Prediction

Ran Four Prediction Model

Prediction of Loan Data

Selection of Columns using correlation and RFE

Deploy the Rest API using R-Shiny

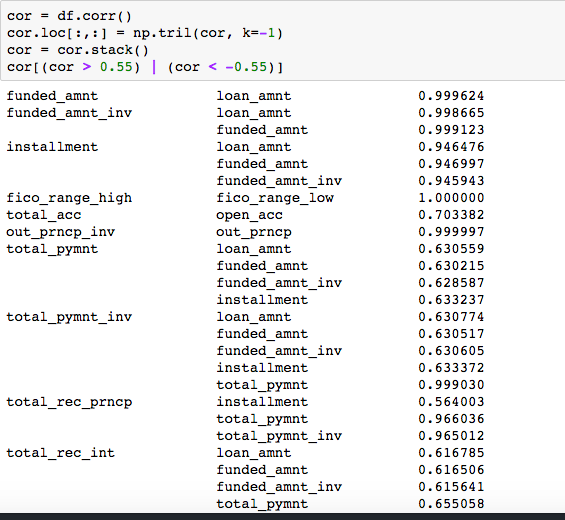
Ran the best model on Azure

Chose the best model

Once we have decided that we will give a loan and clusters are created both manually and using clustering algorithm, we will predict the loan’s interest rate. We have chosen 17 columns for the prediction based on correlation and RFE:

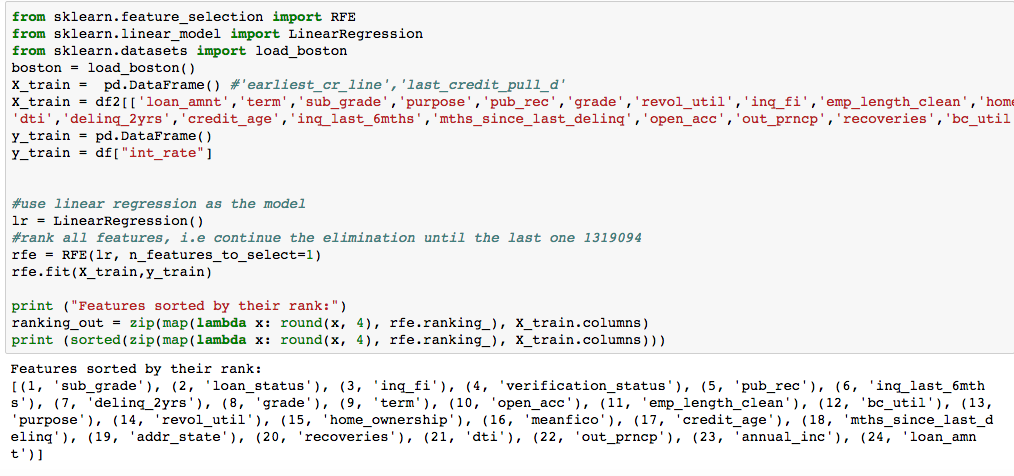
We found correlation between all the columns and columns with high correlations were dropped. We removed id, member id, title, desc and other irrelevant columns.

Correlation:



RFE:

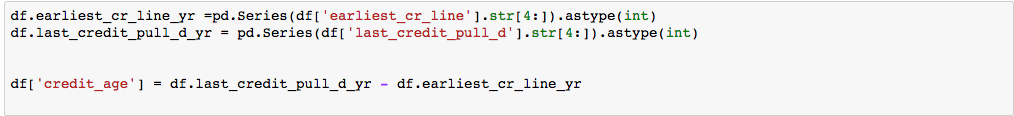
After correlation we calculated ranking of each column with respect to interest rate using RFE. RFE helps in identifying which parameter affects the most in predicting the interest rate.



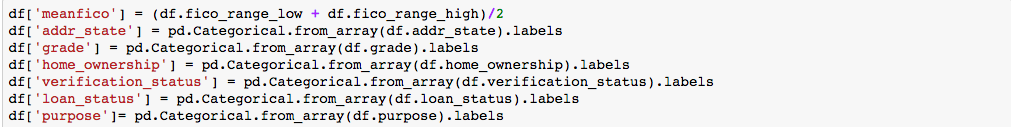
Derived Column:

Age of credit history reflects the length of your experience with the credit system.

Created Credit age by subtracting last credit pull and earliest credit line.



Created average of fico and labels for categorical columns as Regression models take only numeric data.

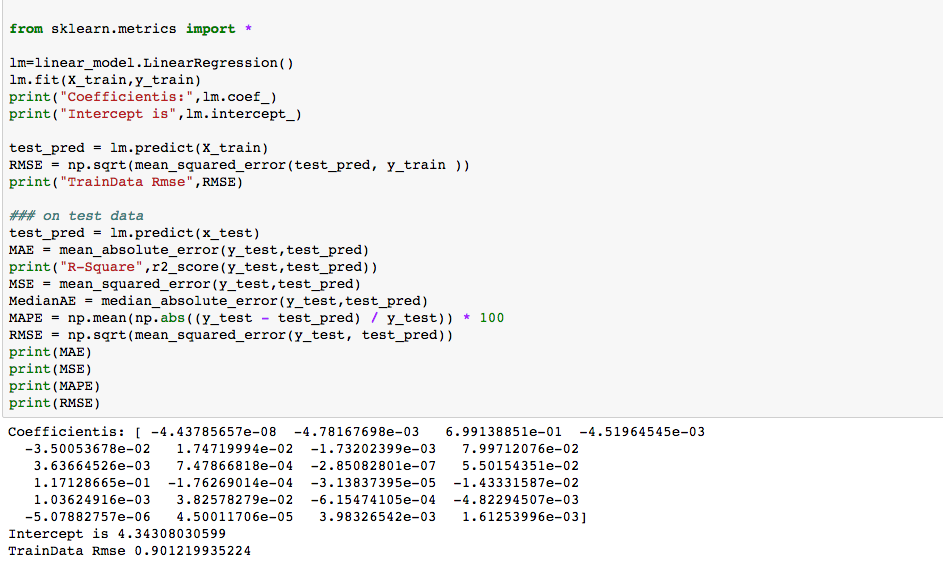


Screen%20Shot%202017-04-14%20at%201.53.12%20PM.png

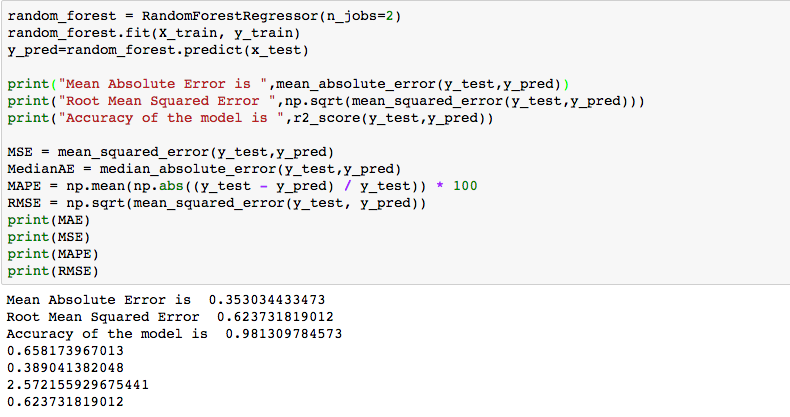
Selected Columns For Prediction:

* loan\_amnt: The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
* Term: The number of payments on the loan. Values are in months and can be either 36 or 60.
* sub\_grade: The number of payments on the loan. Values are in months and can be either 36 or 60.
* pub\_rec:
* home\_ownership: The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER
* annual\_inc: The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER
* verification\_status: Indicates if income was verified by LC, not verified, or if the income source was verified.
* Purpose: A category provided by the borrower for the loan request.
* addr\_state: The state provided by the borrower in the loan application
* dti: A ratio calculated using the borrower’s total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower’s self-reported monthly income.
* delinq\_2yrs: The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
* inq\_last\_6mths: The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
* meanfico: average fico of fico\_range\_high and fico\_range\_low
* emp\_length: Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
* Open\_acc: The number of open credit lines in the borrower's credit file.
* Revol\_util: Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.

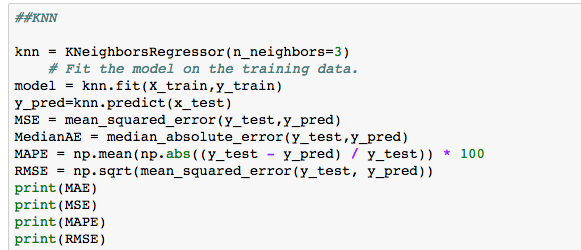
Linear Regression:



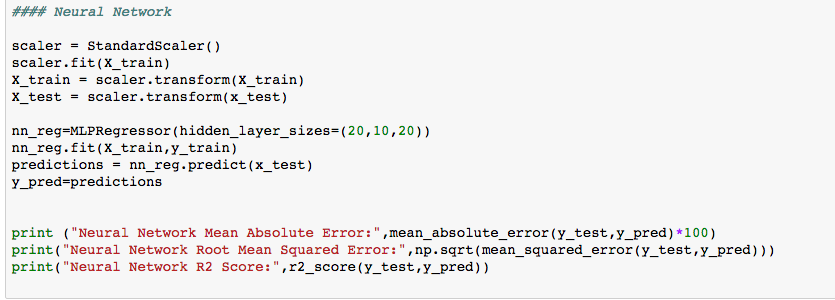
Random Forest:



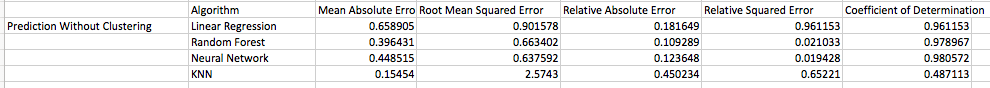
KNN:



Neural Network:



Matrix for comparison:



Since Neural Network’s RMSE and Coefficient of Determination was less compared to others we chose Neural Network as the best model.

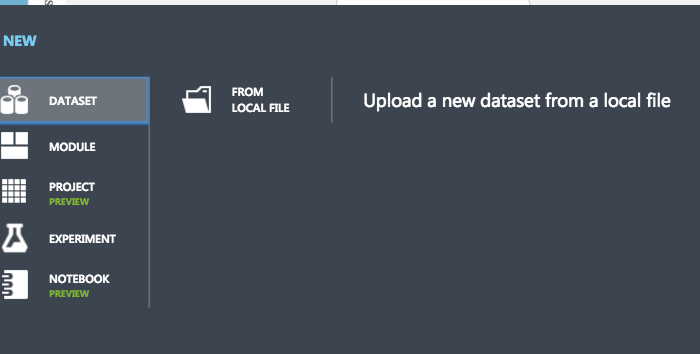
Prediction using Azure:

We created cluster for each grade (A, B, C,D,E,F,G) , clustering using K-means and no-cluster for prediction on Azure.

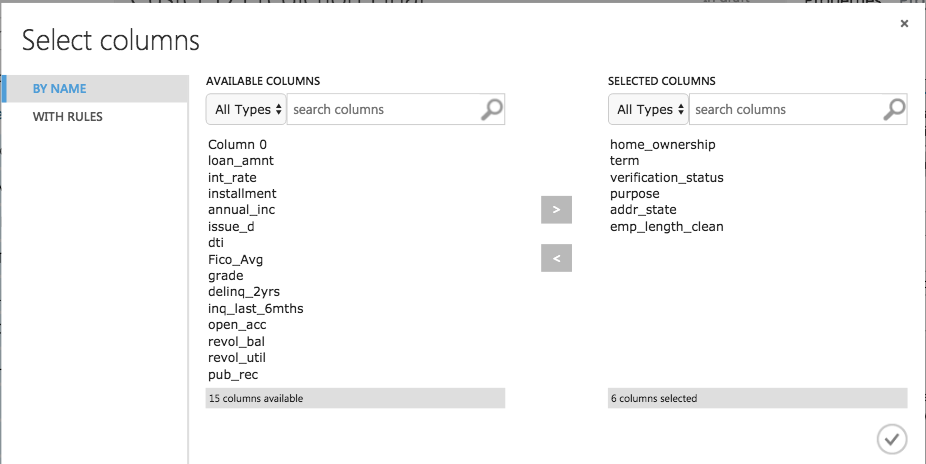
Prediction without cluster:

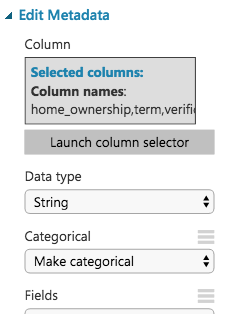
We saved the dataset with the columns that are required for the prediction in a csv file from jupyter python after feature selection.

Next, we imported the data from the local into the new experiment created.

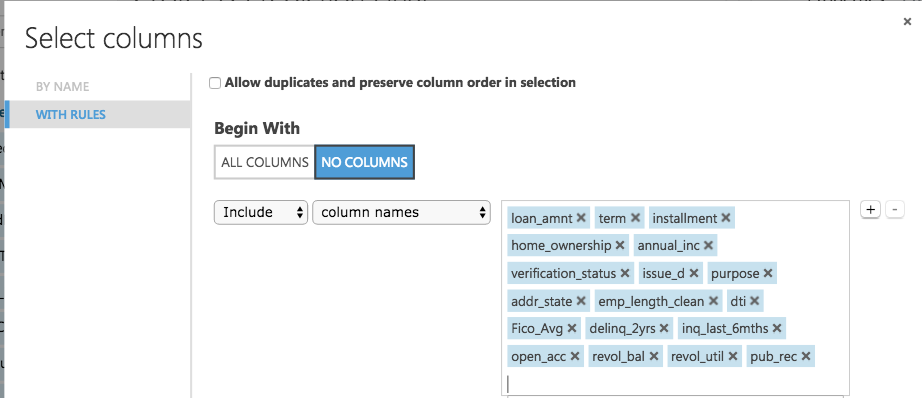


The next module converted string type columns to category type using edit metadata module.

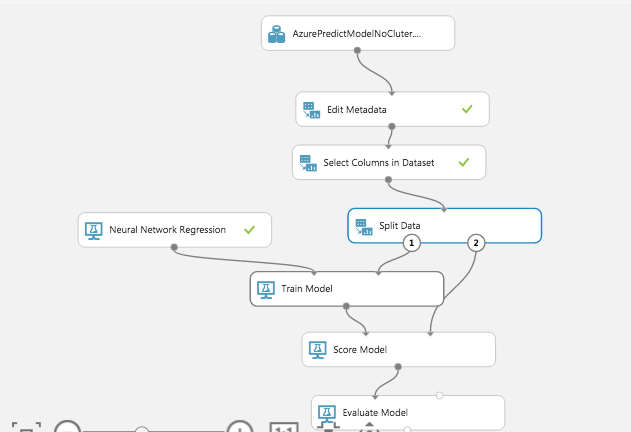




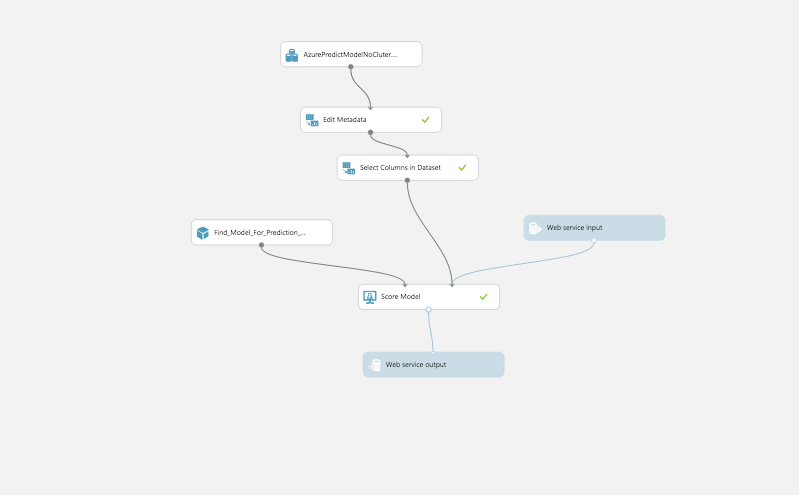
Next, we selected the columns we need to predict the interest rate using column in dataset module as below.



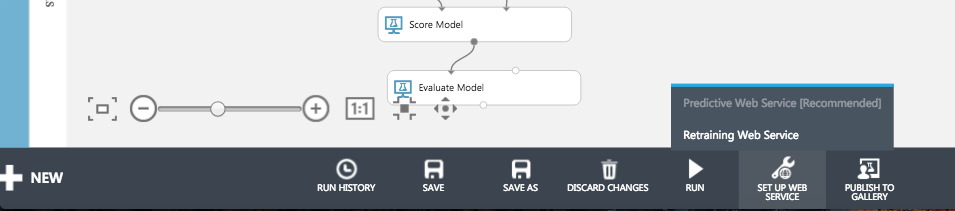
Then we split the data into 70-30 using the split module. The output of spilt goes to the train model module and to the score module. We chose Neural Network Regression module in the prediction and train model that the model as the input and trains the model. Once the model is trained its score and other parameters such as RMSE, RME can be viewed in score and evaluate modules.



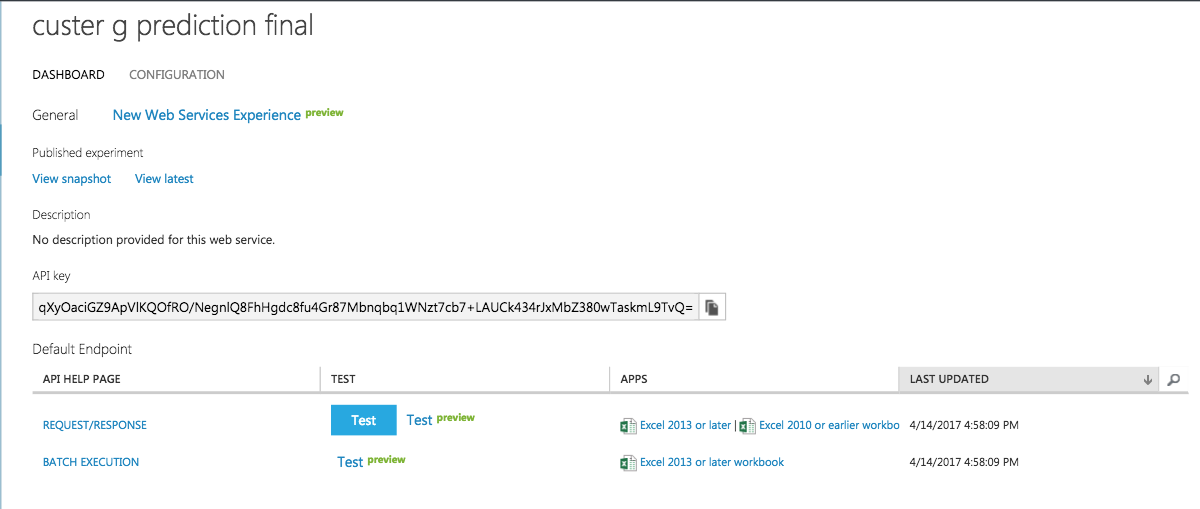
Once the model is trained we saved the model and then used that model on the test data to predict the interest rate as shown below.



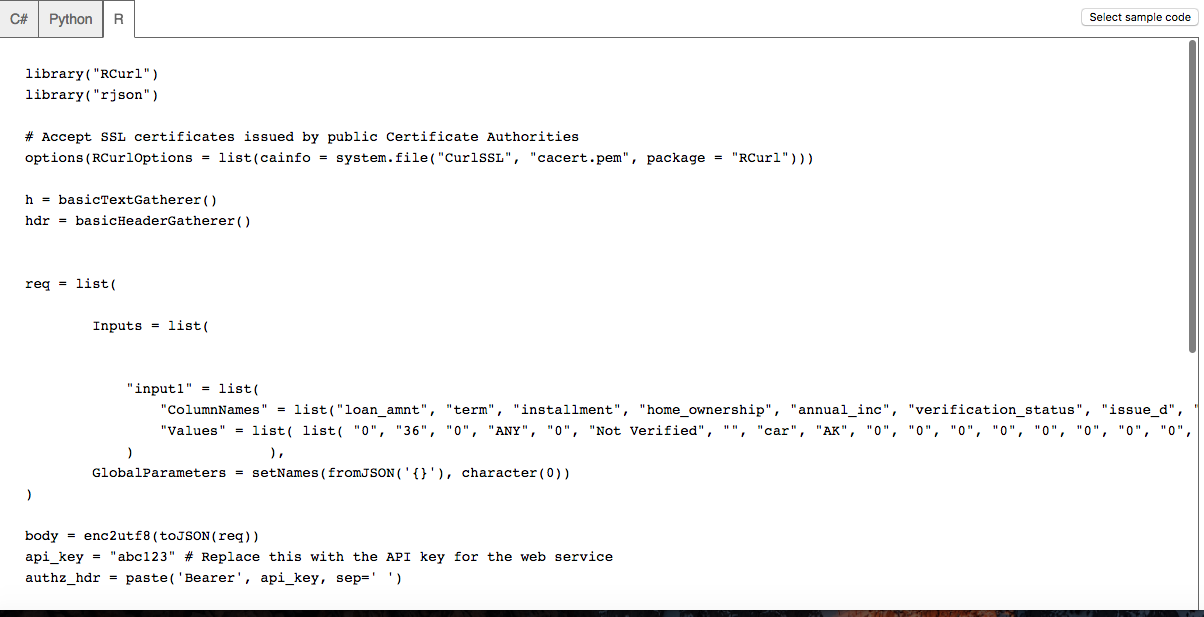
Once, the model runs successfully, we set it up as a web service as shown below.



Once it is deployed an API key will be generated.



We have used this API and the R code present in the Request/Repose link to create the rest API.



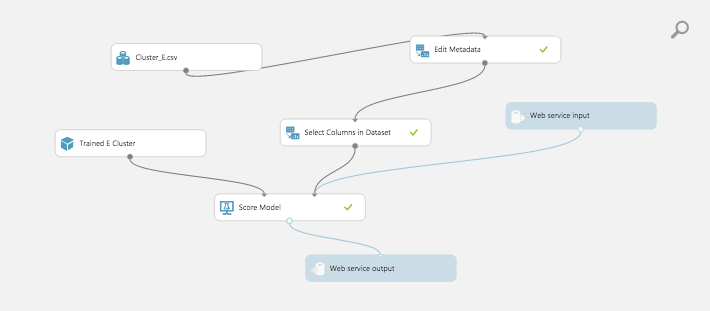
Manual Clustering for Lending Club Data:

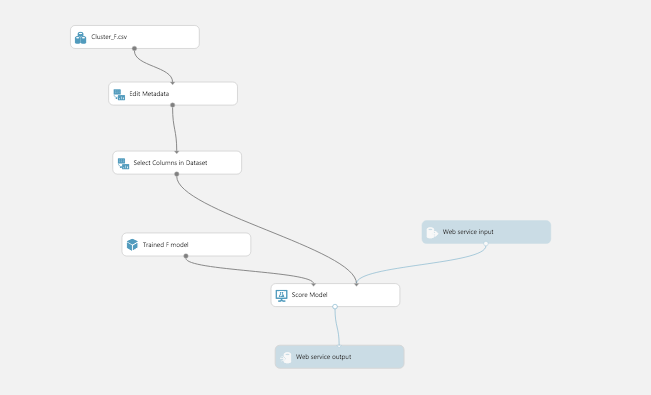
We have created 7 clusters for each grade (A – G) manually.

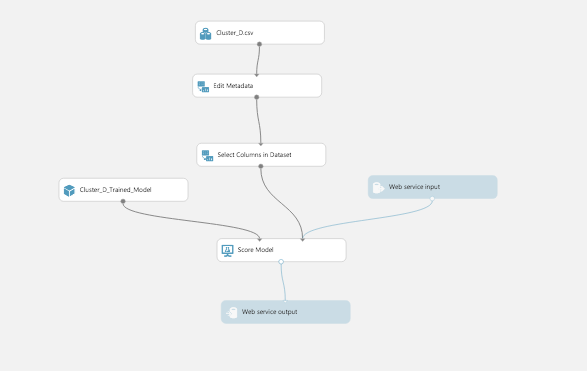
After feature selection we saved the dataset for each cluster separately in 7 csv files.

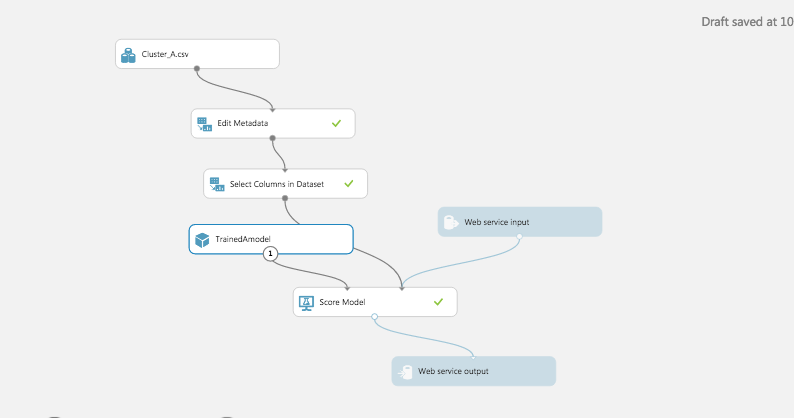
We used this as the input data to our prediction model in Azure. The files are imported into the experiment as mentioned previously. Once that is done we took each cluster individually and created a trained model of it. Once the trained model is saved we ran the model with the test data for prediction and set up the web service. With the help of the API key and the R sample code we created the rest API.

Below are the screenshots of how models are run in Azure.

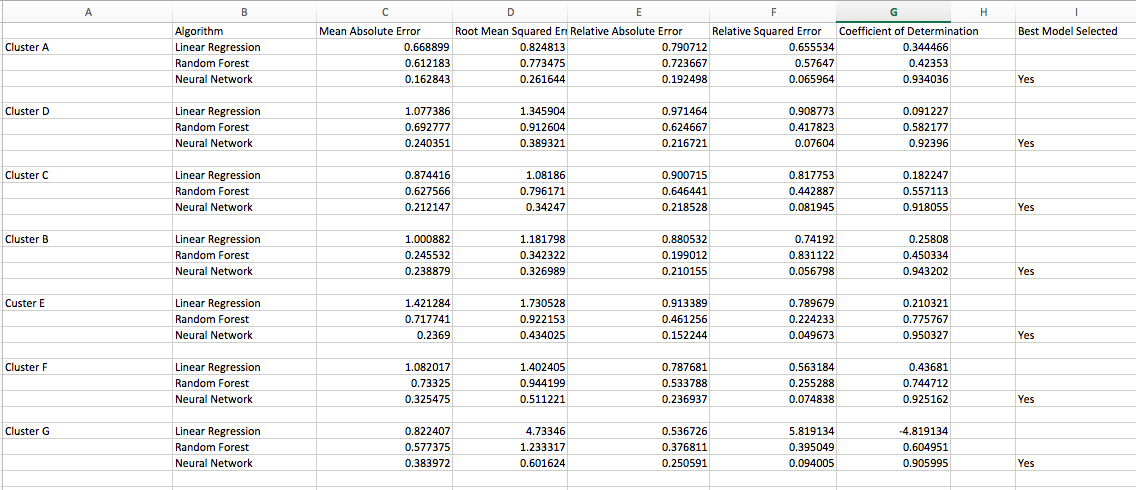








The matrix for each cluster is:



Based on Root Mean Squared Error and Co-efficient of Determination the best model for each cluster is Neural Network.

R-Shiny:

We chose R-Shiny for adhoc prediction of interest rate.

Shiny is a new package from RStudio that makes it incredibly easy to build interactive web applications with R.

**Features**

* Build useful web applications with only a few lines of code—no JavaScript required.
* Shiny applications are automatically “live” in the same way that spreadsheets are live. Outputs change instantly as users modify inputs, without requiring a reload of the browser.
* Shiny user interfaces can be built entirely using R, or can be written directly in HTML, CSS, and JavaScript for more flexibility.
* Pre-built output widgets for displaying plots, tables, and printed output of R objects.
* Fast bidirectional communication between the web browser and R using the websockets package.
* Uses a reactive programming model that eliminates messy event handling code, so you can focus on the code that really matters.

Sample Code in R-shiny

