**Azure ML Studio:**

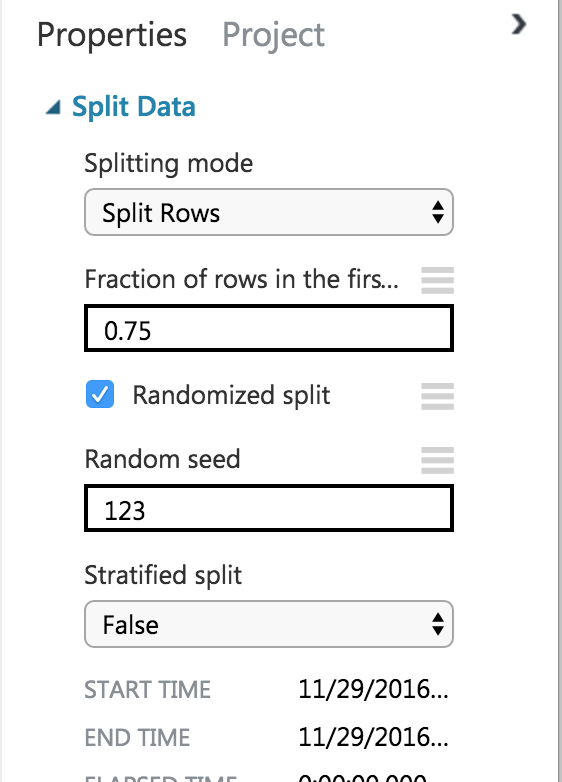
Read the consolidated csv file.

Select the desired column day, month, day hour, Temperature,Consumption and predicted/Classification values

Column for Day of the week has be assumed as in 0-6 format representing sun to Sat

**Azure Modules:**

* **Split Data -** To split a dataset into two equal parts, just addthe [Split Data](https://msdn.microsoft.com/en-us/library/azure/dn905969) module after the dataset without no other changes. By default, the module splits the dataset in two equal parts. For data with an odd number of rows, the second output gets the remainder.



**Train Model** - Training a classification or regression model is a kind of*supervised**machine learning*. That means you must provide a dataset that contains historical datafrom which to learn patterns. The data should contain both the outcome you are trying to predict, and related factors (variables). The machine learning model uses the data to extract statistical patterns and build a model.

When you configure Train Model**,** you must also connect an already configured model, such as a regression algorithm, decision tree model, or another machine learning module.

**Score Model -** Score Model is used to generate predictions using a trainedclassification or regression model. The predicted value can be in many different formats, depending on the model and your input data: If you are using a classification model to create the scores, Score Model outputs a predicted value for the class, as well as the probability of the predicted value. For regression models, Score Model generates just the predicted numeric value.

* + **Evaluate Model** - Evaluate Model is used to measure the accuracy of a trainedclassification model or regression model. You provide a dataset containing scores generated from a trained model, and the Evaluate Model module computes a set of industry-standard evaluation metrics. The metrics returned by Evaluate Model depend on the type of model that you are evaluating

**Classification Models**

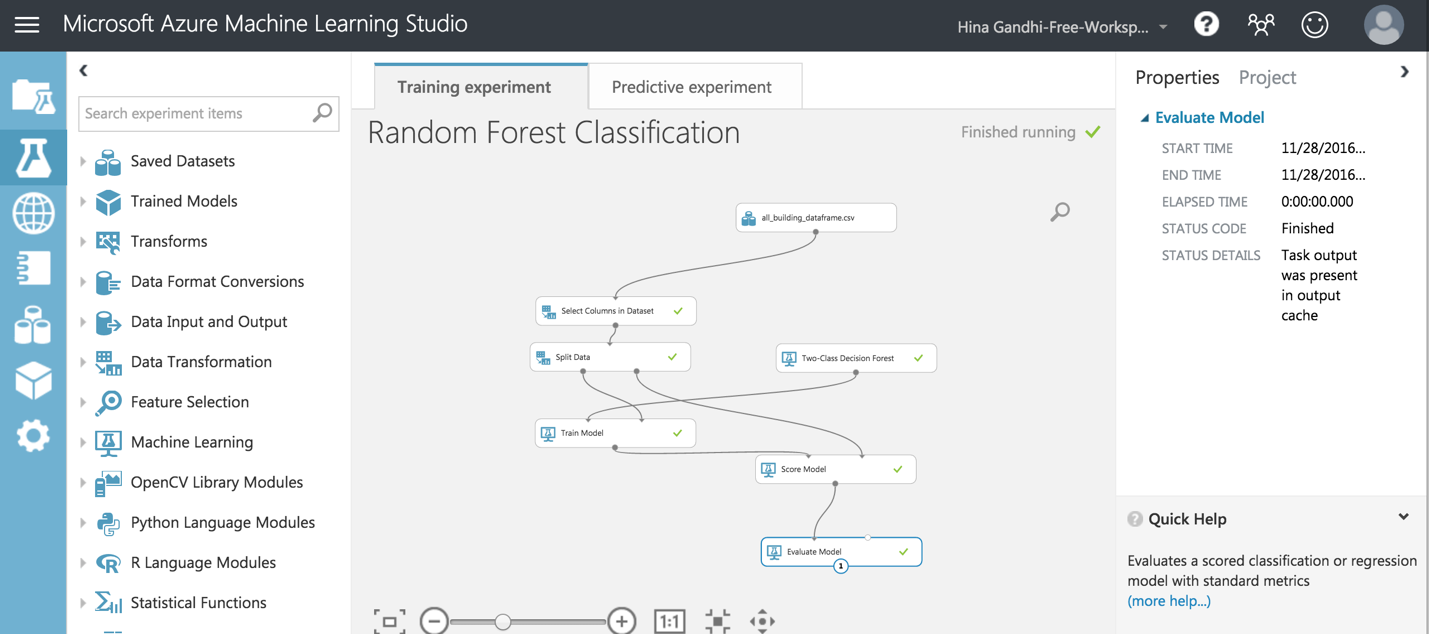
We build classification model to classify the given consumption into high and low.

**Two class Decision Forest**

The decision forest algorithm is an ensemble learning method for classification. The algorithm works by building multiple decision trees and then voting on the most popular output class. Voting is a form of aggregation, in which each tree in a classification decision forest outputs a non-normalized frequency histogram of labels. The aggregation process sums these histograms and normalizes the result to get the

“probabilities” for each label. The trees that have high prediction confidence will have a greater weight in the final decision of the ensemble.

**Azure Modules:**



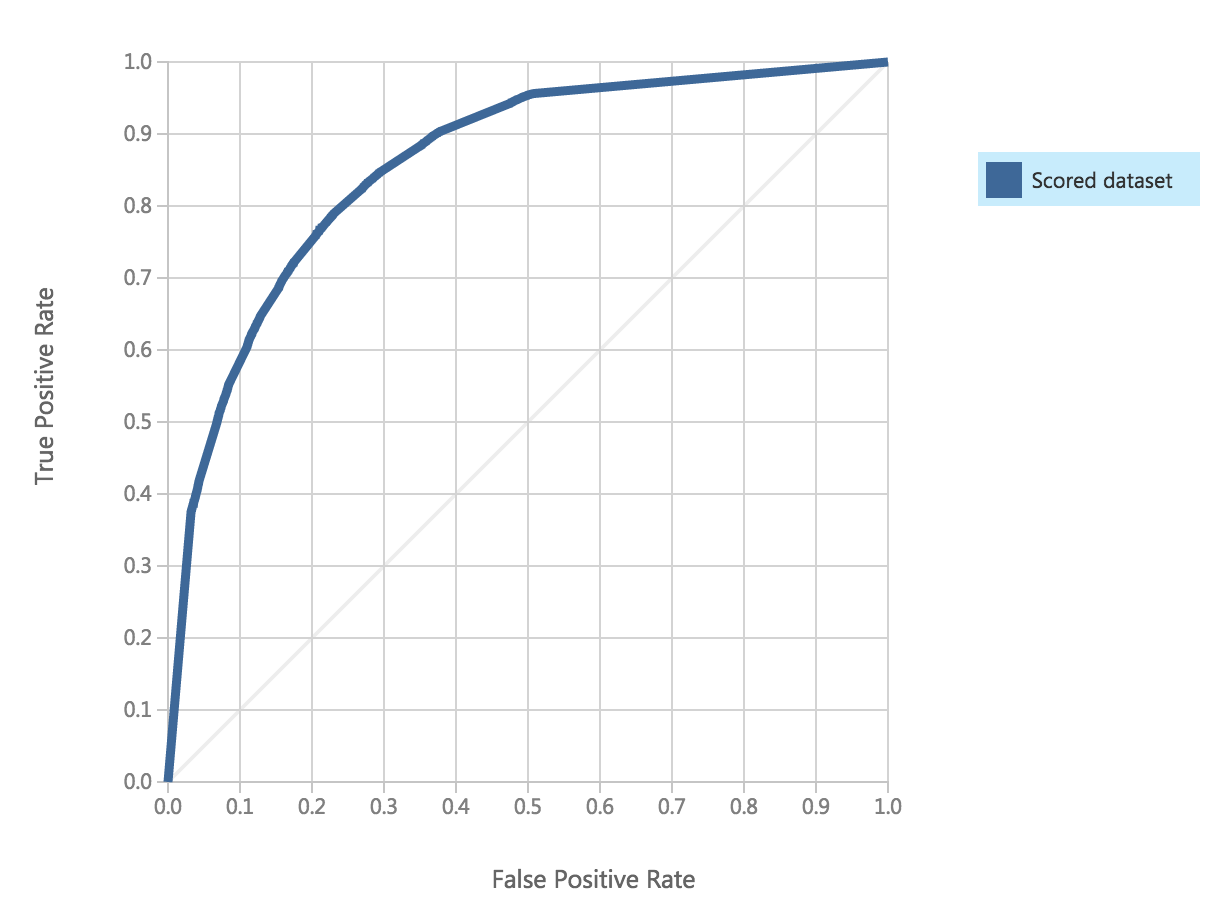
**ROC curve**

The ROC curve plots the pairs {sensitivity, 1-specificity} as the cutoff value increases from 0 and 1

* **Sensitivity** (also called the **true positive rate**, or the **recall** in some fields) measures theproportion of positives that are correctly identified (e.g., the percentage of sick people who are correctly identified as having the condition).
* **Specificity** (also called the **true negative rate**) measures the proportion of negatives thatare correctly identified

as such (e.g., the percentage of healthy people who are correctly identified as not having the condition).

• Better performance is reflected by curves that are closer to the top left corner

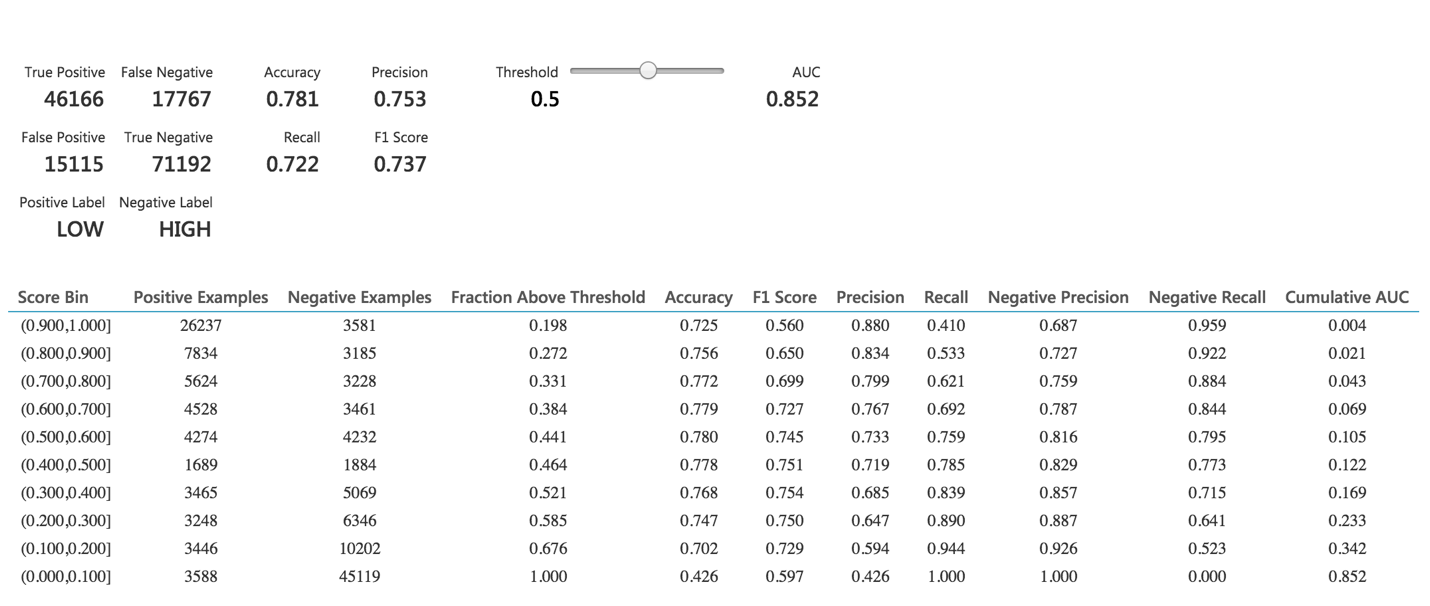


**Confusion Matrix and other performance metrics :**

The following statistics are shown for our model:

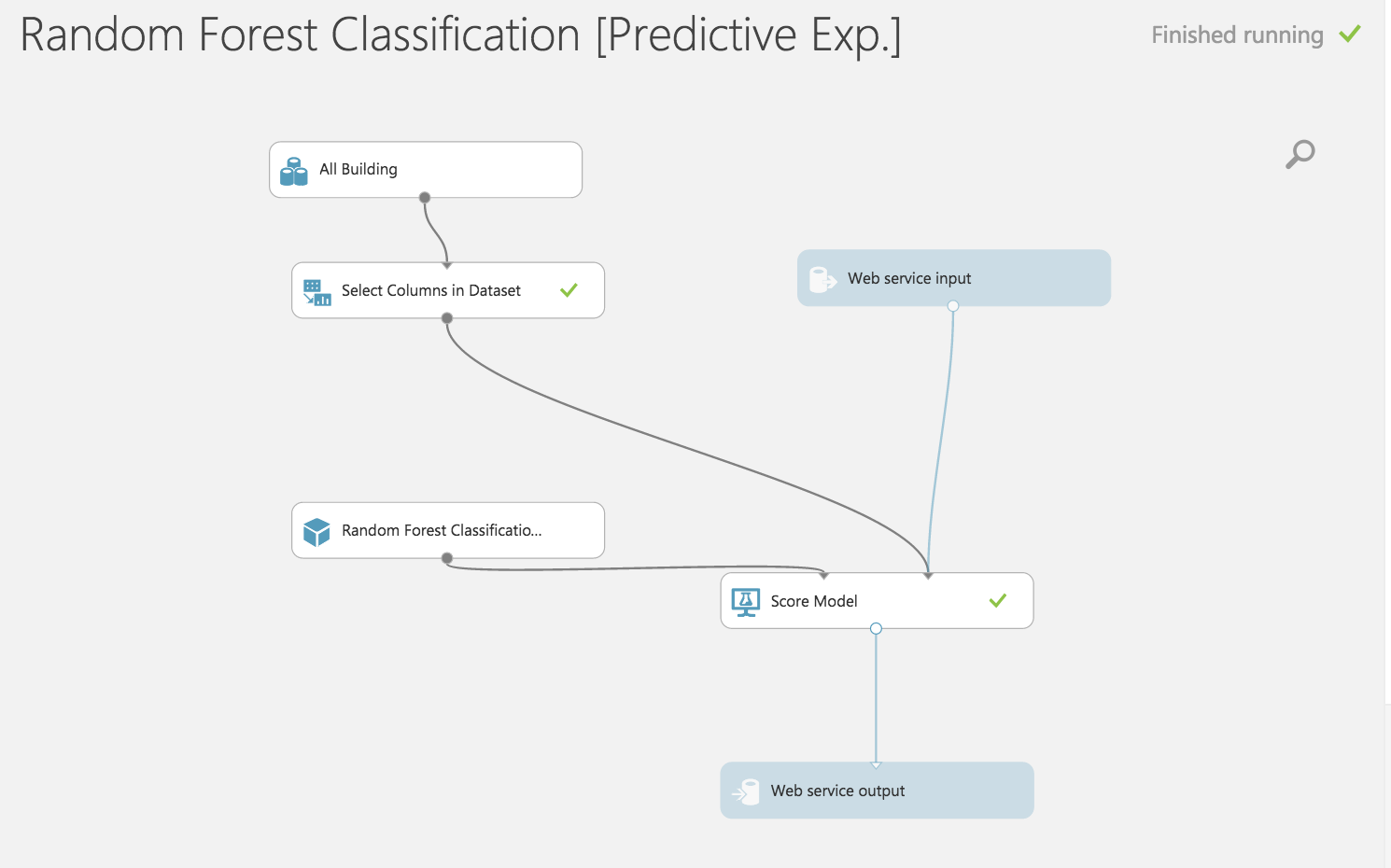
* **Mean Absolute Error** (MAE): The average of absolute errors (an*error*is thedifference between the predicted value and the actual value).
* **Root Mean Squared Error** (RMSE): The square root of the average of squared errorsof predictions made on the test dataset.
* **Relative Absolute Error**: The average of absolute errors relative to the absolutedifference between actual values and the average of all actual values.
* **Relative Squared Error**: The average of squared errors relative to the squareddifference between the actual values and the average of all actual values.
* **Coefficient of Determination**: Also known as the **R squared value**, this is a statisticalmetric indicating how well a model fits the data.

For each of the error statistics, smaller is better. A smaller value indicates that the predictions more closely match the actual values. For **Coefficient of Determination**, the closer its value is to one (1.0), the better the predictions.

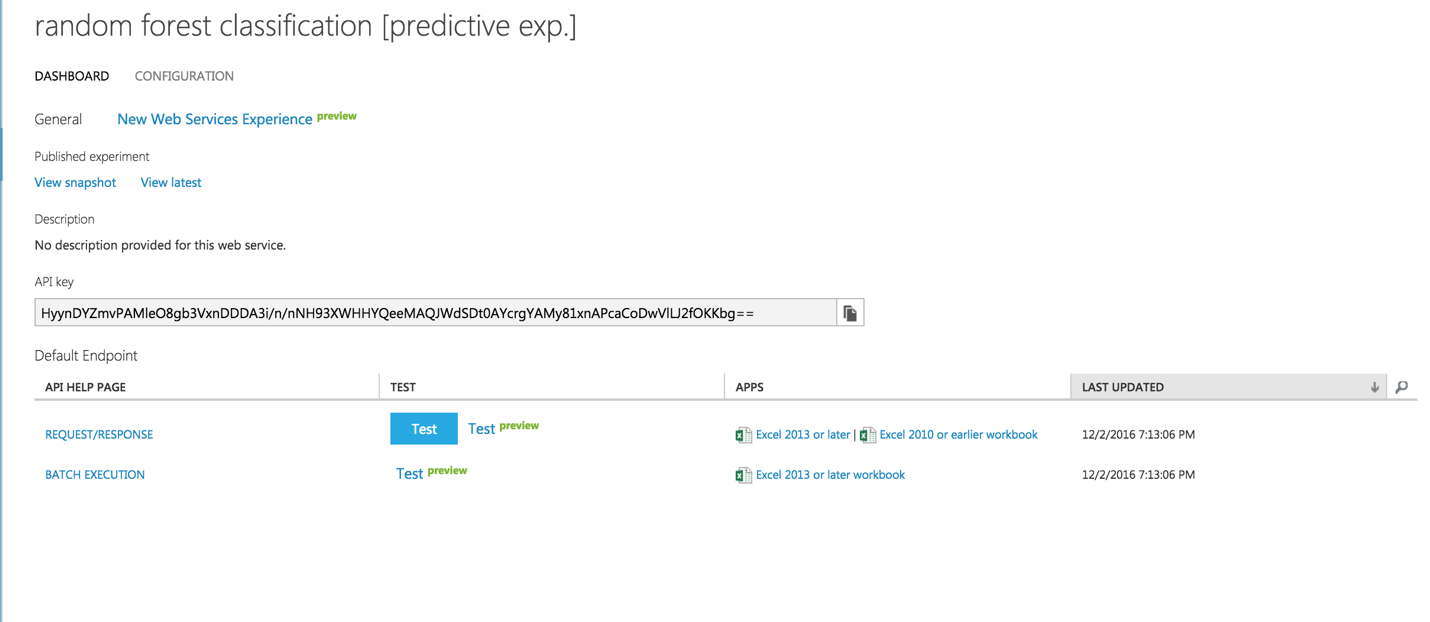


**Web Service**

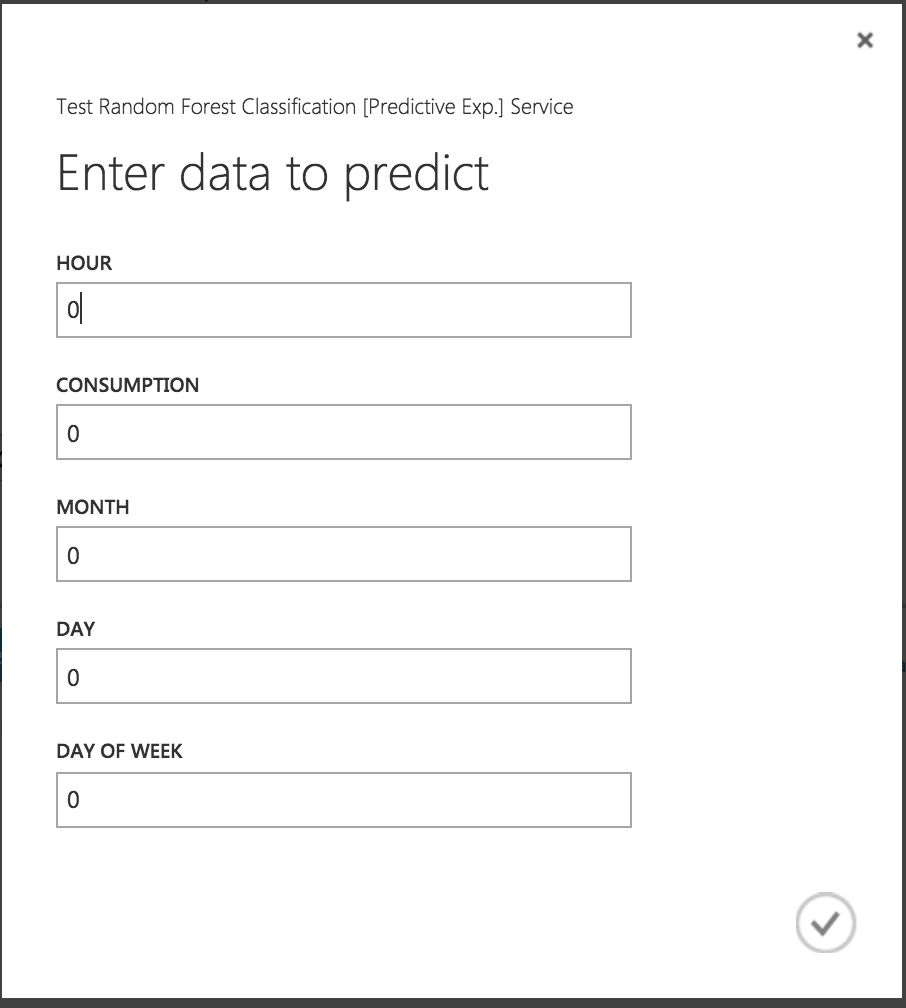
* Once the classification model is ready, we set up **Web Service**.
* The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**)
* Then we added the saved trained model back into the experiment.
* **Web service input** and **Web service output** modules are added.



* Now run the model and publish the web service



* On running the web service, we get the following form which can be used to invoke the web service and do prediction.

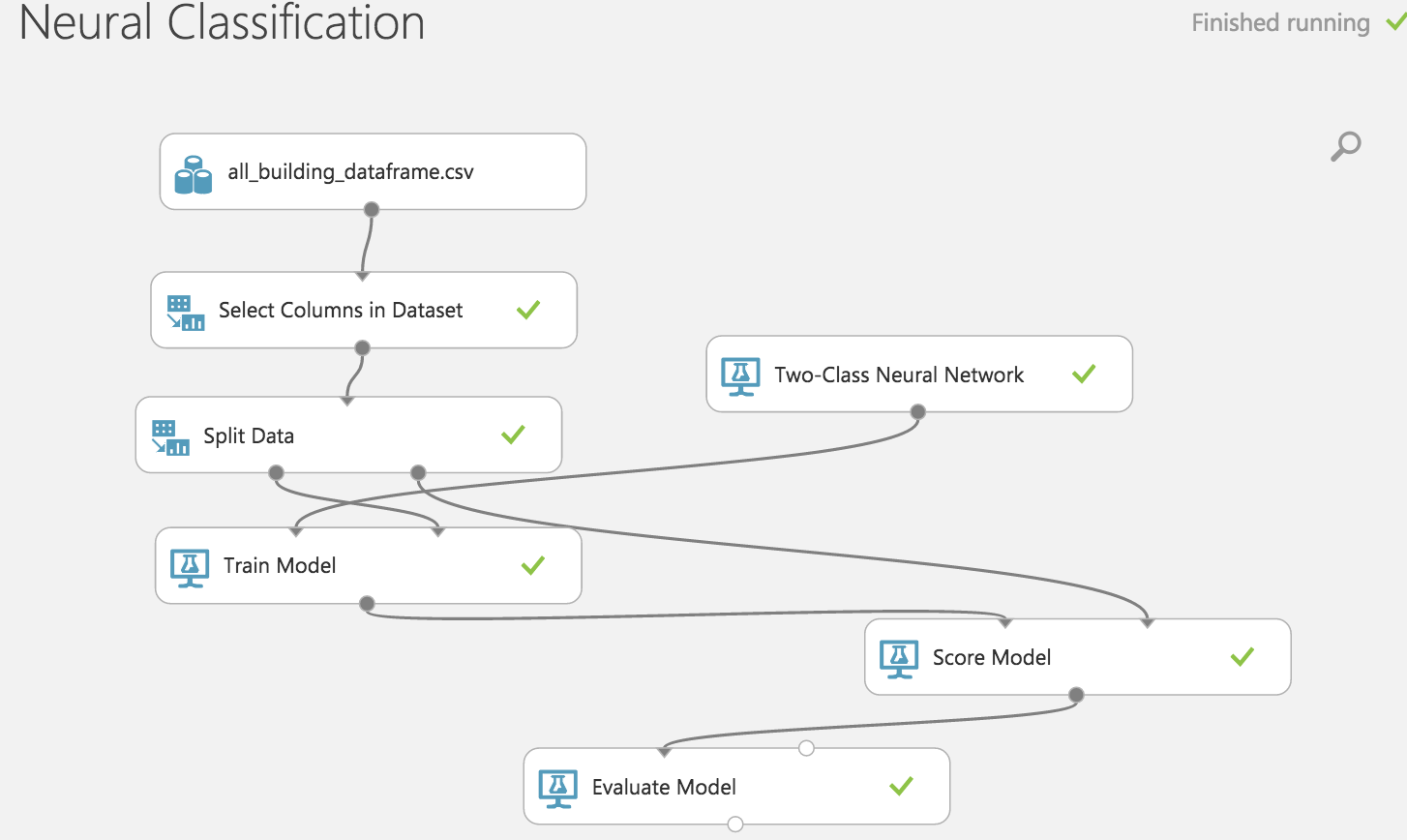


**Two class Neural network**

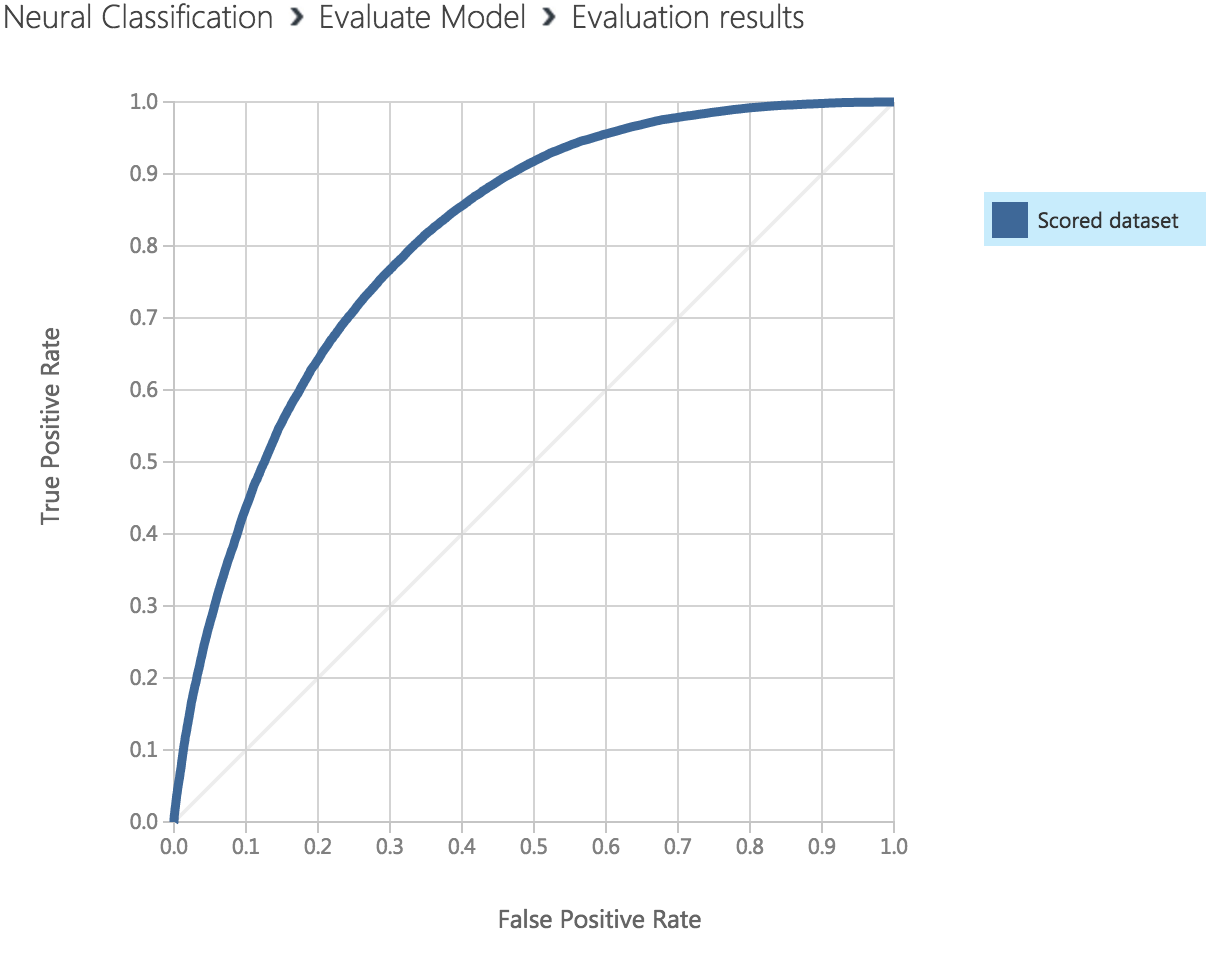
A neural network is a set of interconnected layers, in which the inputs lead to outputs by a series of weighted edges and nodes. The weights on the edges are learned when training the neural network on the input data. The direction of the graph proceeds from the inputs through the hidden layer, with all nodes of the graph connected by the weighted edges to nodes in the next layer. Most predictive tasks can be accomplished easily with only one or a few hidden layers.

Recent research has shown that deep neural networks (DNN) can be very effective in complex tasks such as image or speech recognition, in which successive layers are used to model increasing levels of semantic depth. To compute the output of the network for any given input, a value is calculated for each node in the hidden layers and in the output layer. For each node, the value is set by calculating the weighted sum of the values of the nodes in the previous layer and applying an activation function to that weighted sum

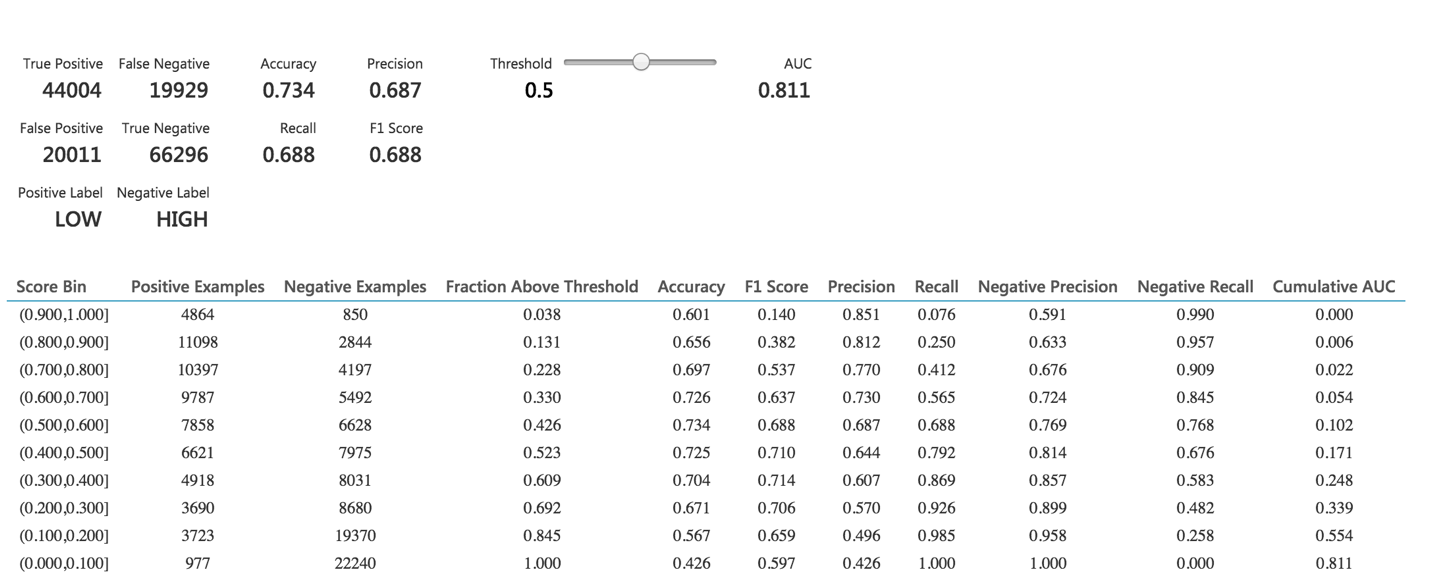
**Azure Modules:**



**ROC Curve:**

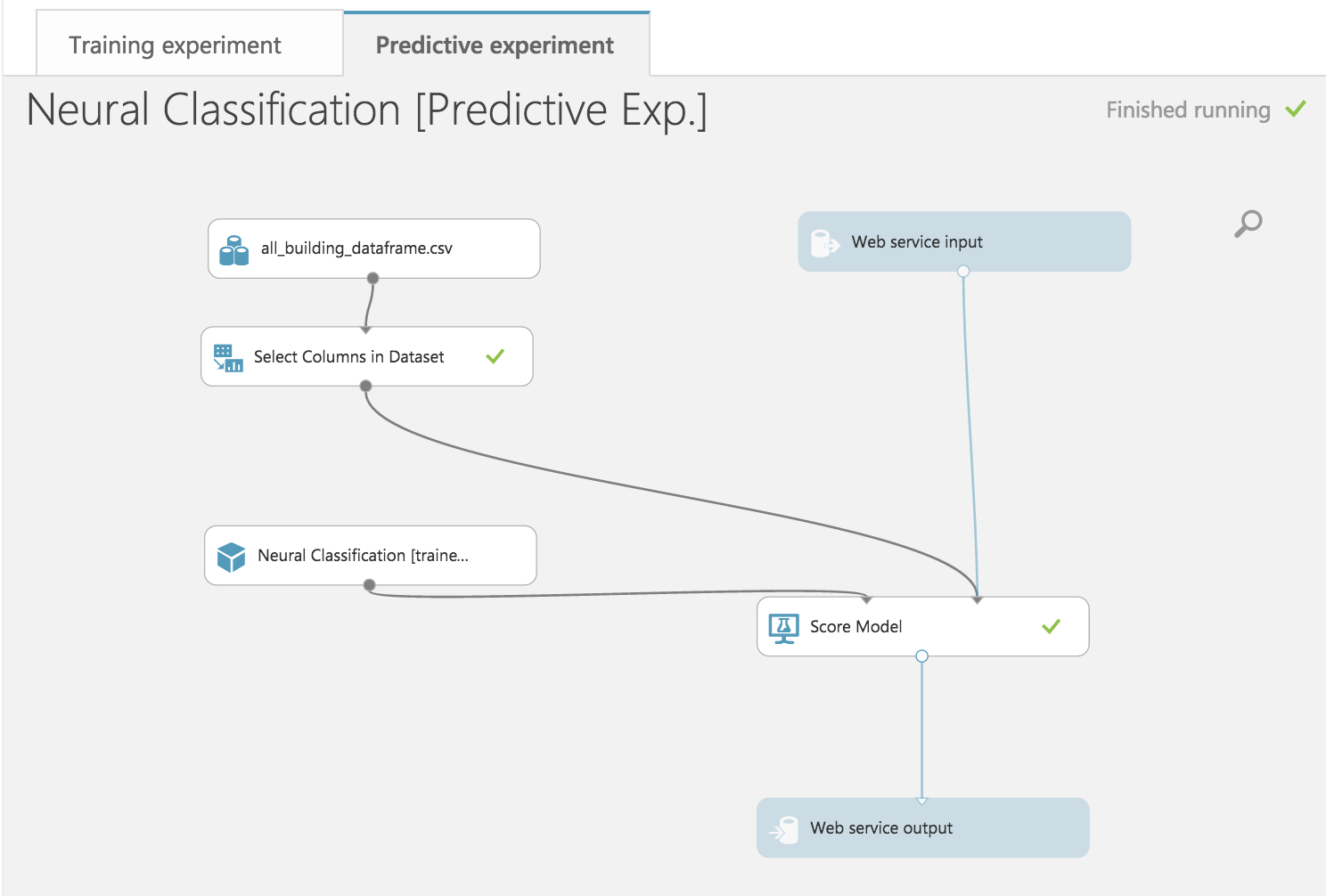
****

**Confusion Matrix and other performance metrics :**

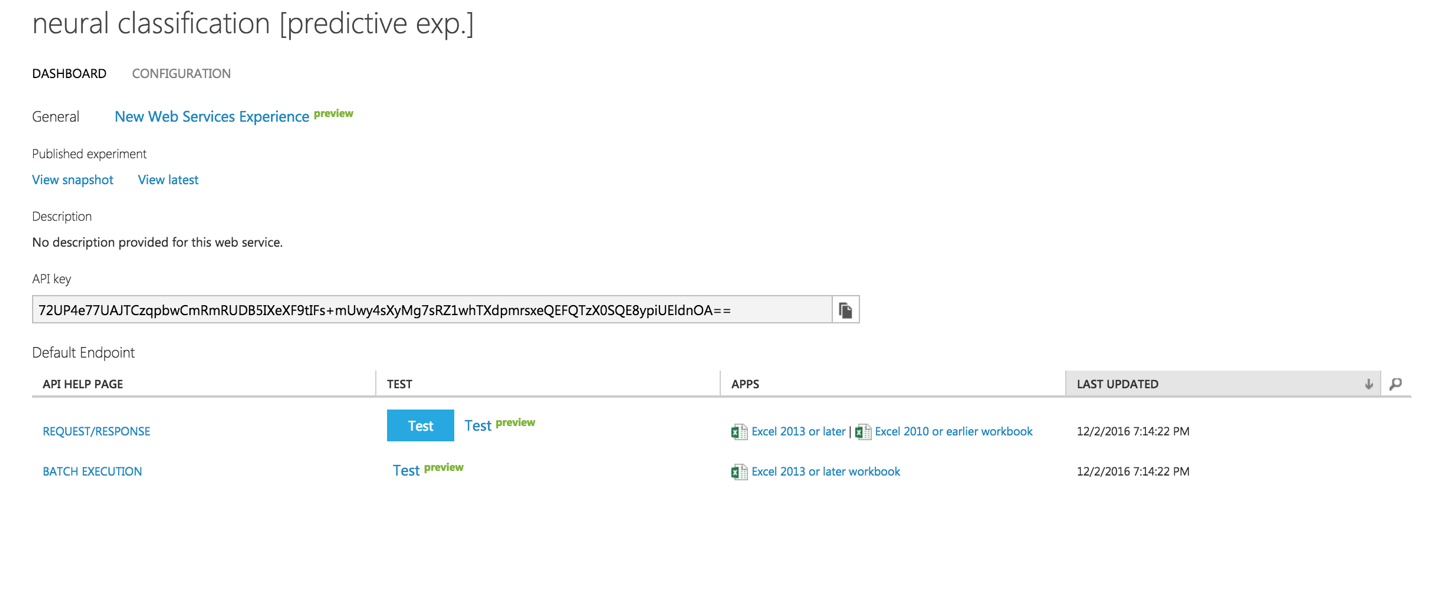


**Web Service**

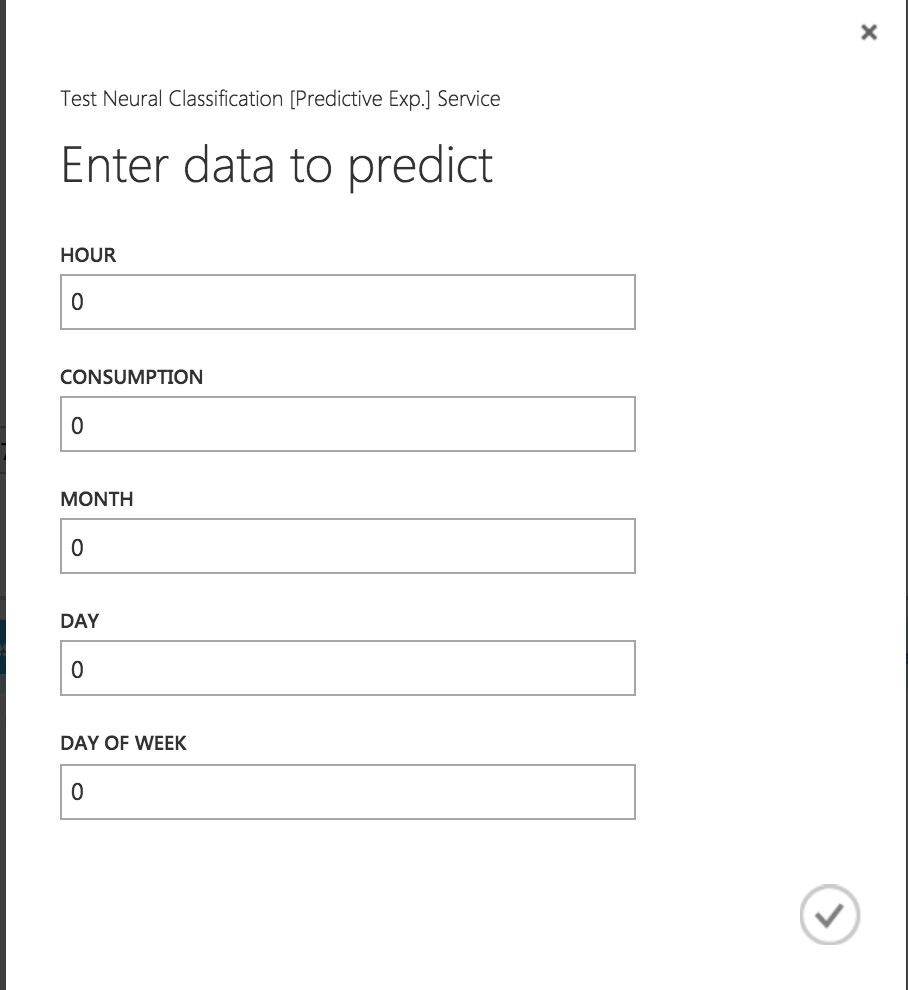
* Once the classification model is ready, we set up **Web Service**.
* The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**)
* Then we added the saved trained model back into the experiment.
* **Web service input** and **Web service output** modules are added.



* Now run the model and publish the web service



On running the web service, we get the following form which can be used to invoke the web service and do prediction

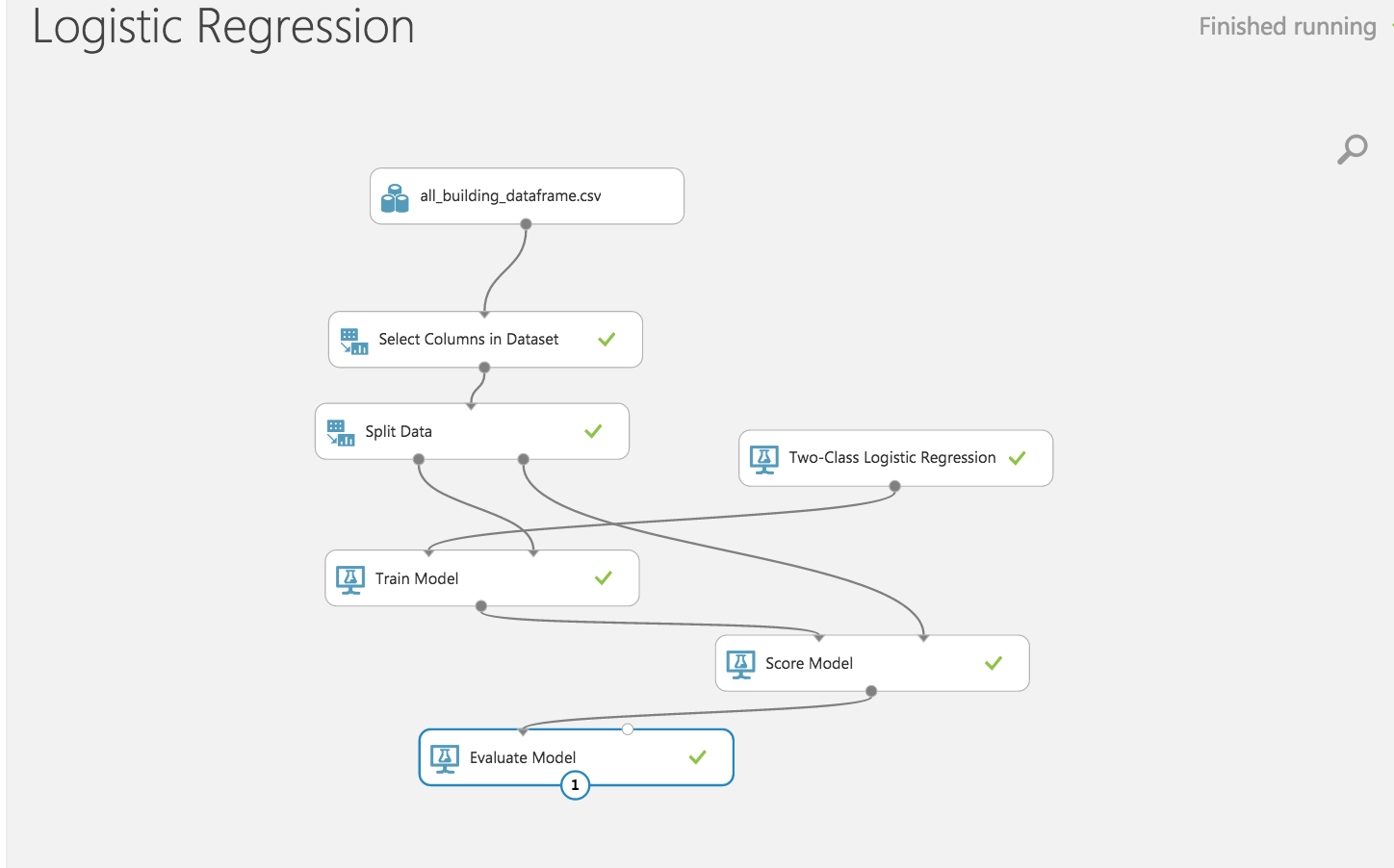
****

**Two class Logistic Regression**

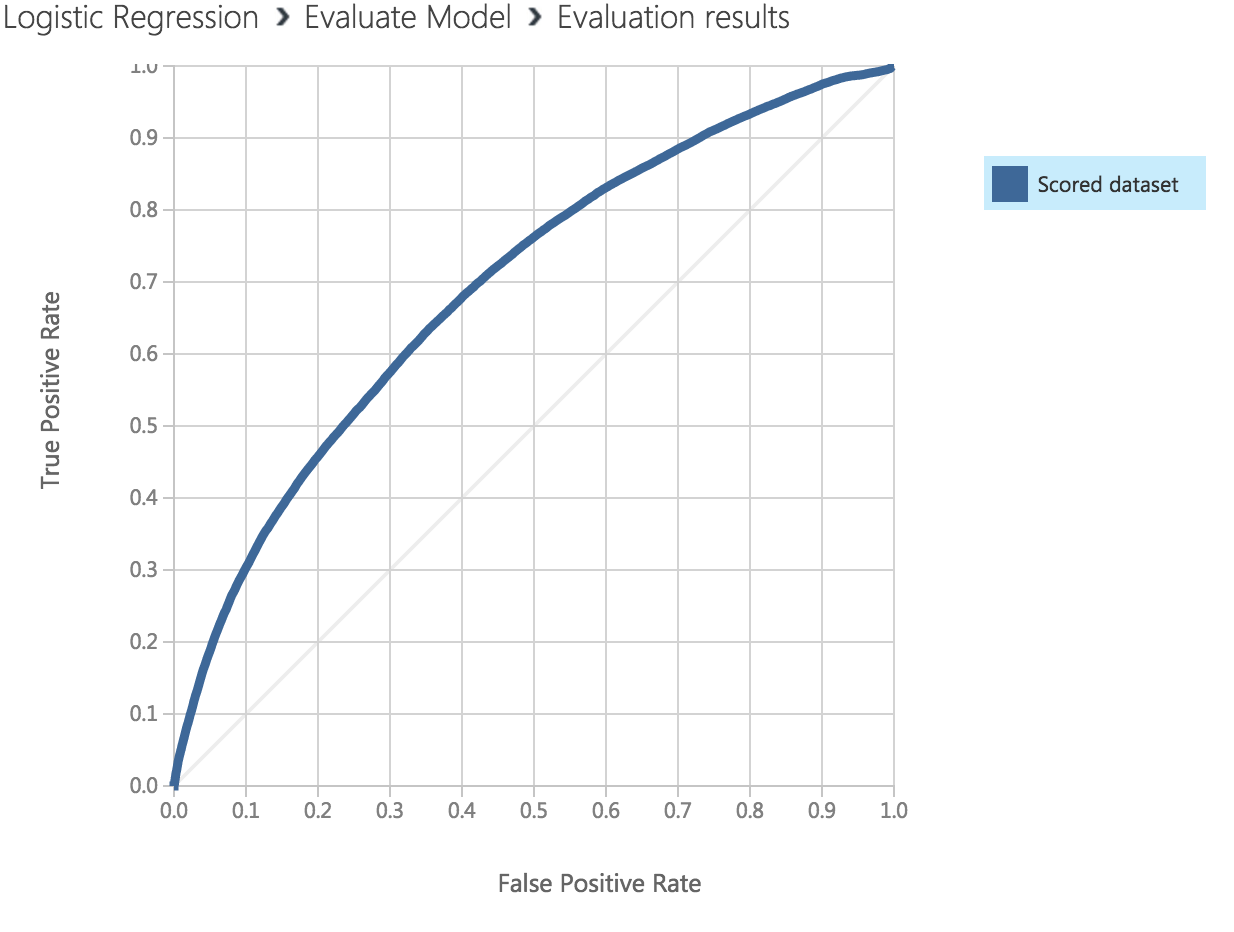
Logistic regression is a well-known method in statistics that is used to predict the probability of an outcome, and is especially popular for classification tasks. The algorithm predicts the probability of occurrence of an event by fitting data to a logistic function.

Logistic regression requires numeric variables. Therefore, when you use categorical columns as variable, Azure Machine Learning converts the values to an indicator array internally.

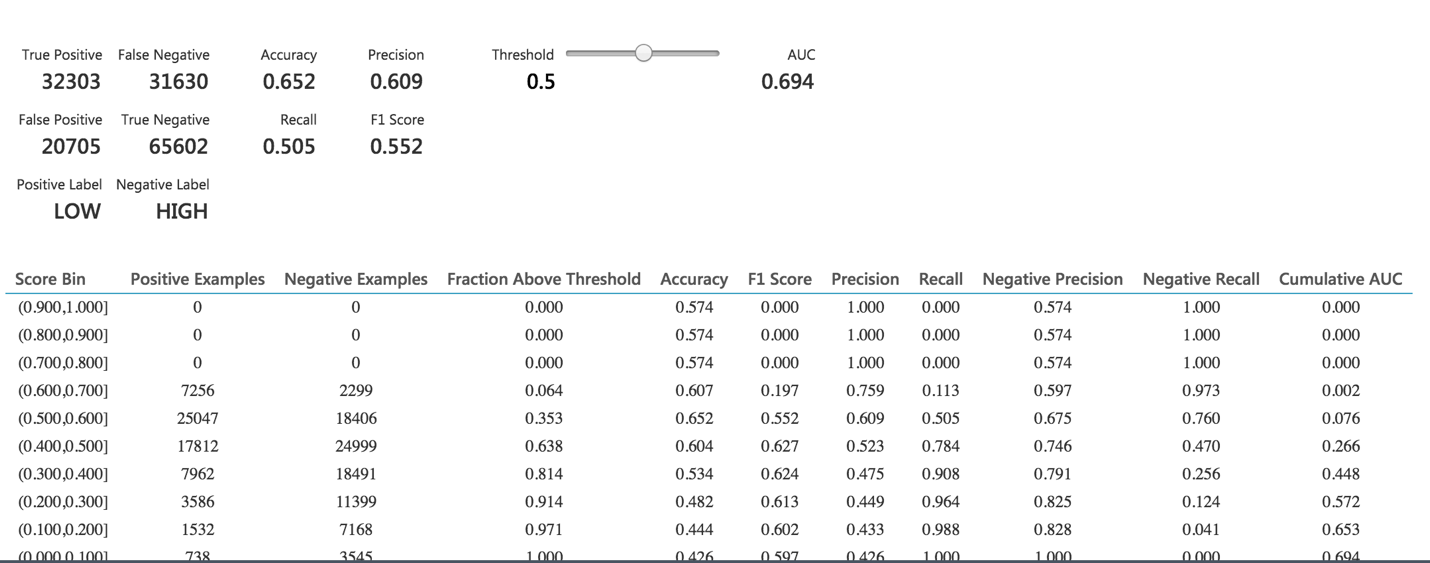
**Azure Model:**

****

**ROC:**

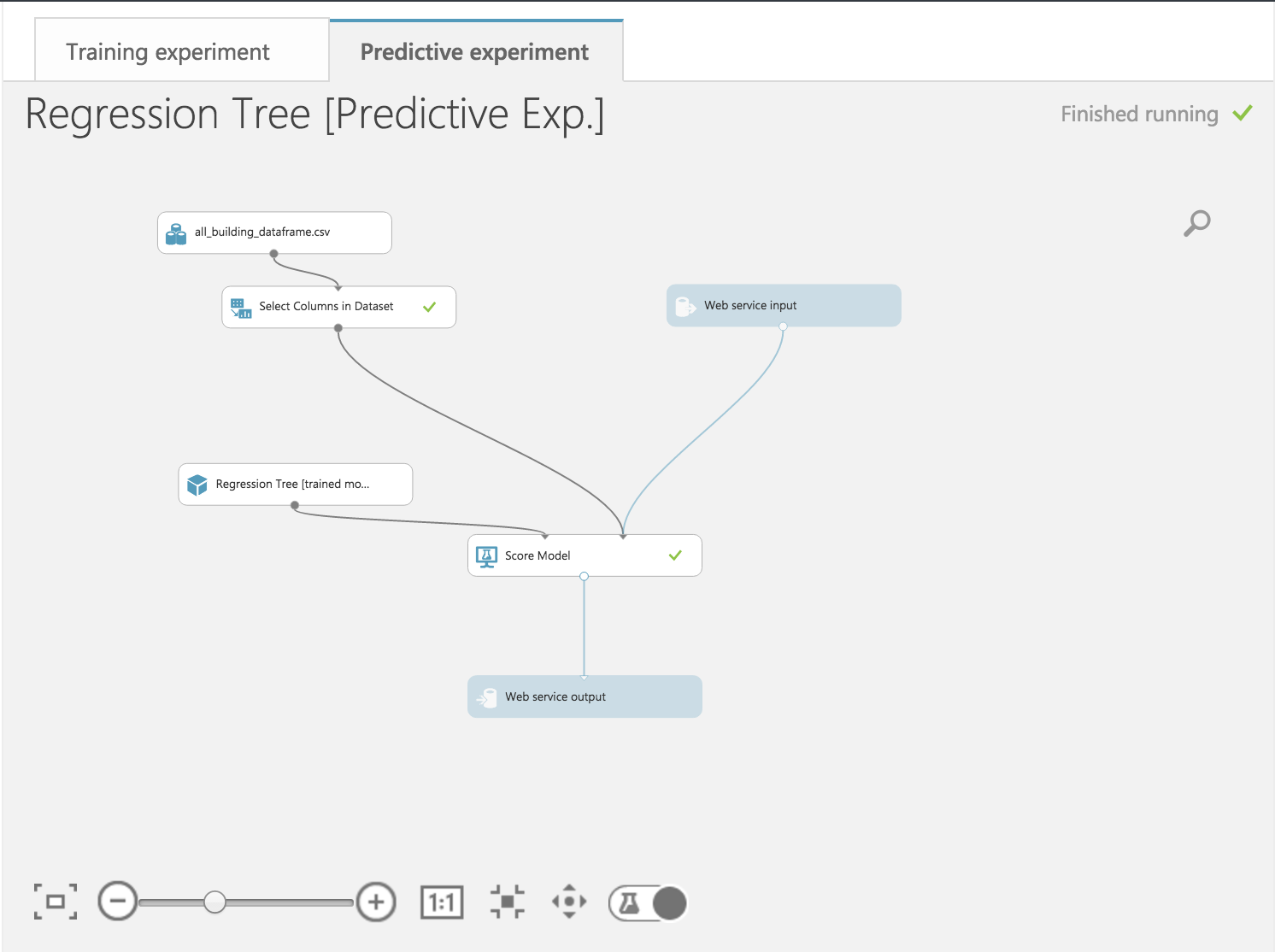
****

**Confusion Matrix:**

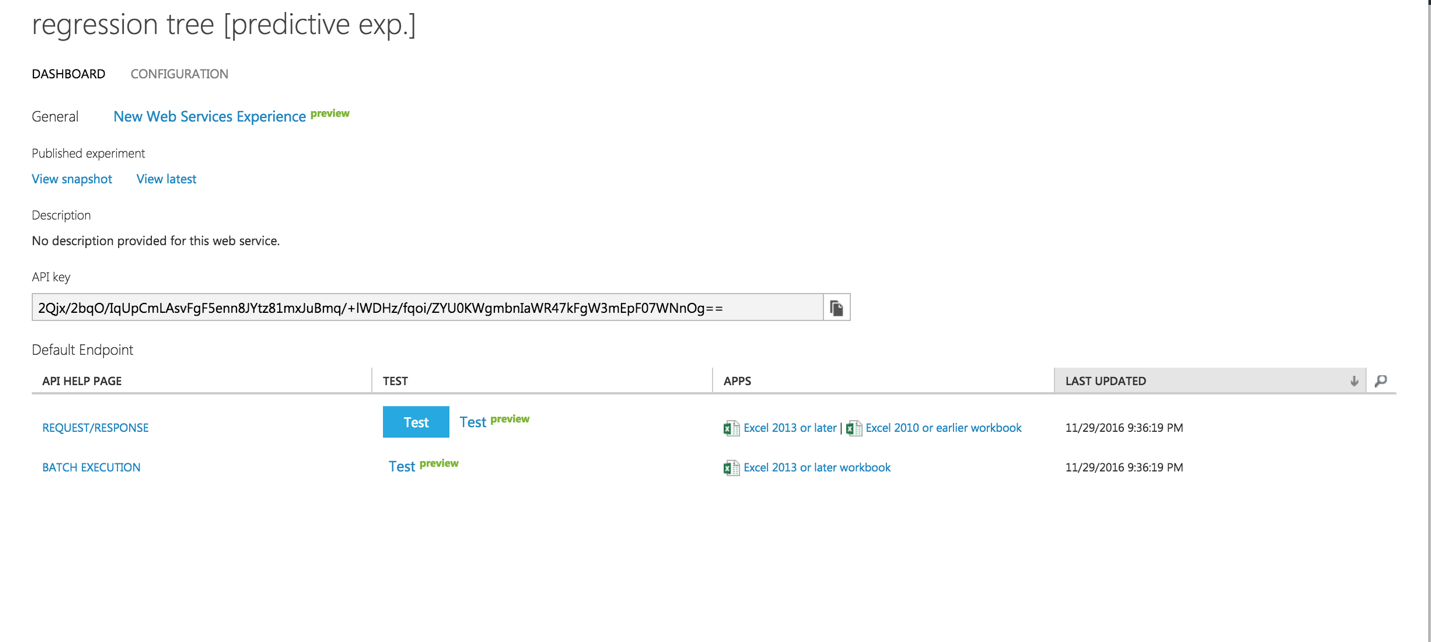
****

**Web Service**

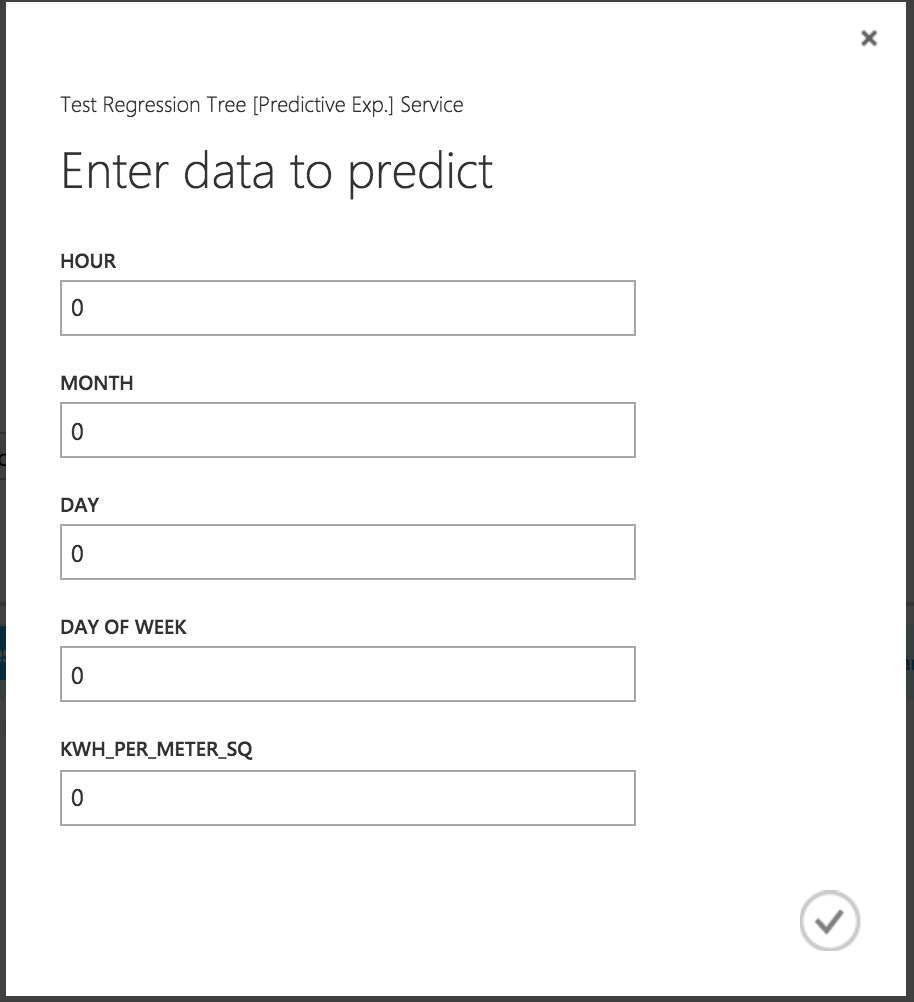
* Once the classification model is ready, we set up **Web Service**.
* The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**)
* Then we added the saved trained model back into the experiment.
* **Web service input** and **Web service output** modules are added



* Now run the model and publish the web service

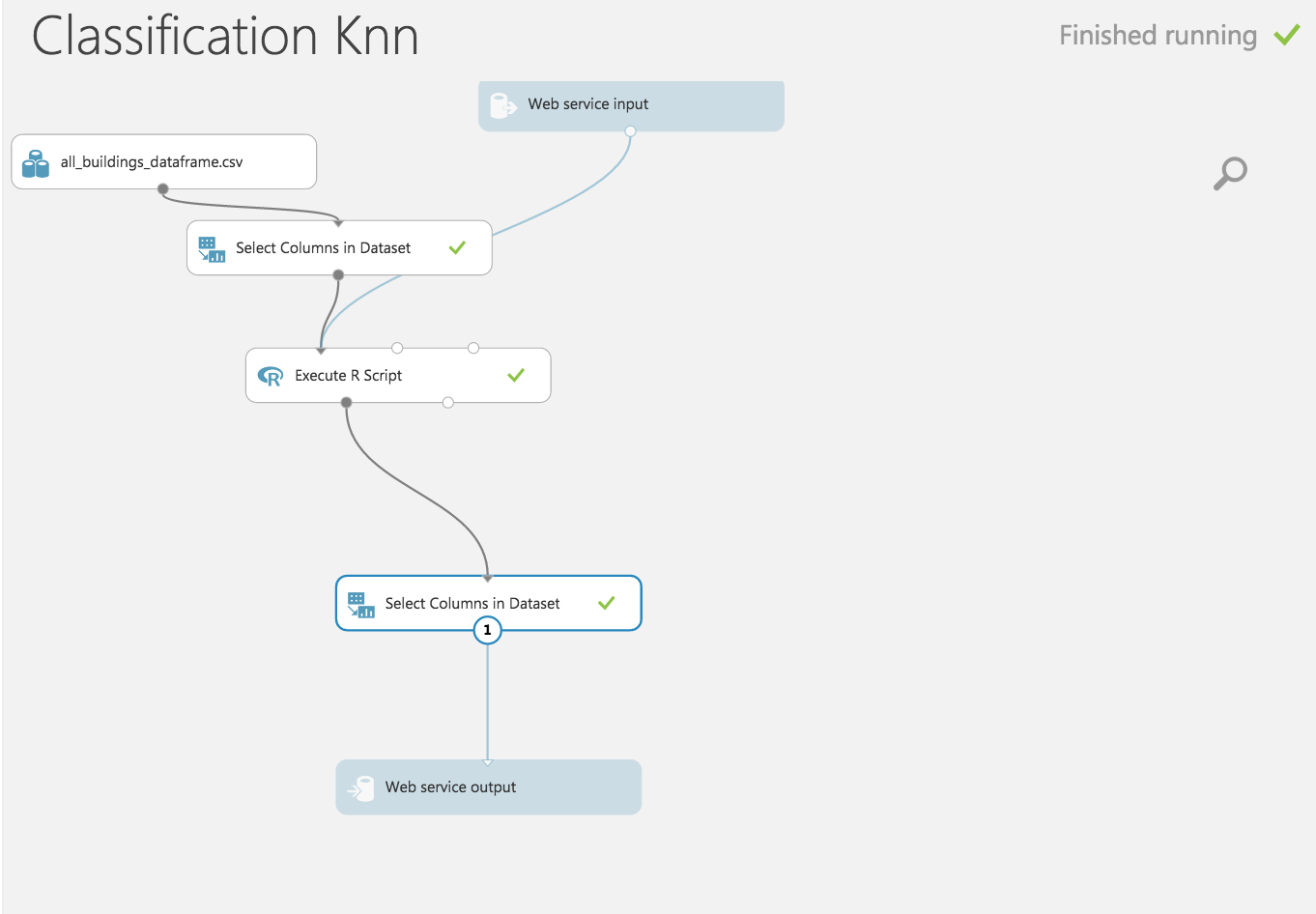


On running the web service, we get the following form which can be used to invoke the web service and do prediction



**Two class KNN Classification**

**Azure Model:**

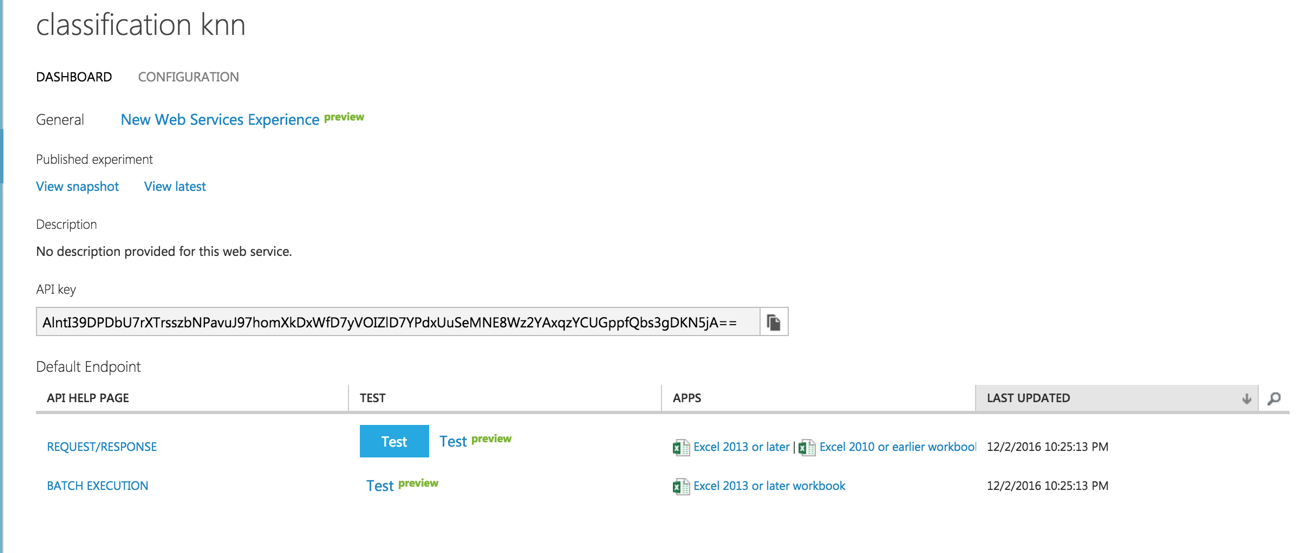
****

**R Script:**

****

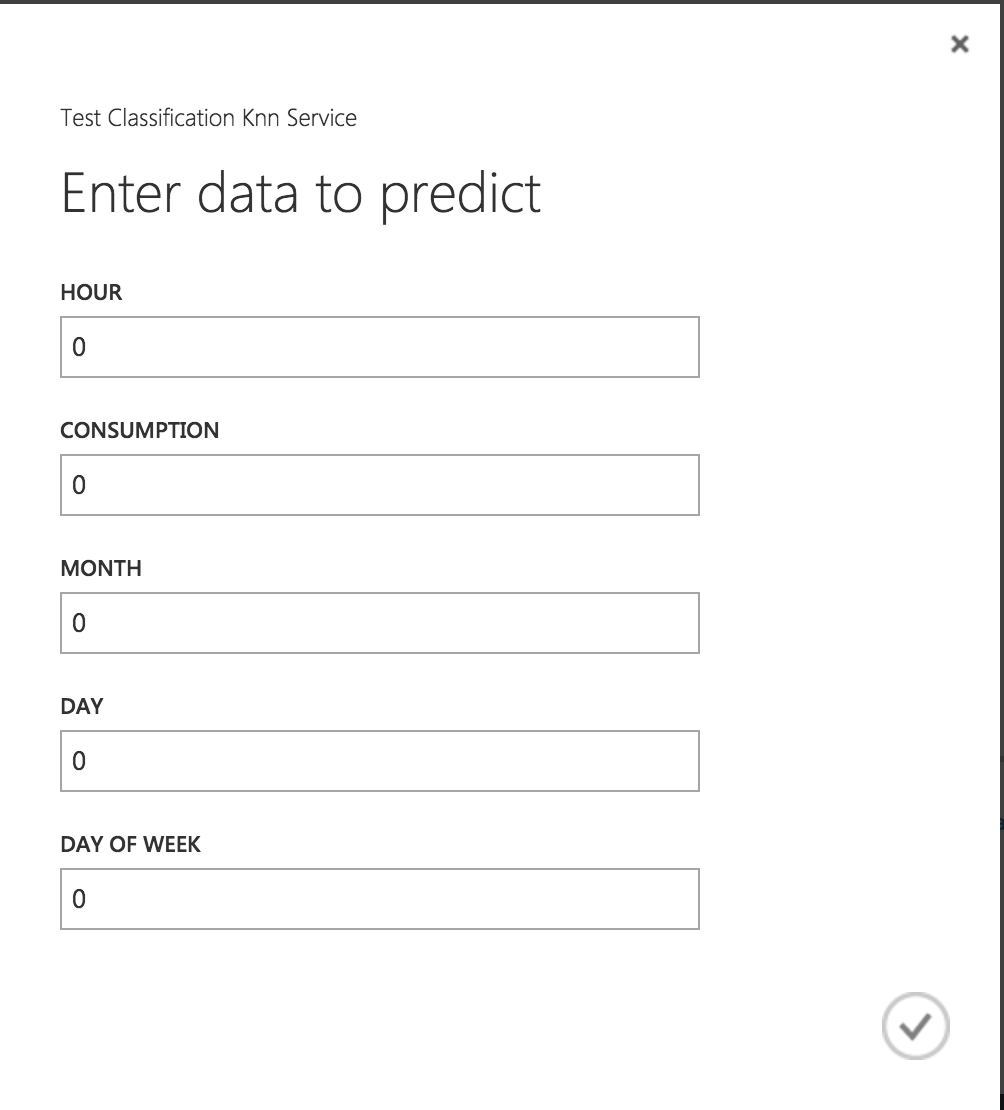
**Web Service**

* Once the classification model is ready, we set up **Web Service**.
* The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**)
* Then we added the saved trained model back into the experiment.
* **Web service input** and **Web service output** modules are added

****

On running the web service, we get the following form which can be used to invoke the web service and do prediction

**Conclusion:** Thus from above it’s clear that the best model amongthe classificationmodels is Two class Decision Tree as it has high accuracy rate at 85.2%. Also, the Area Under Curve (AUC) is highest in Two class Decision Tree .

****

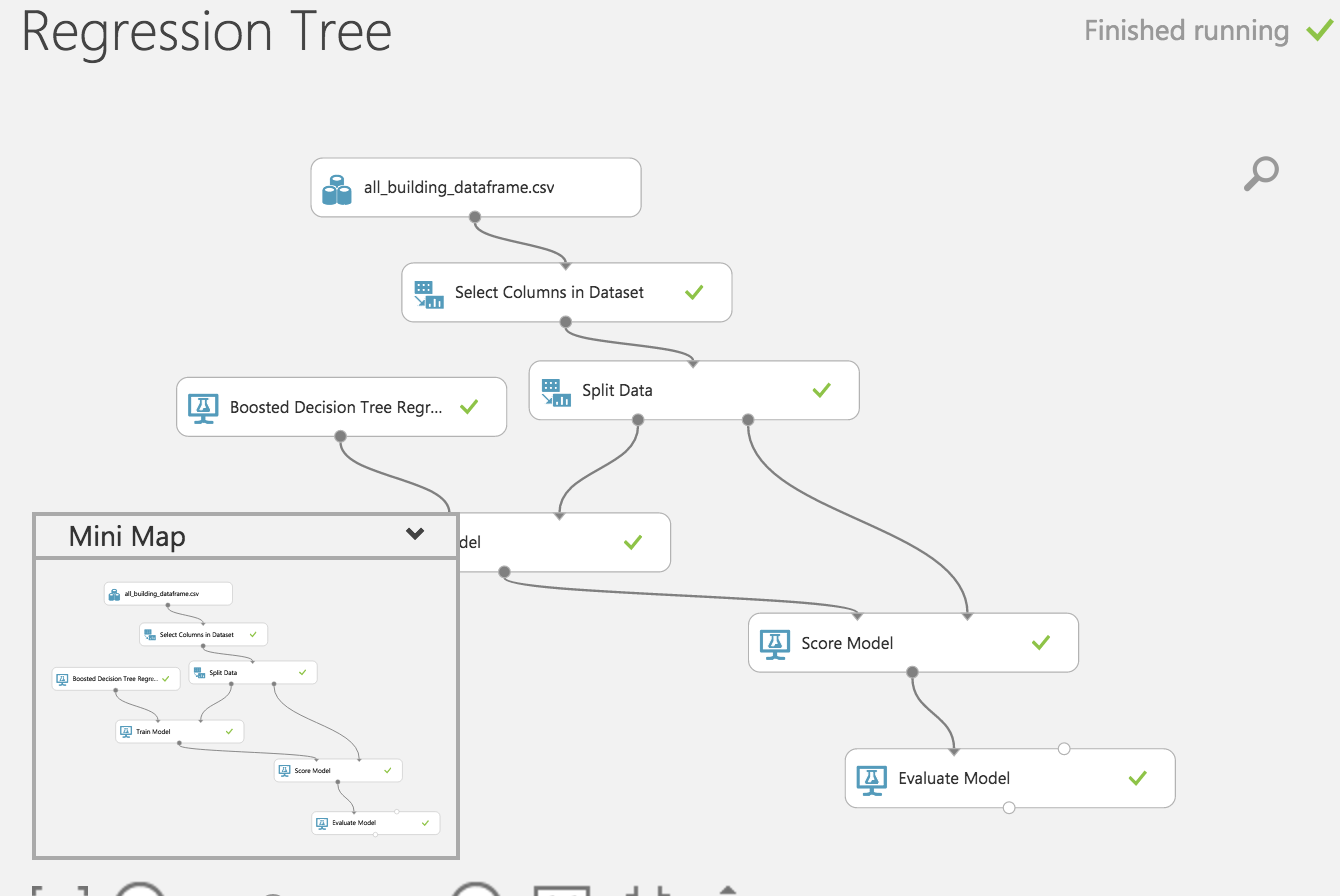
**Regression Models**

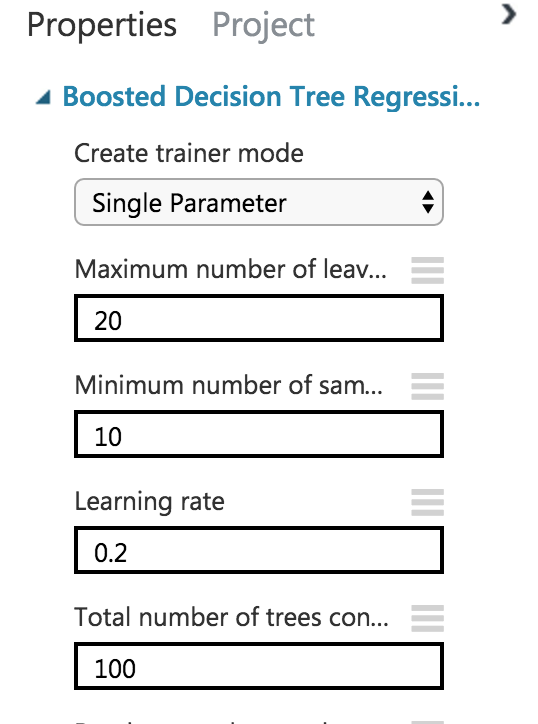
**Boosted Decision Tree Regression**

use the Boosted Decision Tree Regression module to create an ensemble of regression trees using boosting. Boosting means that each tree is dependent on prior trees, and learns by fitting the residual of the trees that preceded it. Thus, boosting in a decision tree ensemble tends to improve accuracy with some small risk of less coverage. This regression method is a supervised learning method, and therefore requires a labeled dataset. The label column must contain numerical values

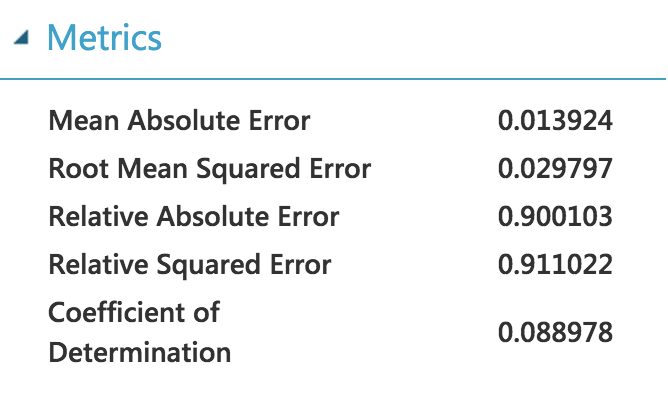
**Azure models**

Below is the telecommunication network clustering model created.





**Performance metrics :**



**Web Service**

* Once the classification model is ready, we set up **Web Service**.
* The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**)
* Then we added the saved trained model back into the experiment.
* **Web service input** and **Web service output** modules are added.
* Now run the model and publish the web service

On running the web service, we get the following form which can be used to invoke the web service and do prediction

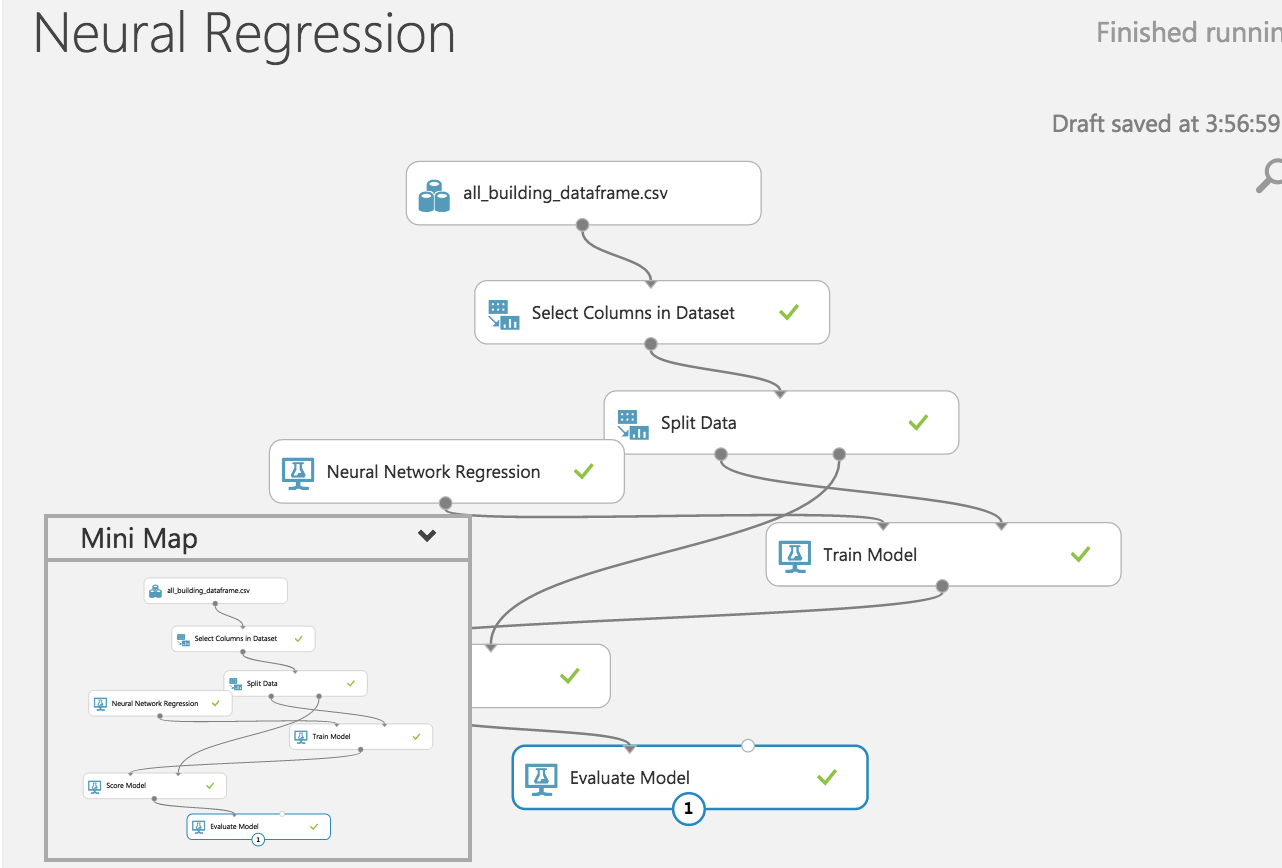
**Neural Network Regression**

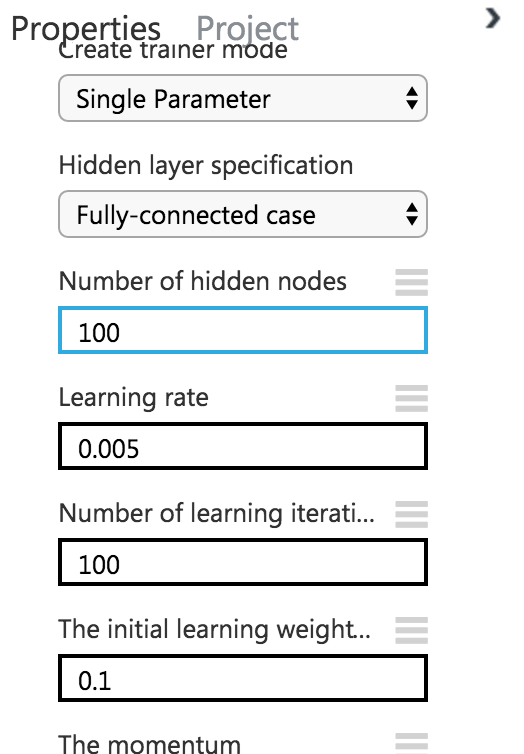
Neural Network regression is a well-known method in statistics that is used to predict the probability of an outcome, and is especially popular for classification tasks. The algorithm predicts the probability of occurrence of an event by fitting data to a logistic function.

Neural Network regression requires numeric variables. Therefore, when you use categorical columns as variable, Azure Machine Learning converts the values to an indicator array internally.

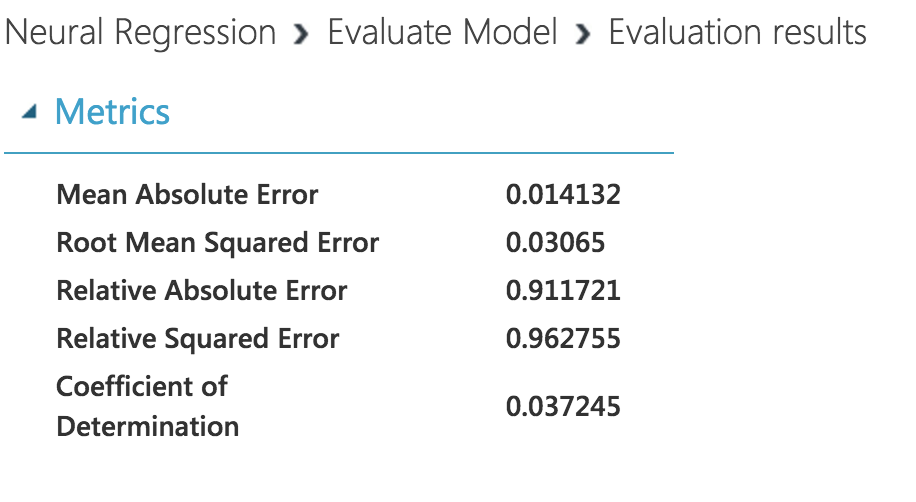
**Azure models**

Below is the telecommunication network clustering model created.





**Performance metrics :**

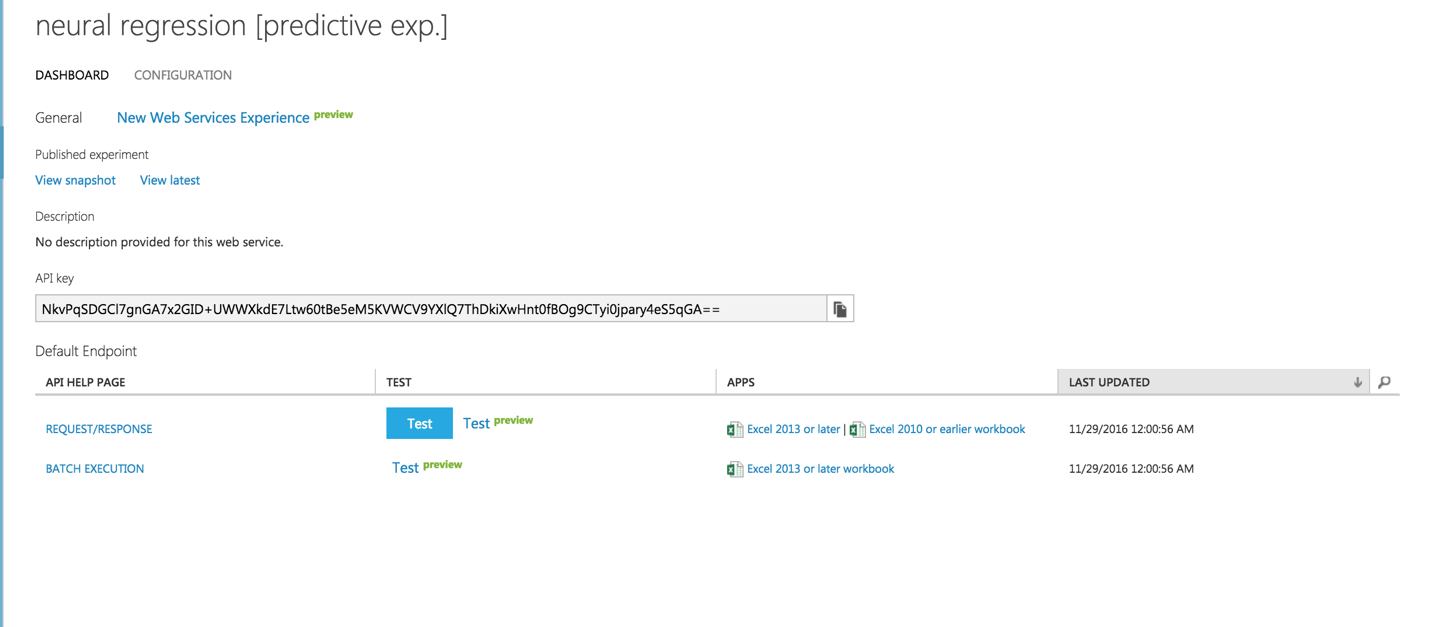


**Web Service**

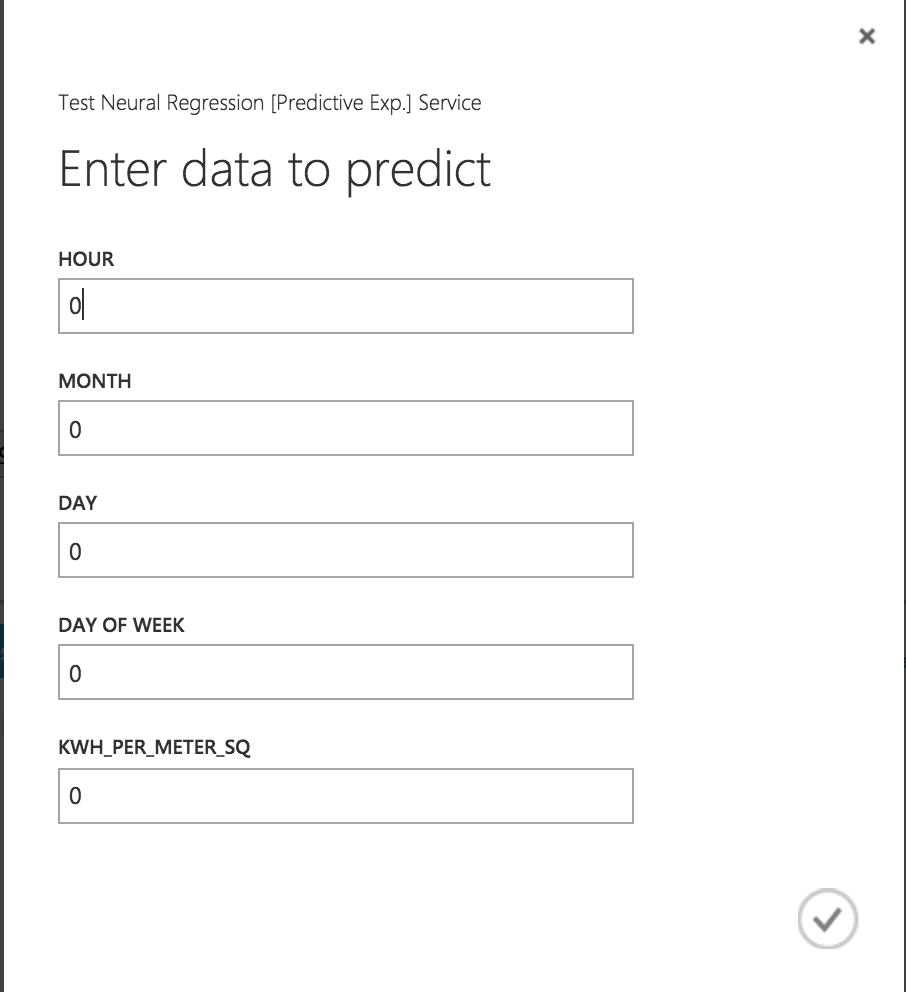
* Once the classification model is ready, we set up **Web Service**.
* The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**)
* Then we added the saved trained model back into the experiment.
* **Web service input** and **Web service output** modules are added.



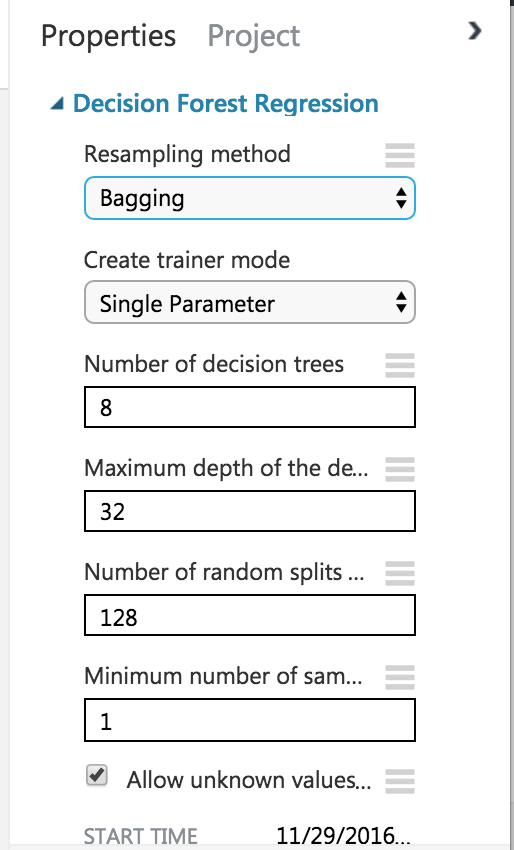
* Now run the model and publish the web service



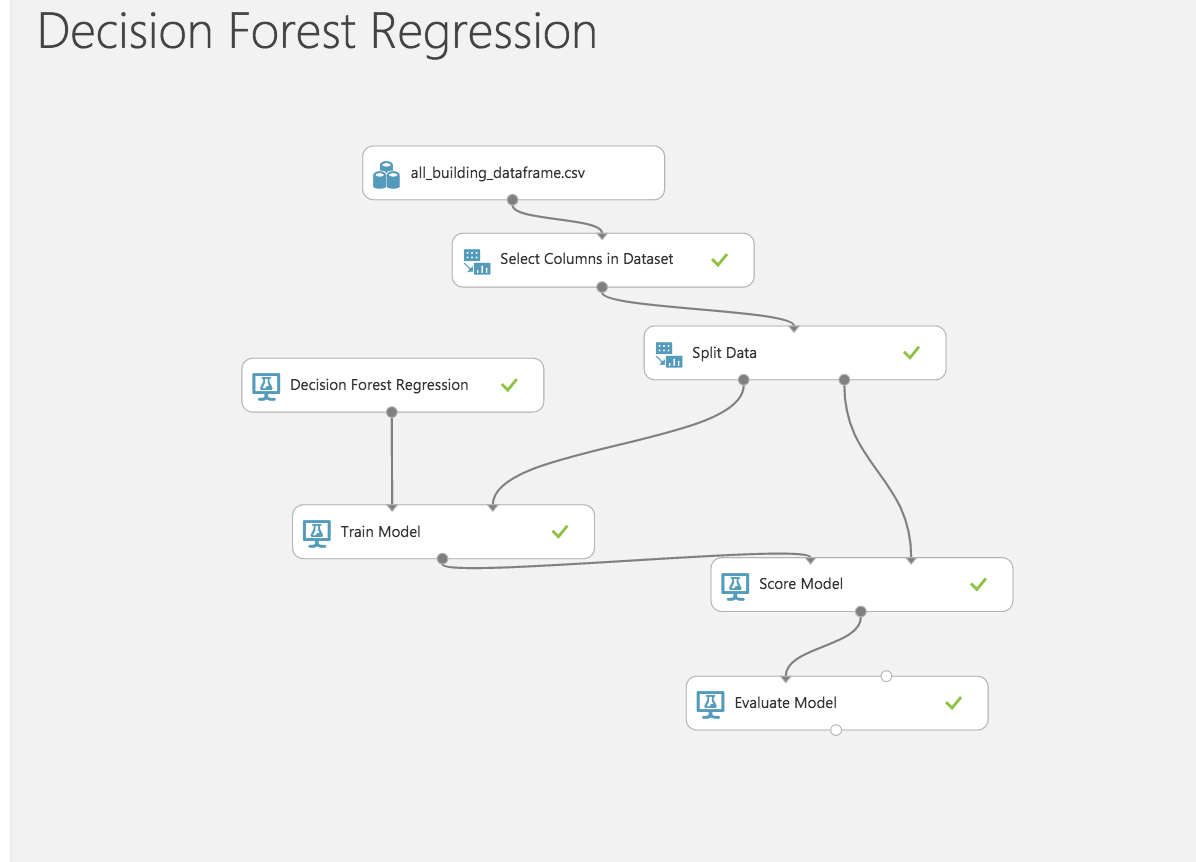
On running the web service, we get the following form which can be used to invoke the web service and do prediction

****

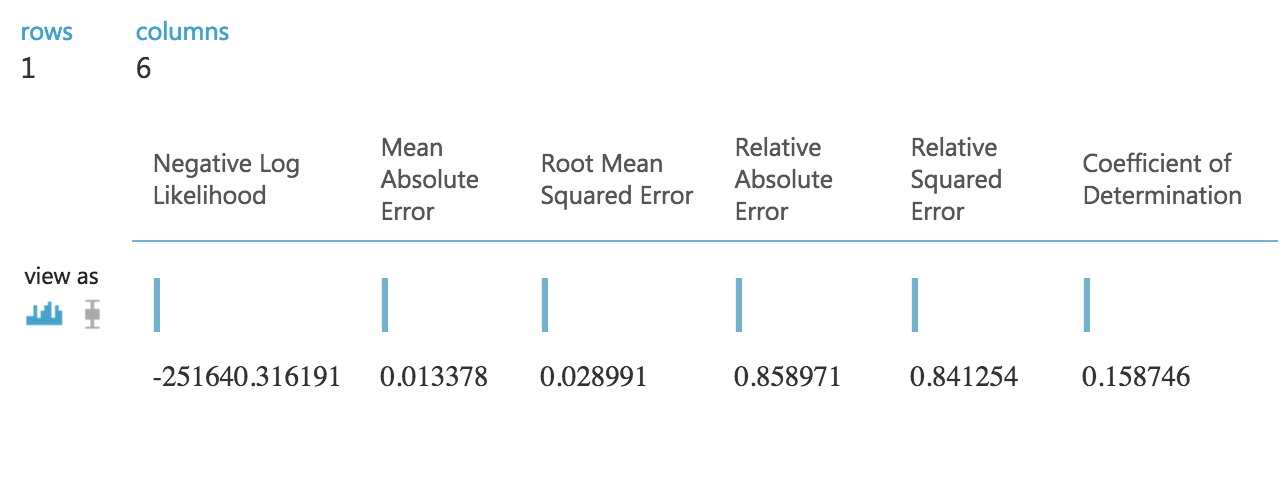
**Decision Forest Regression**

****

**Azure Model:**

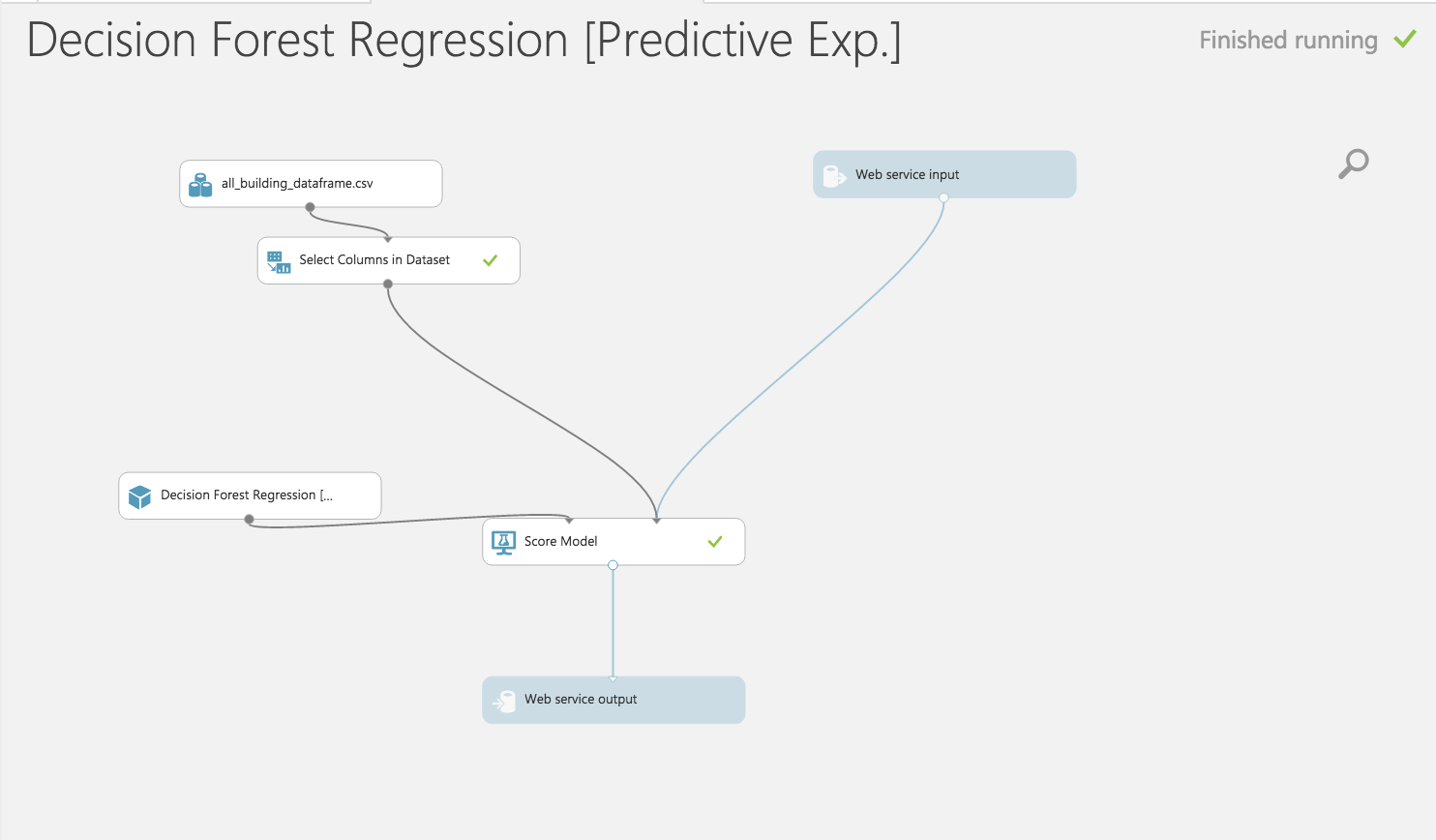
****

**Performance Matrix:**

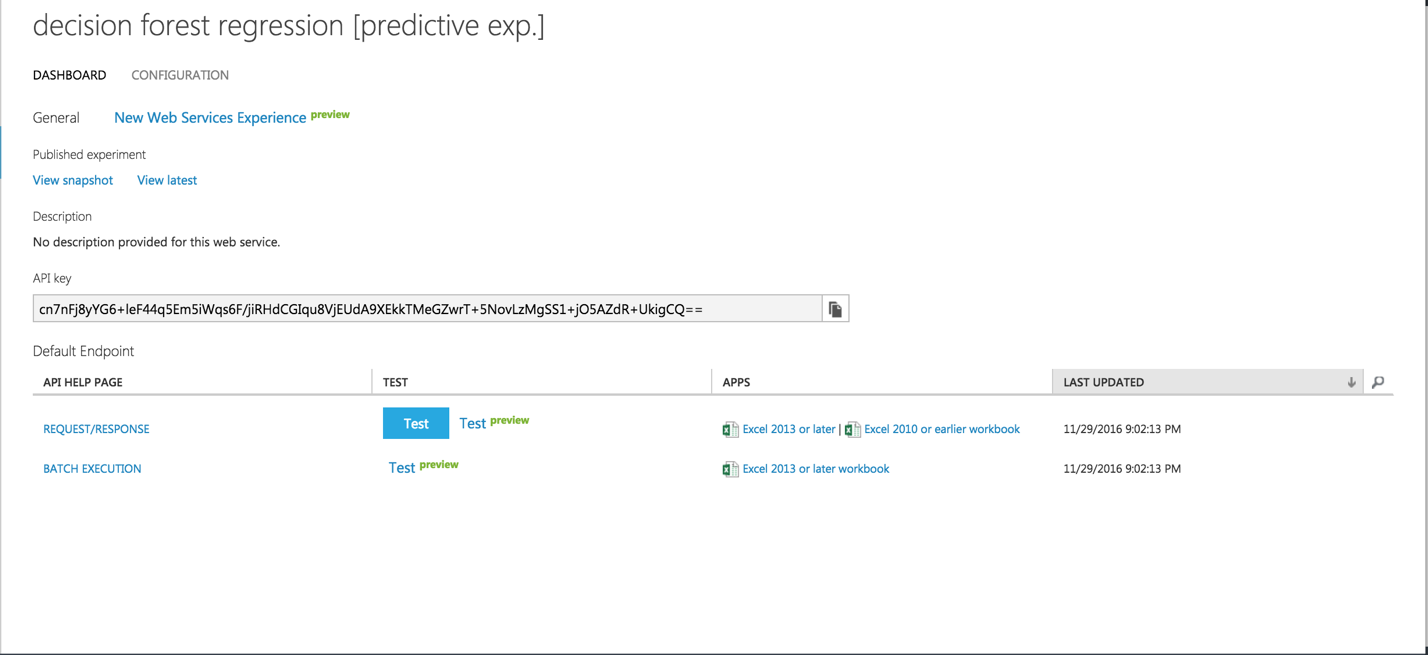


**Web Service**

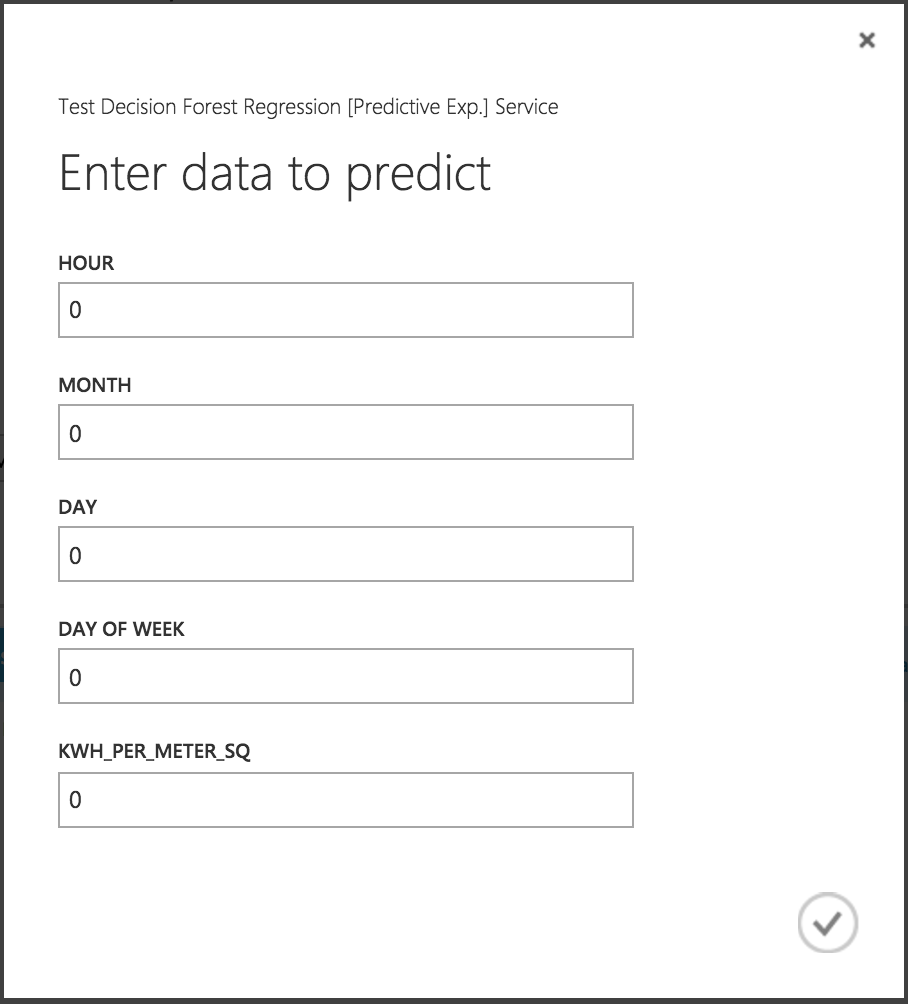
* Once the classification model is ready, we set up **Web Service**.
* The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**)
* Then we added the saved trained model back into the experiment.
* **Web service input** and **Web service output** modules are added.



* Now run the model and publish the web service

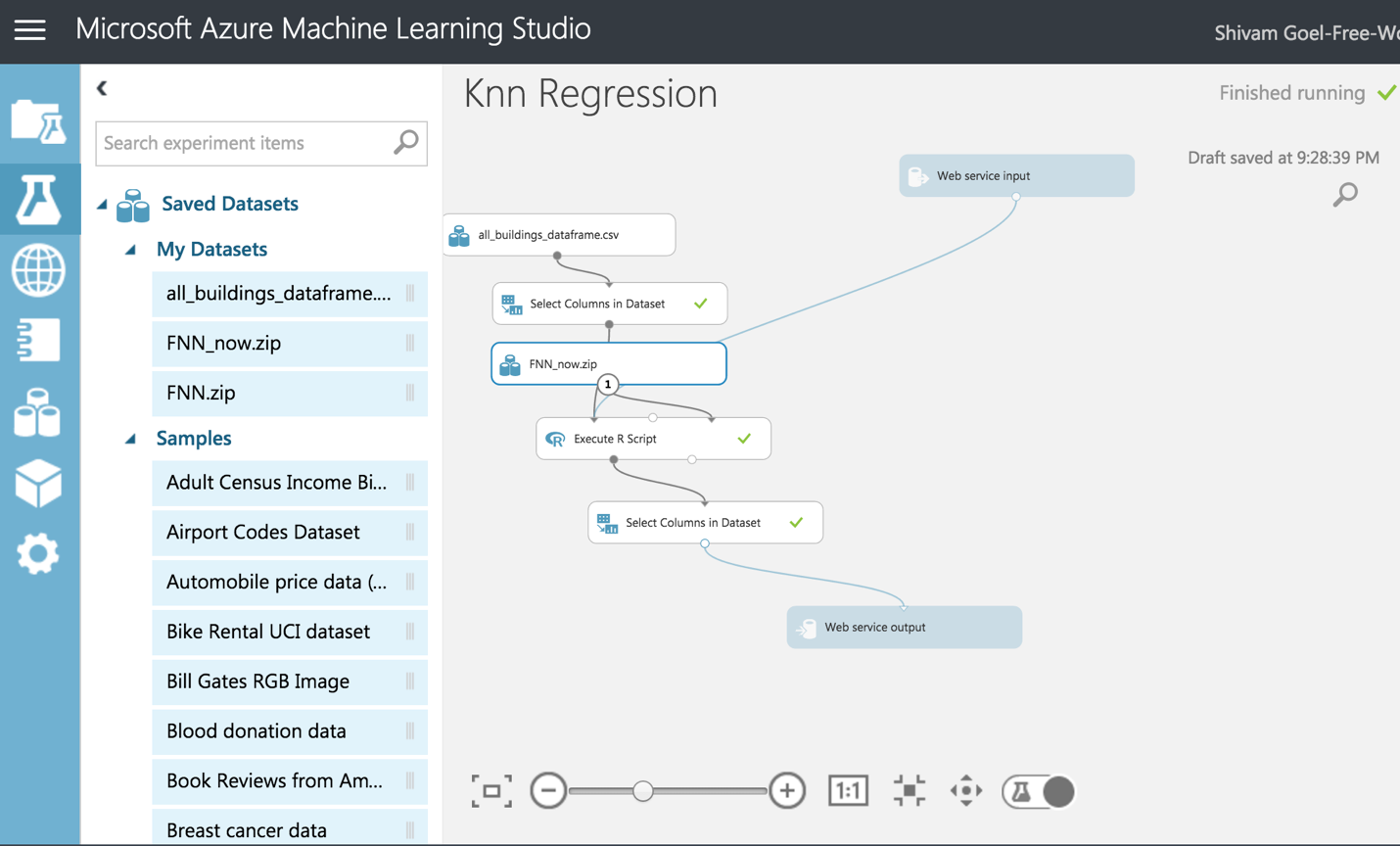


On running the web service, we get the following form which can be used to invoke the web service and do prediction

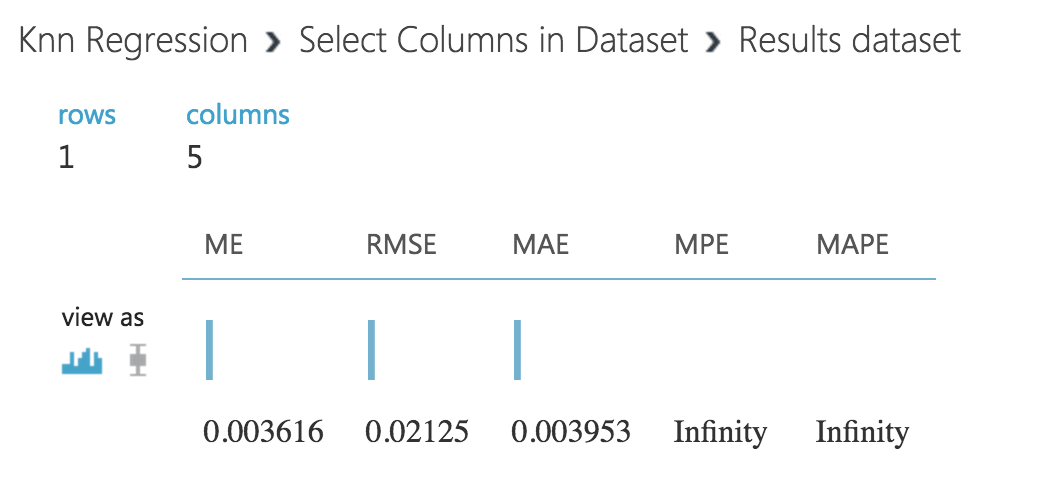
****

**KNN Regression Model**

**Azure Model:**

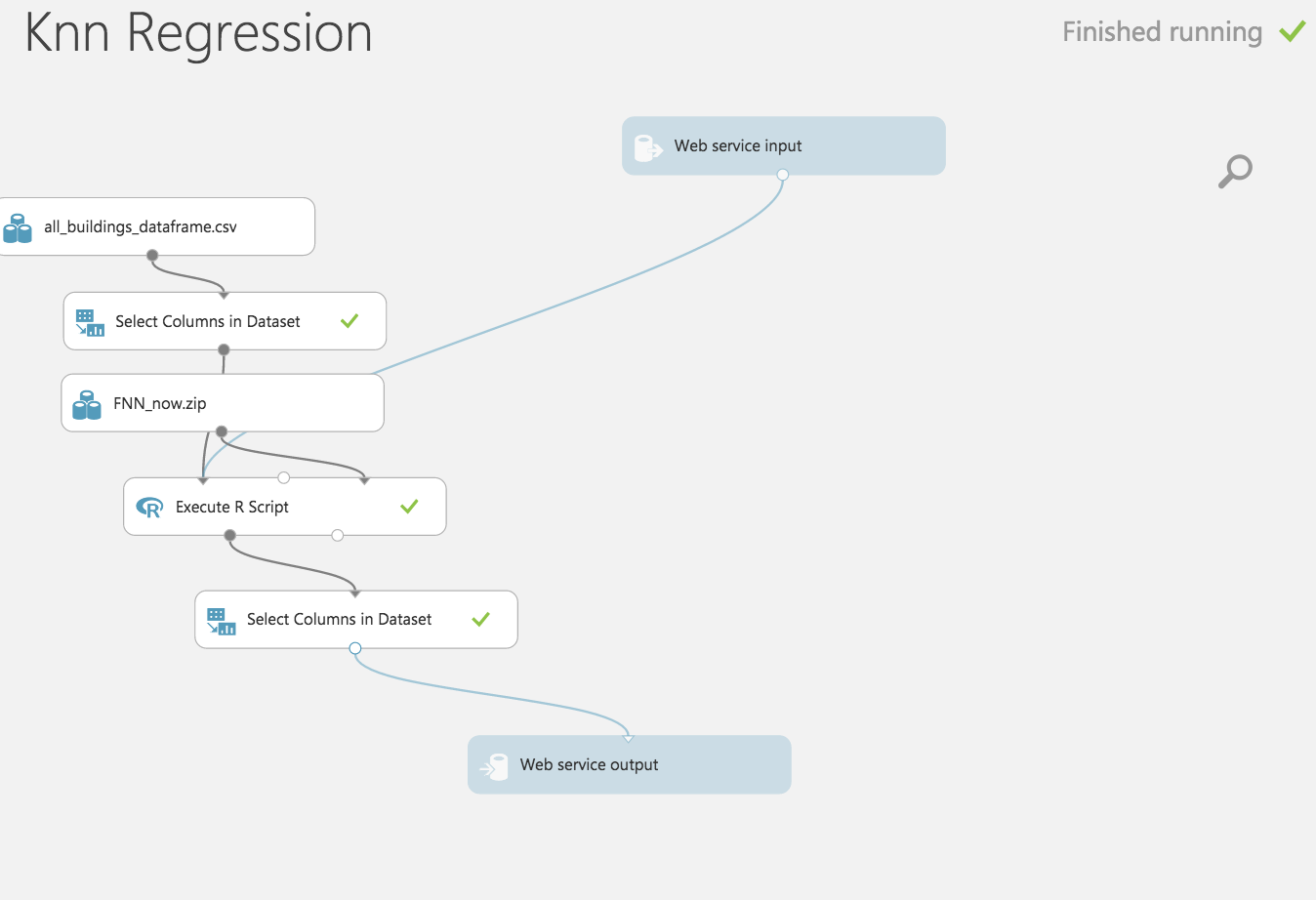
****

**Performance Matrix:**

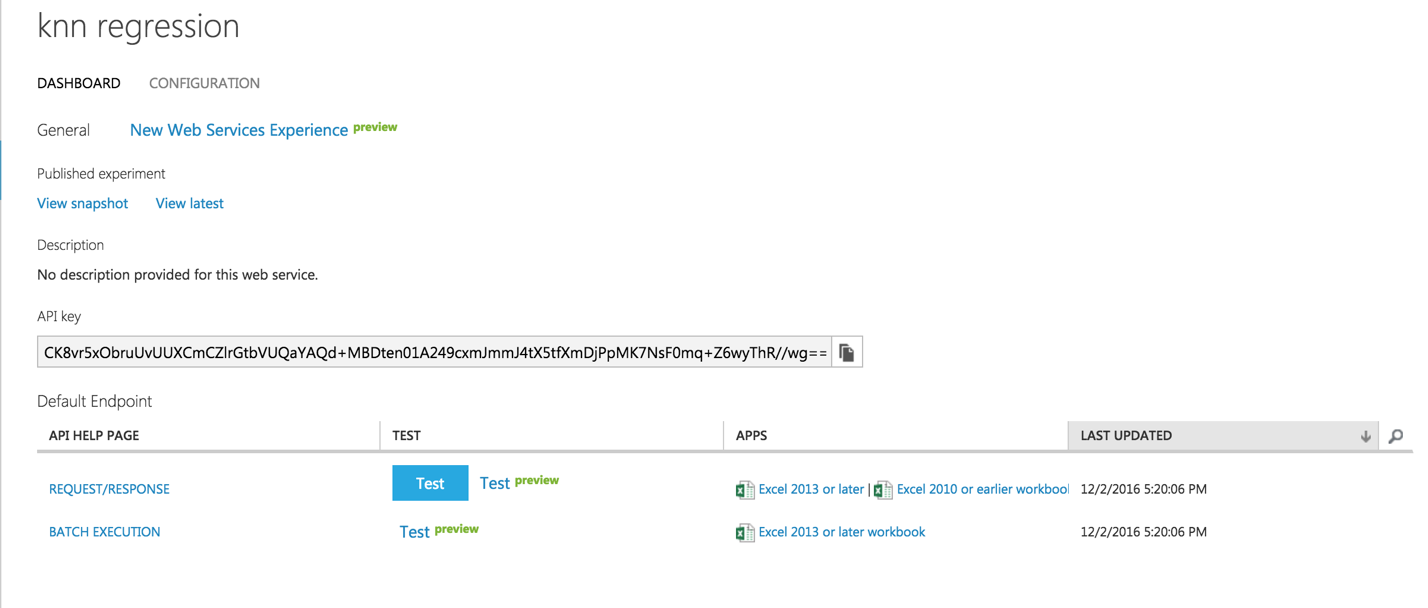
****

**Web Service**

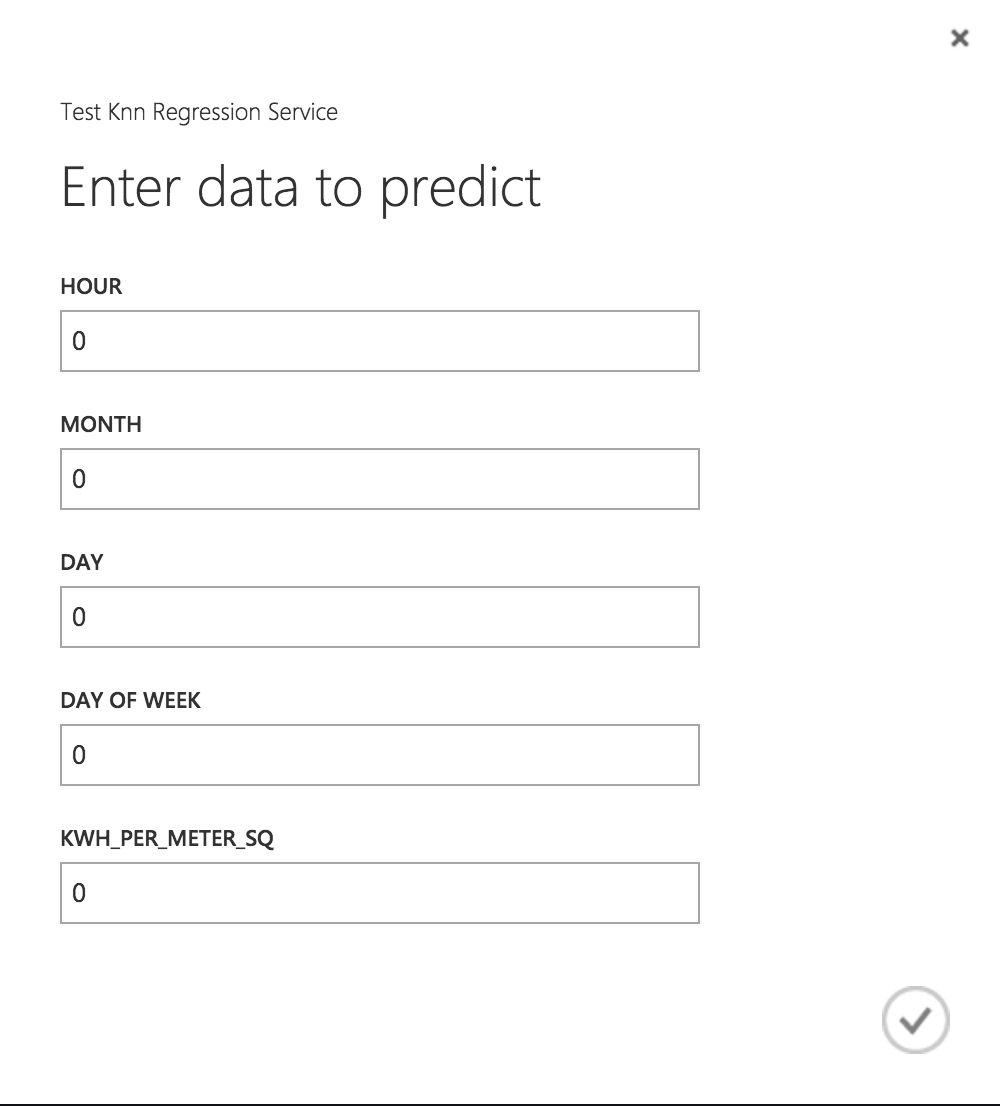
* Once the classification model is ready, we set up **Web Service**.
* The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**)
* Then we added the saved trained model back into the experiment.
* **Web service input** and **Web service output** modules are added.



* Now run the model and publish the web service



On running the web service, we get the following form which can be used to invoke the web service and do prediction

****

**Conclusion:** Thus from above it’s clear that the best model amongthe regressionmodels is Decision Forest Regression as it has the highest coefficient of determination i.e. 15%. Also, mean absolute error and root mean squared error is least of all.

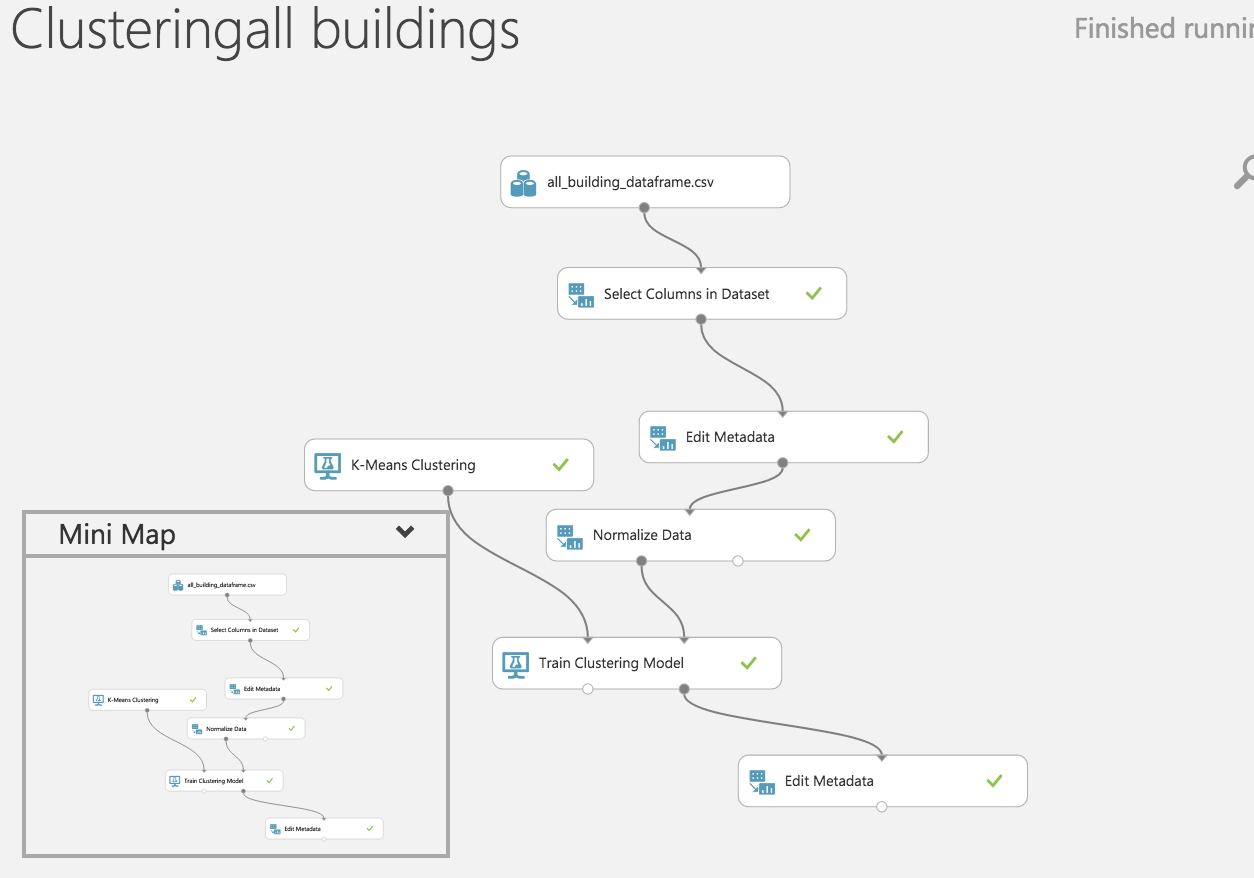
**Clustering**

Unsupervised learning methods are known as methods deal with finding patterns or classes of unlabeled data objects. In other word in such datasets there is no continuous or discrete responds associated with observations.

•Methods include of clustering (partitioning and hierarchical), P.C.A, association rules mining and Kernel density clustering.

**Azure models**

Below is the telecommunication network clustering model created.



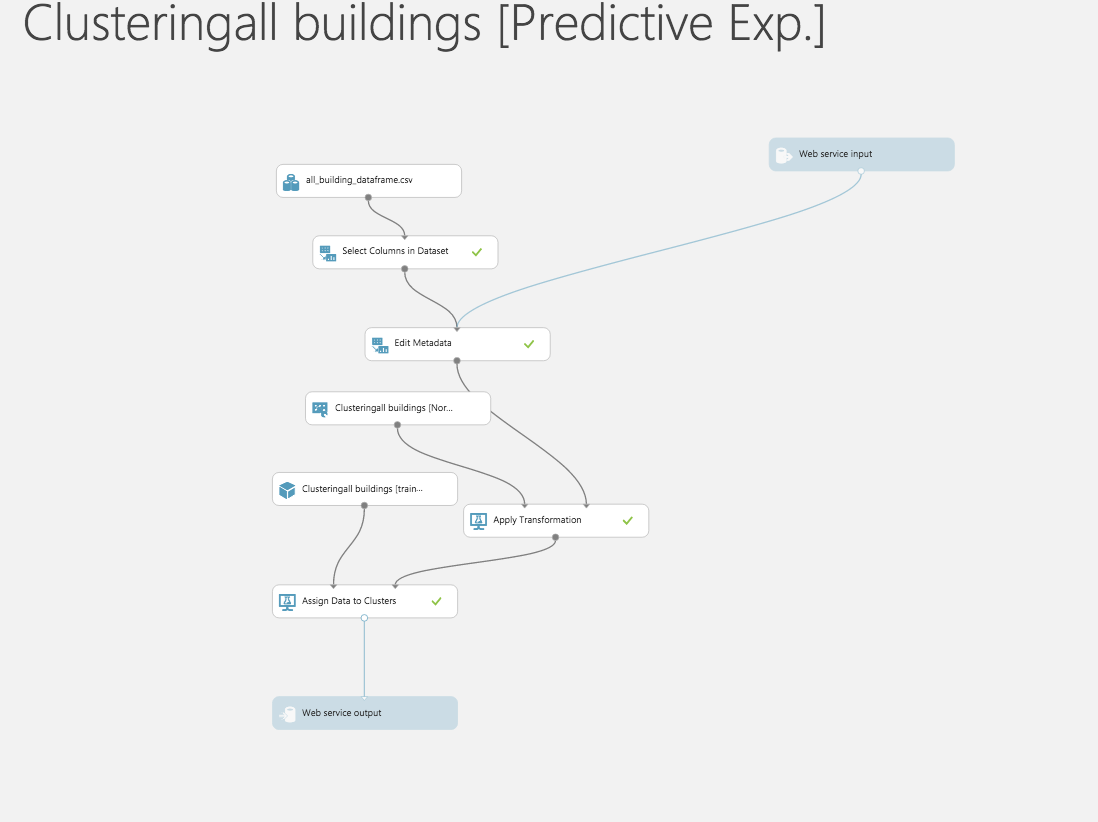
**K-means Clustering**

K-means is one of the simplest and the best known unsupervised learning algorithms, and can be used for a variety of machine learning tasks, such as detecting abnormal data, clustering of text documents, and analysis of a dataset prior to using other classification or regression methods.

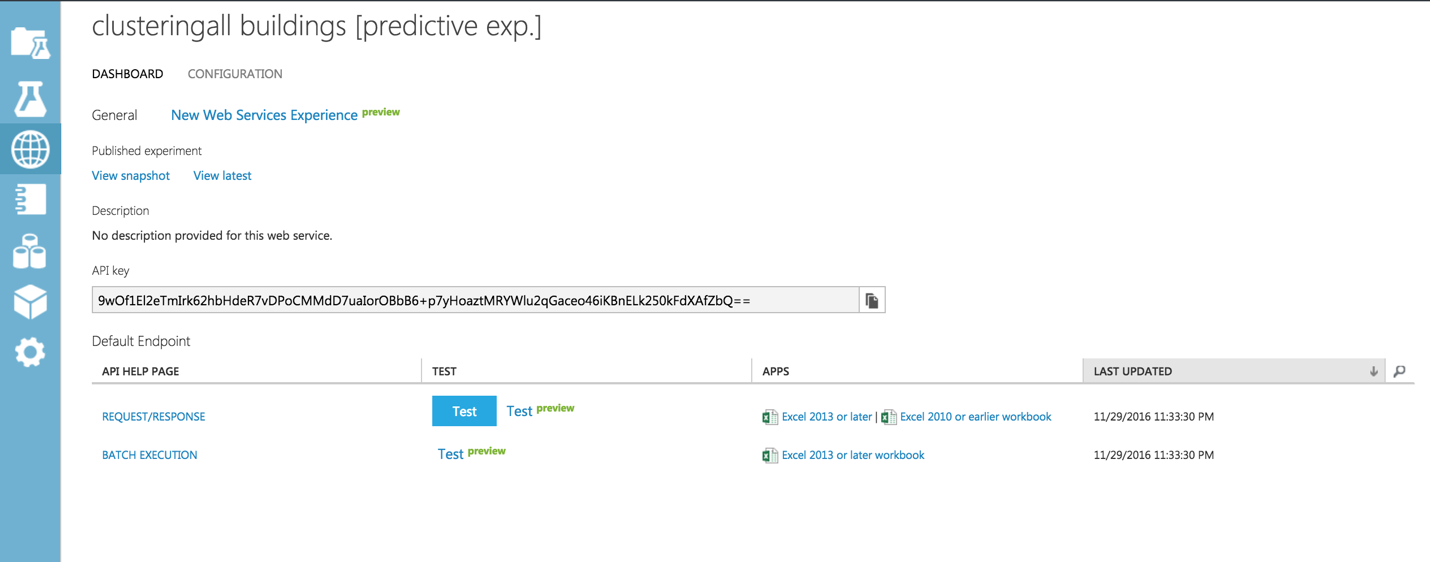
Because the K-means algorithm is an unsupervised learning method, the data you use to train the model does not need a label column. In other words, you don’t need to know any of the cluster categories in advance; the algorithm will find possible categories based solely on the data.

**Web Service**

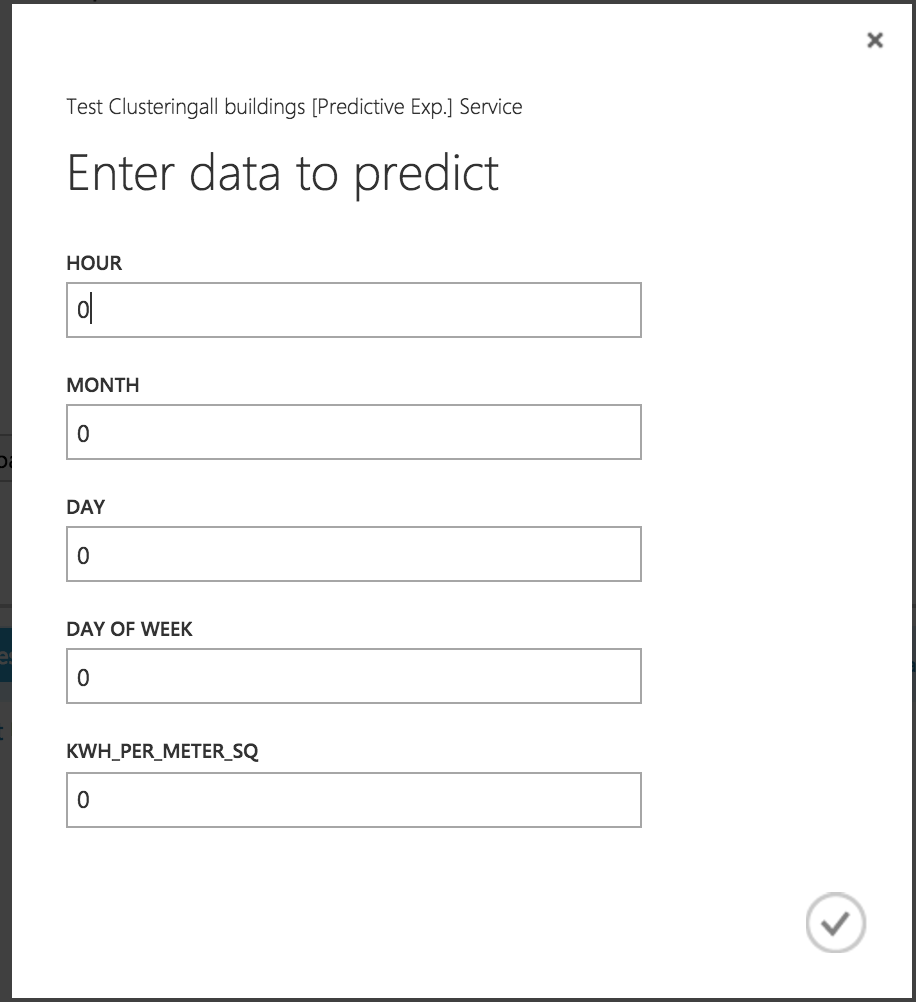
* Once the Clustering model is ready, we set up **Web Service**.
* The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**)
* Then we added the saved trained model back into the experiment.
* **Web service input** and **Web service output** modules are added.



* Now run the model and publish the web service



* On running the web service, we get the following form which can be used to invoke the web service and do prediction.



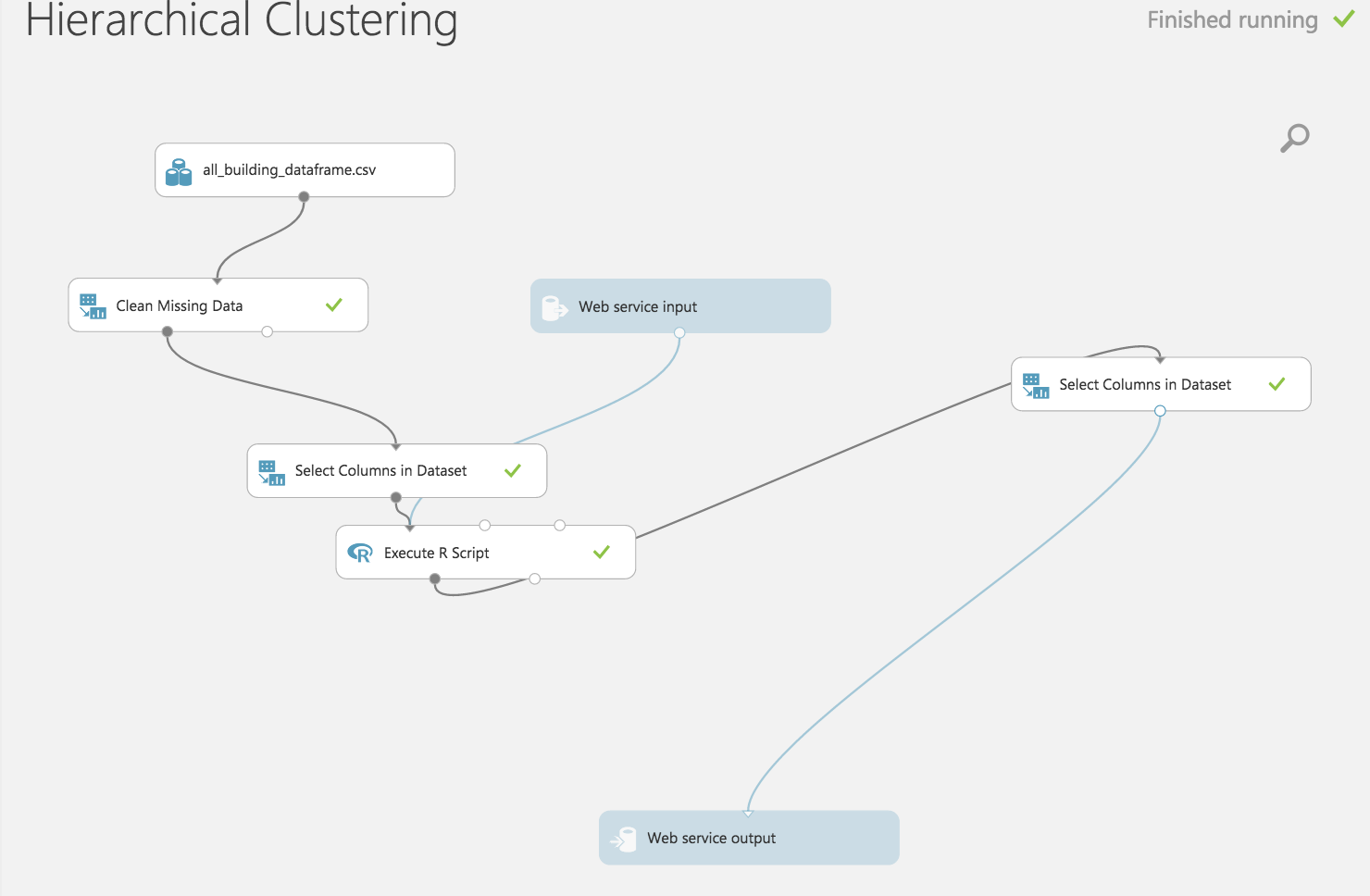
**Hierarchical Clustering**

We don’t need the to know about no of clusters like K-Means In hierarchical clustering it applies bottom up approach to create tree based visualization of clusters which is Dendrogram.

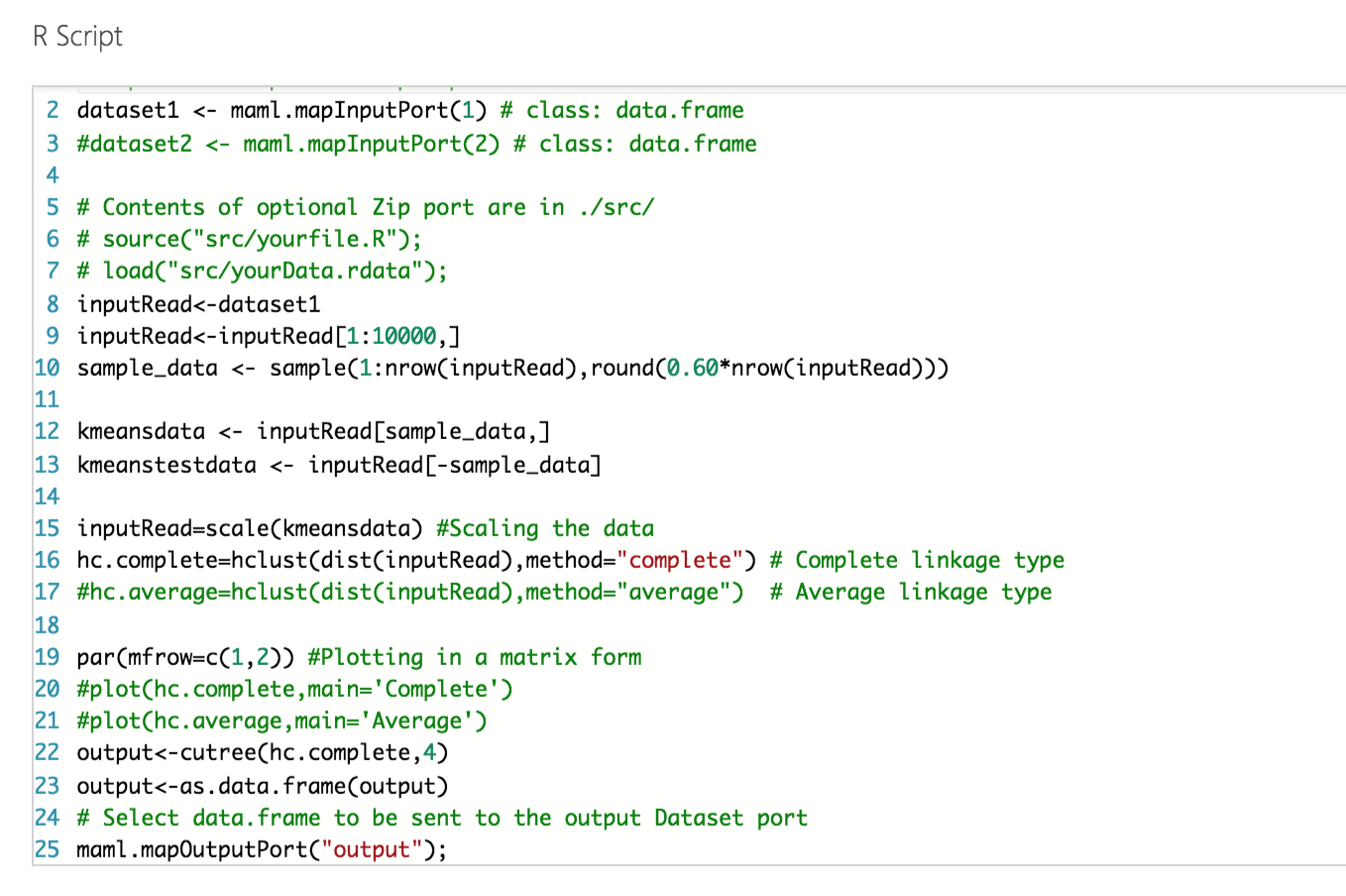
By cutting Dendrogram at any specific point we will have different no of clusters.

Linkage method can be complete, single, average and centroid while the distance can be Euclidean and correlation.

**Azure Model**

****

**R Script:**

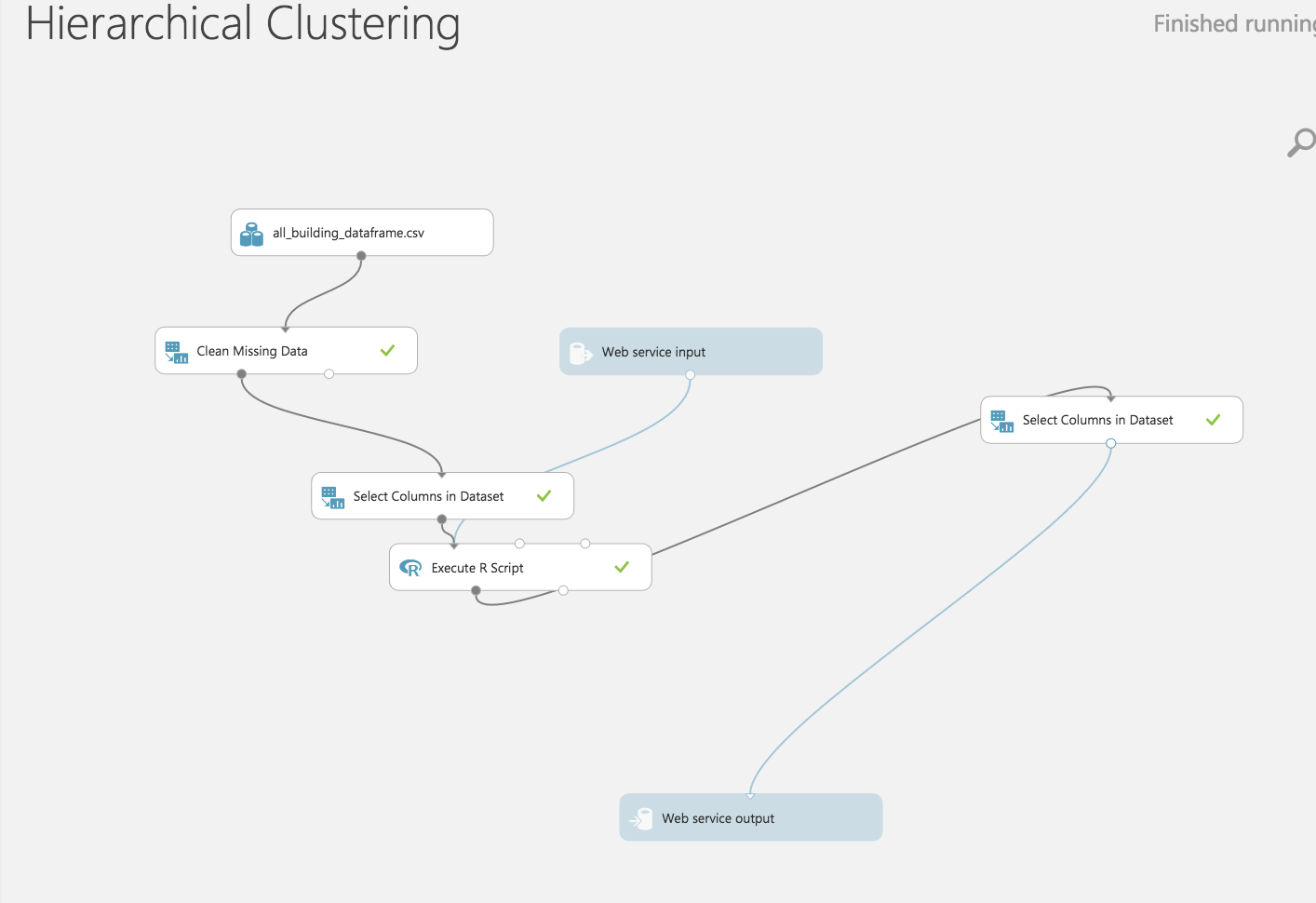
****

**Output:**

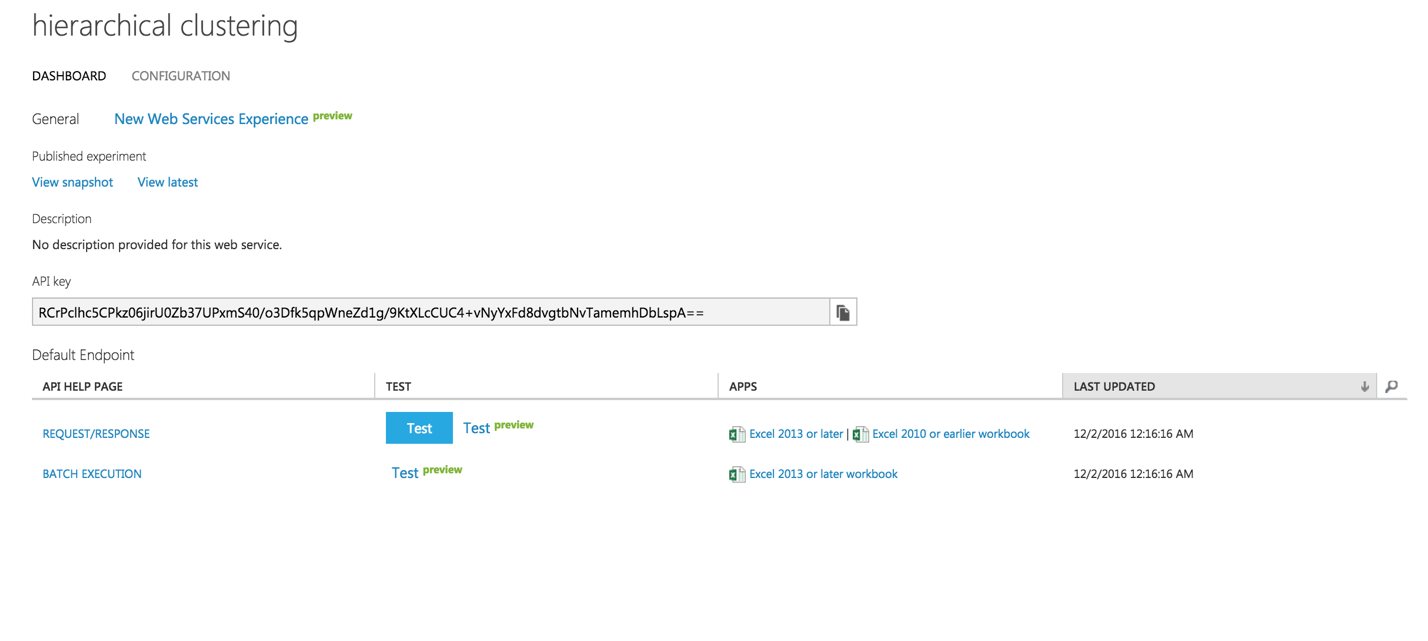
****

**Web Service**

* Once the Clustering model is ready, we set up **Web Service**.
* The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**)
* Then we added the saved trained model back into the experiment.
* **Web service input** and **Web service output** modules are added.

**

* Now run the model and publish the web service



* On running the web service, we get the following form which can be used to invoke the web service and do prediction.

