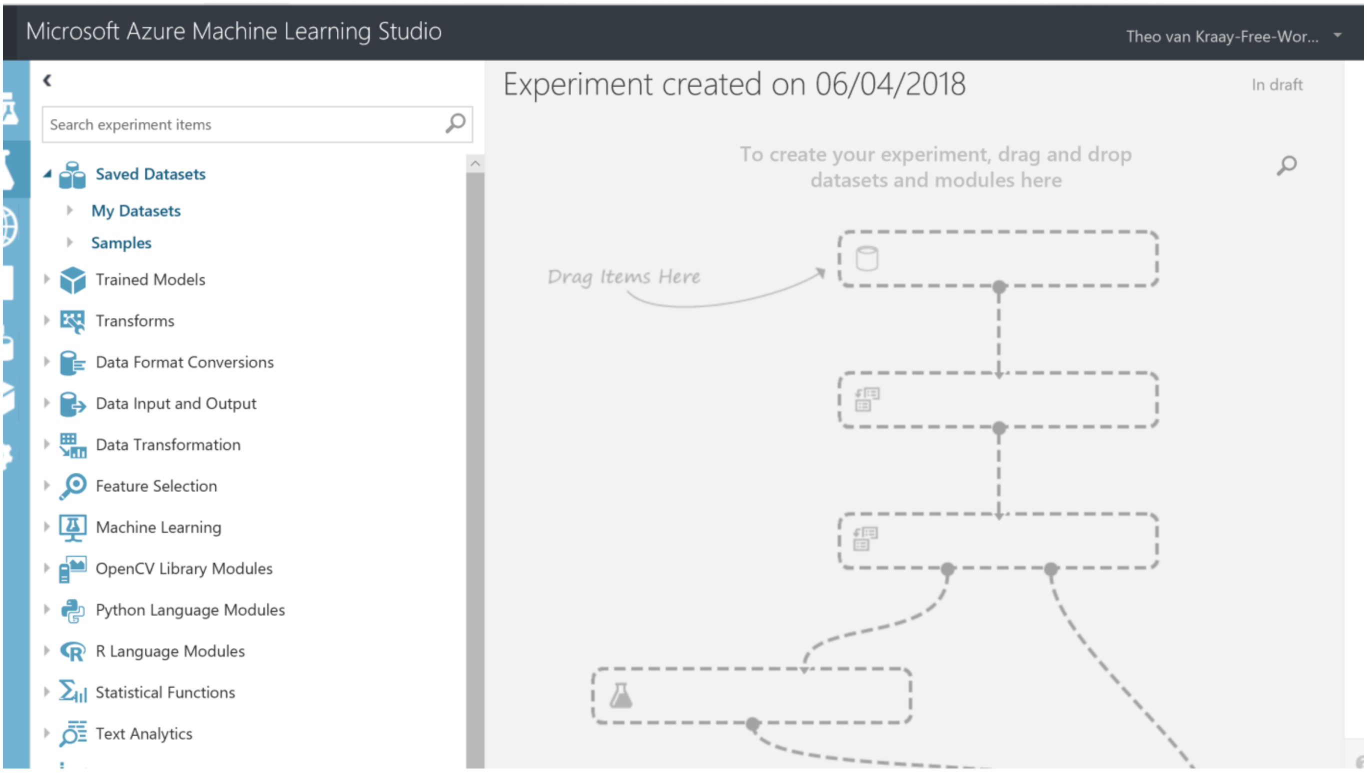
	A	B	C	D	E
1	Sepal Lengt	Sepal Widt	Petal Lengt	Petal Width	
2	5.1	3.5	1.4	0.2	
3					
4					

Use the same process as above to upload your iris_input.csv

Next, hit "new" and this time click "Blank Experiment":



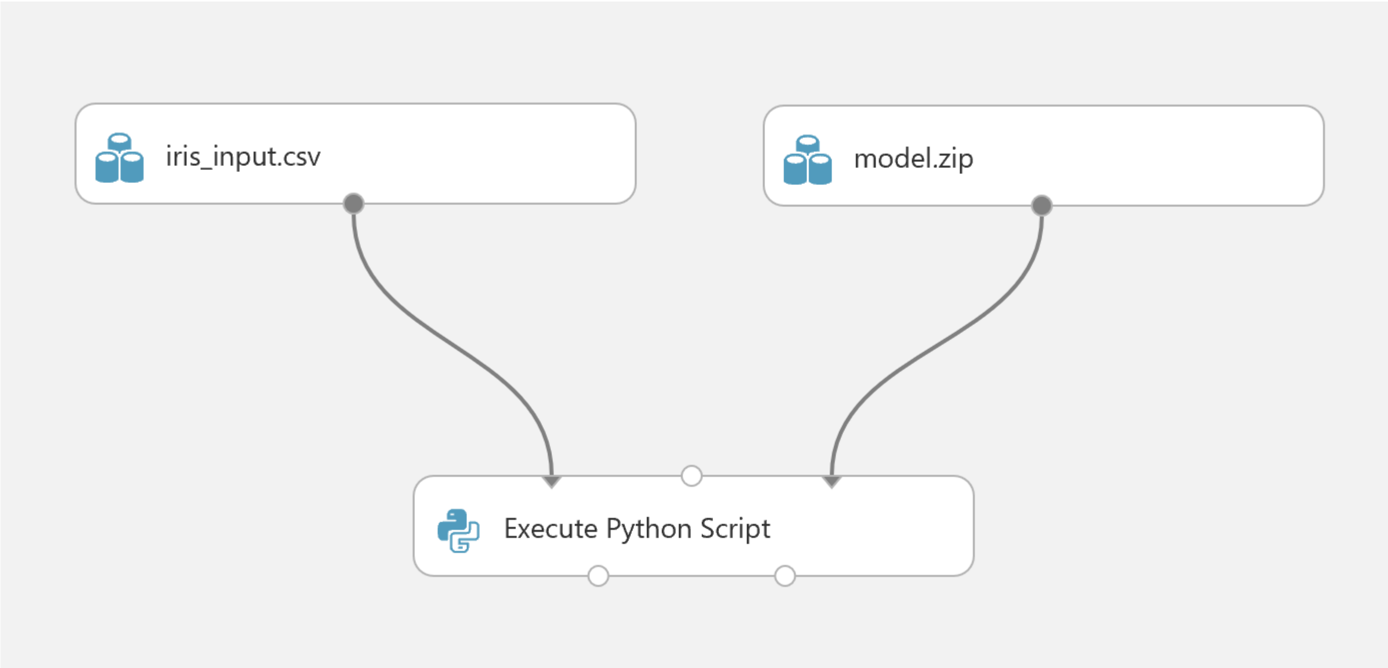
You will be presented with the experiment canvas:



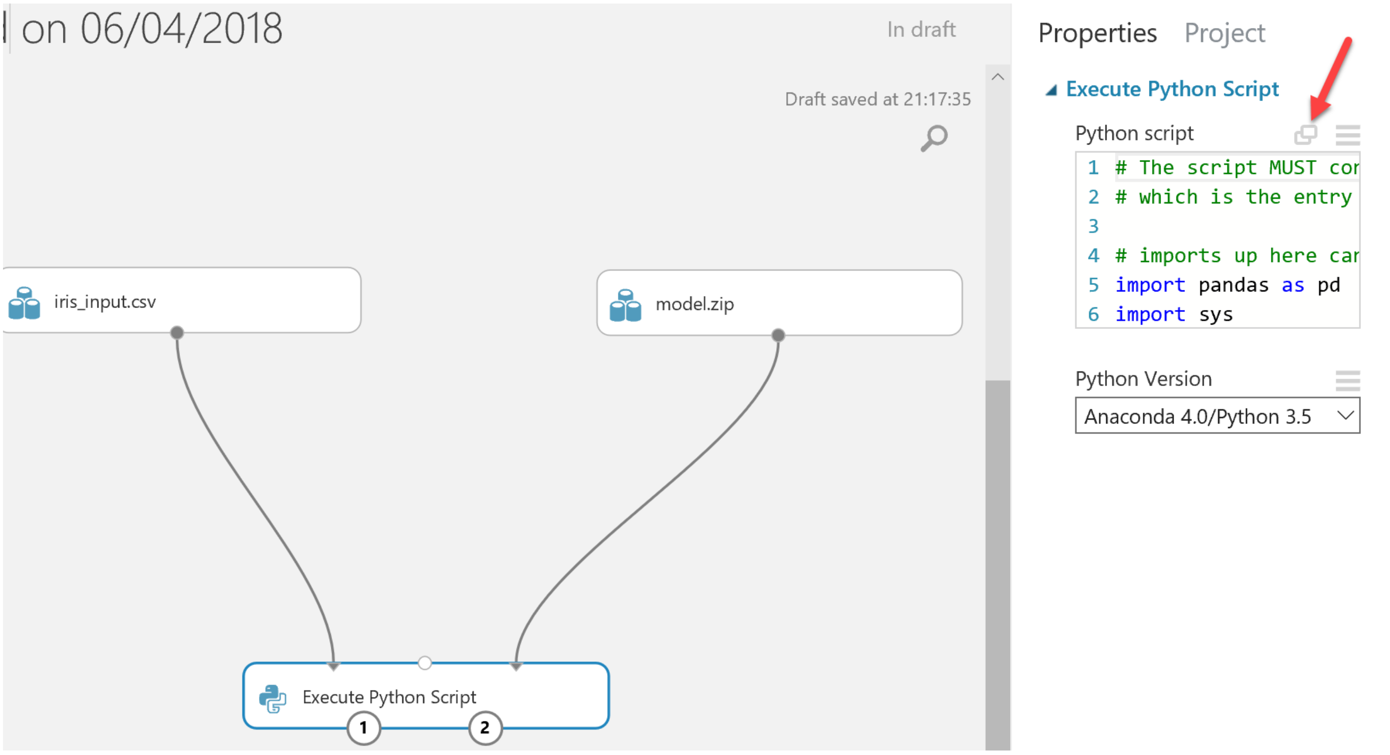
In the “search experiment items” box, search for each of the below, and drag each into the canvas:

- You serialised “model.zip” that you uploaded earlier
- Your “iris_input.csv”
- A module named “Execute Python Script”

When they are on the canvas, connect iris_input.csv and model.zip to the “Execute Python Script” module as illustrated below:

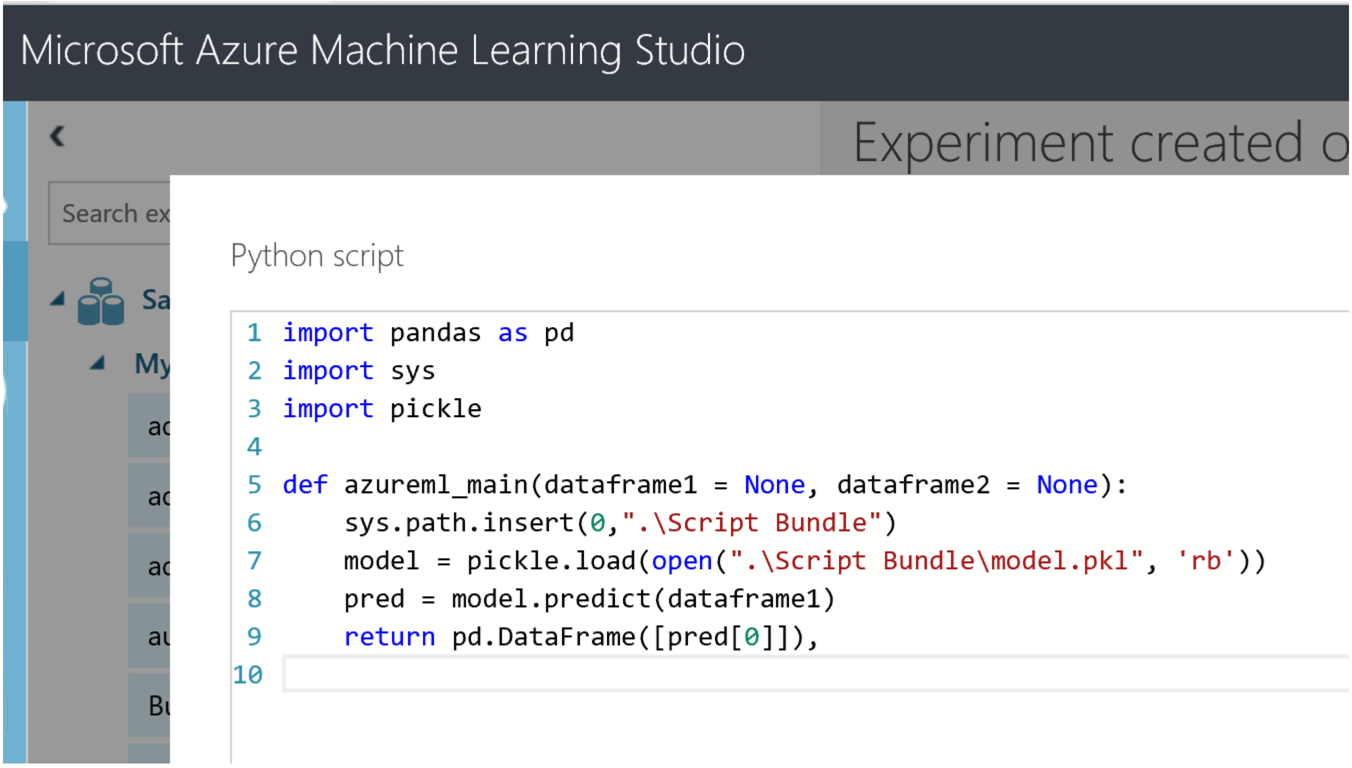


Highlight the execute Python Script Module, and an Execute Python Script pane will appear, click the highlighted icon below to expand it so you can edit the code (**note:** you will need to ensure that the Python version selected contains a version of the pickle module that matches the one used to originally create the serialised model) :

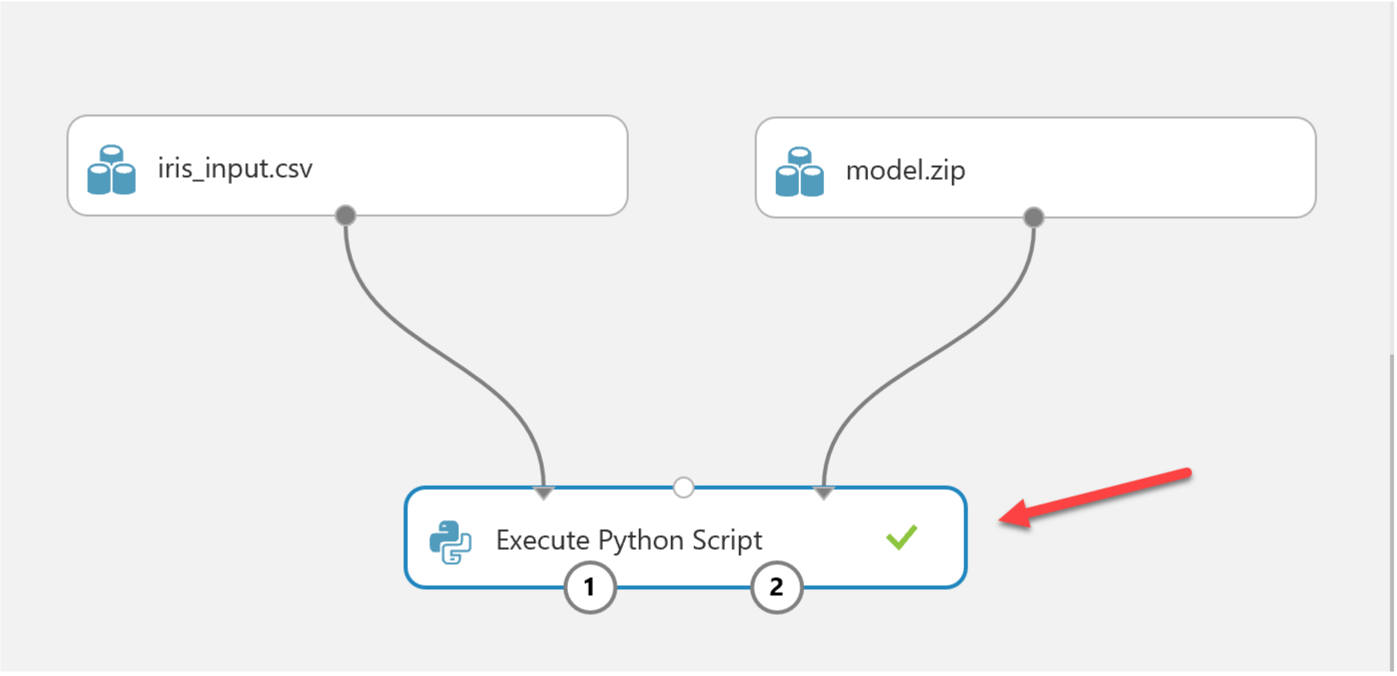


Replace the auto-generated code with the simple script below:

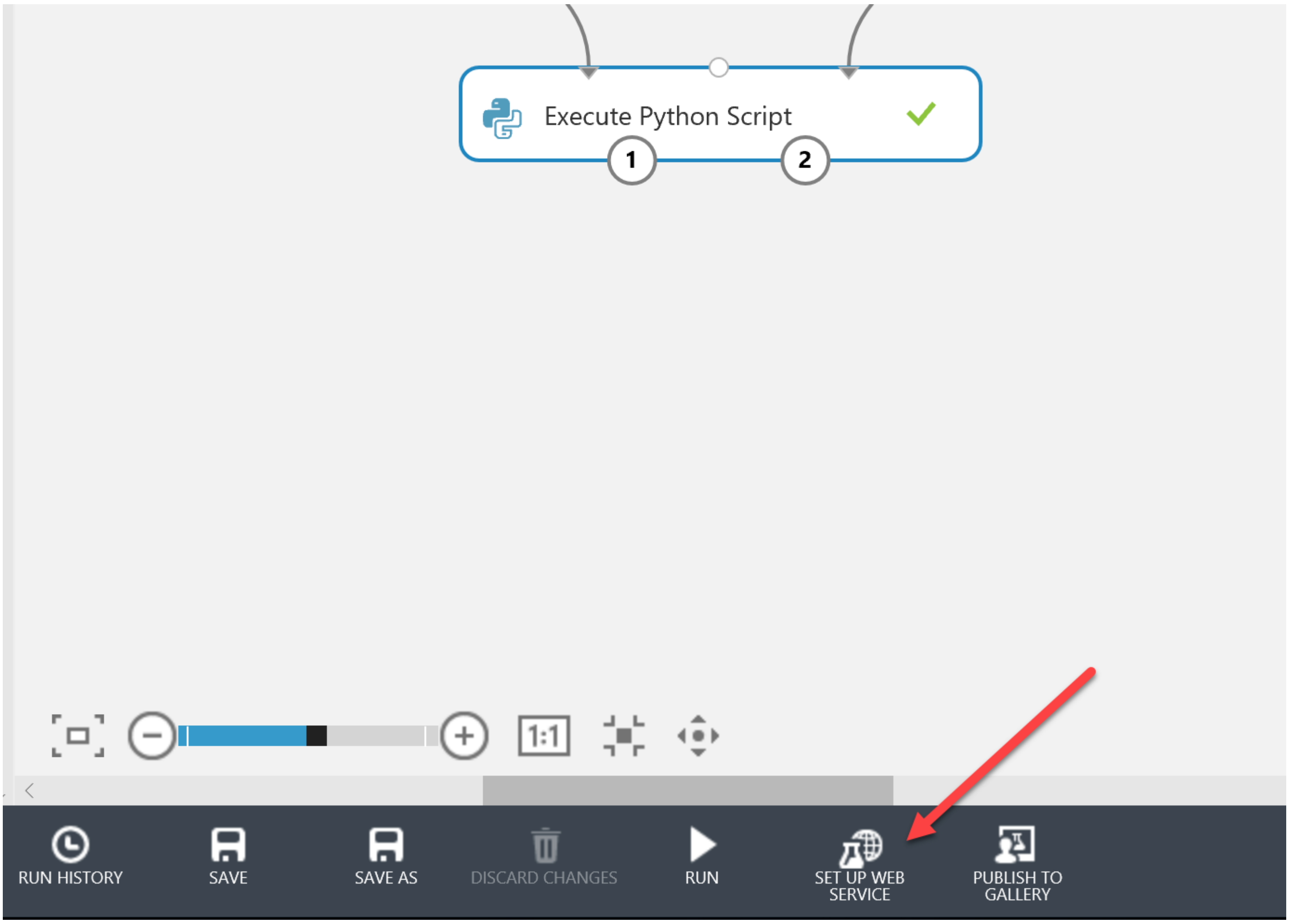
```
import pandas as pd
import sys
import pickle

def azureml_main(dataframe1 = None, dataframe2 = None):
    sys.path.insert(0,".\Script Bundle")
    model = pickle.load(open(".\Script Bundle\model.pkl", 'rb'))
    pred = model.predict(dataframe1)
    return pd.DataFrame([pred[0]]),
```



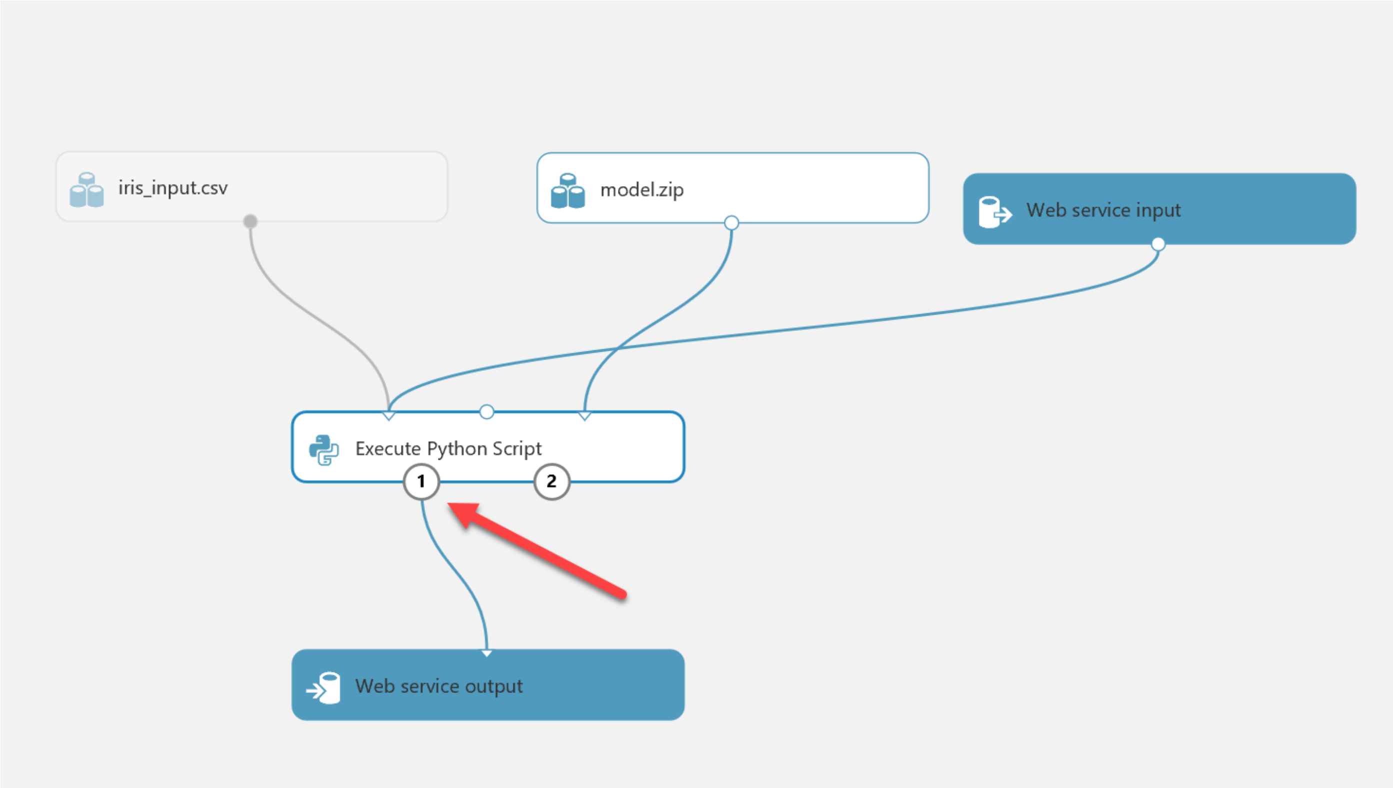
Click the tick, ensure you **save** the experiment using the icon in the bottom left, and then hit “Run” to run the experiment. This will de-serialise the model into the Azure Machine Learning Studio Environment.



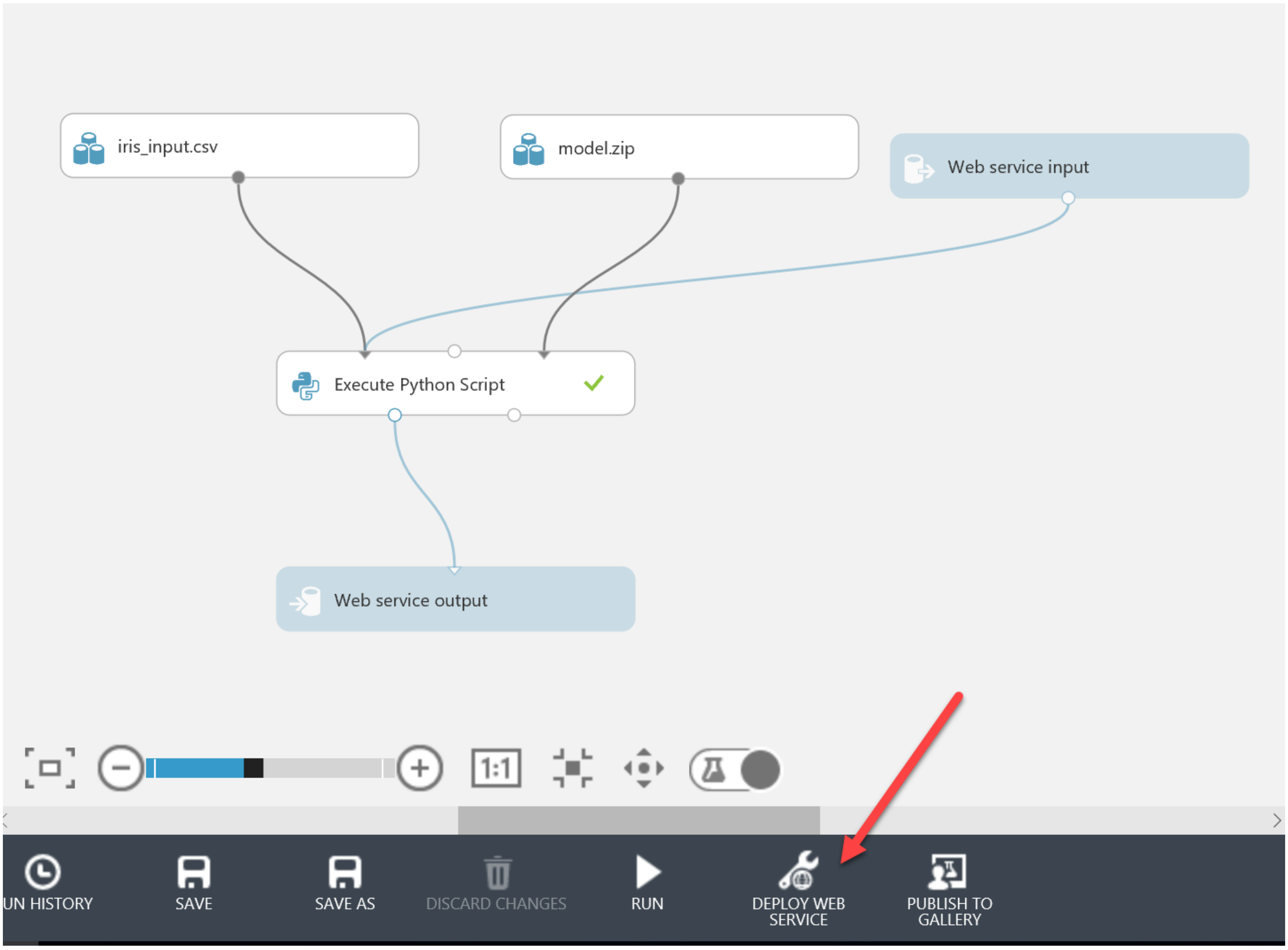
When finished the Execute Python Script module should have a green tick, You can now hit "Set up web service":



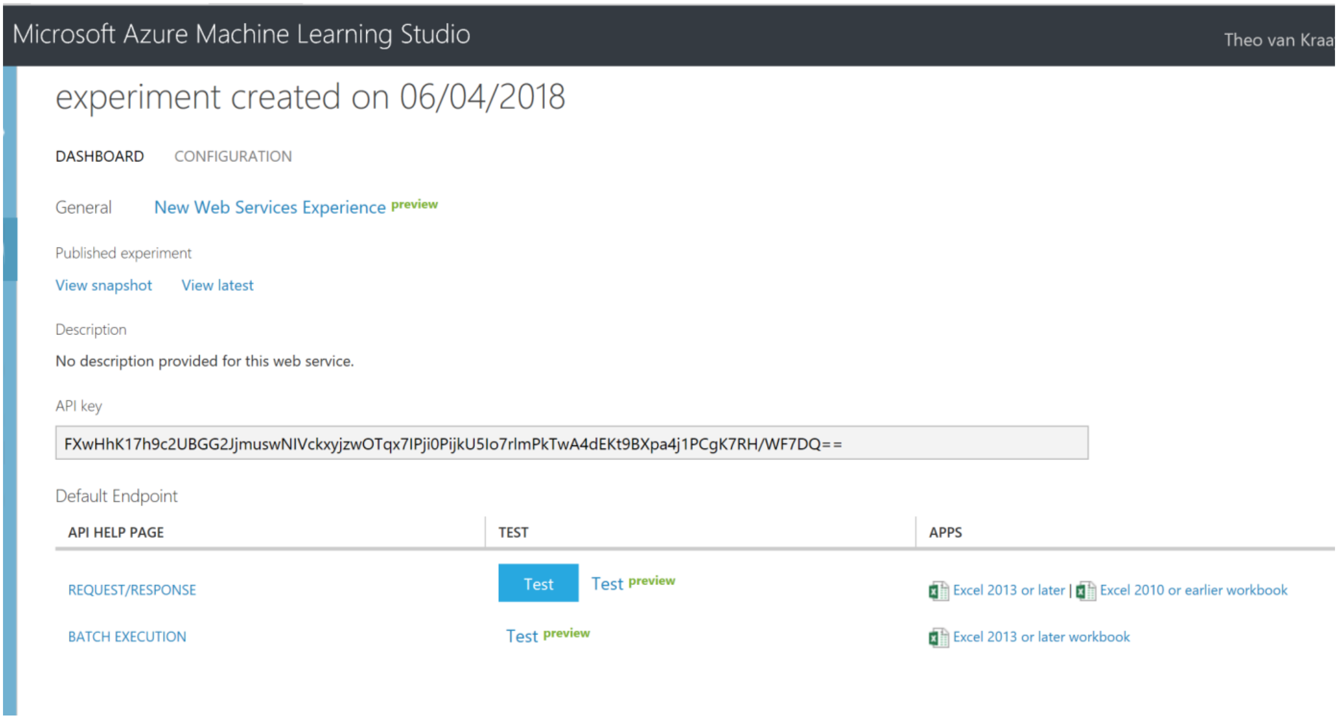
This will generate web service input and output modules. By default the output module will connect from the 2nd output port of the Python script. You will need to change this so that it connects from the 1st port, which is the result data set. Make sure your pane looks like the below:



Save the experiment. Before deploying the web service, you will need to run the experiment again (this is so Machine Learning Studio can determine the correct inputs and outputs from running the end-to-end model). When this is run and you have a green tick, you can hit “Deploy Web Service”:



This will take you to a screen with information about the newly provision web service, including the API key which you should store for later:



If you click on Request/Response, this will open a new window with comprehensive set of information about calling the web service, including Swagger documentation, and sample client API code: