Advances in Data sciences

Assignment 2:Part2

**Under the guidance of Professor Srikanth Krishnamurthy**

**Team 6:** Vaidehi Deshpande,

Puneeth Kumar,

Yamini Sehrawat

**Table of Contents**

[Abstract 3](#_Toc479977218)

[Part 2: Building and evaluating models 4](#_Toc479977219)

[Classification 6](#_Toc479977220)

[Logistic Regression: 6](#_Toc479977221)

[Random Forest: 8](#_Toc479977222)

[Neural Net Classifier 10](#_Toc479977223)

[SVC 12](#_Toc479977224)

[Feature Selection: 12](#_Toc479977225)

[Clustering 13](#_Toc479977226)

[Manual Division 13](#_Toc479977227)

[Clustering Algorithm 13](#_Toc479977228)

[Python Notebook 13](#_Toc479977229)

[Clustering Deployment on Azure 17](#_Toc479977230)

[Prediction 17](#_Toc479977231)

[Feature selection 18](#_Toc479977232)

[Prediction on Full dataset 19](#_Toc479977233)

[Prediction on Clusters – Manual and based on clustering algorithms 24](#_Toc479977234)

[Deployment on Azure 26](#_Toc479977235)

[Prediction with full dataset on Azure 26](#_Toc479977236)

[Prediction with clustering algorithm clusters on Azure 28](#_Toc479977237)

[Prediction using manual clusters in Azure 32](#_Toc479977238)

[Web Page 41](#_Toc479977239)

[GitHub Links: 43](#_Toc479977240)

**Lending Club Loan Data**

# Abstract

Dataset Link: <https://www.lendingclub.com/info/download-data.action>

These files contain complete loan data for all loans issued through the 2007-2015, including the current loan status (Current, Late, Fully Paid, etc.) and latest payment information. The file containing loan data through the "present" contains complete loan data for all loans issued through the previous completed calendar quarter. Additional features include credit scores, number of finance inquiries, address including zip codes, and state, and collections among others. The file is a matrix of about 890 thousand observations and 75 variables. A data dictionary is provided in a separate file.

Here, we are expected to build prediction models that will help us predict the interest rates based on various parameters users would input.

# Part 2: Building and evaluating models

Goal is to build a model to predict interest rates. We will get leads from people with different profiles and we must decide if we will give loans or not and if we will give a loan, how much interest we would charge for those loans.

**Classification**

* Use the “Loan Data” and the “Declined Loan Data” datasets to build classification models that will generate a flag whether to give a loan or not.
* Start with logistic regression using Jupyter and Python/R
* Compute ROC curve and Confusion matrices for training and testing datasets and comment on the results.
* Repeat this using Random Forest, Neural Network models and SVN algorithms.
* Choose one model you will deploy and implement this model on the Microsoft azure machine learning studio and create a REST API
* You should be able to a new record (You can define what features you will use) and the result will be a flag whether you would give a loan or not.

**Clustering**

Once we have decided to give a loan, we should build models to decide what interest rate to give. There are 3 possibilities:

1. Segment data into clusters (you define how many) manually using categorical or numerical features. For example, we can segment by state, by ownership of home, by average dti or a combination of features.

2. We use a clustering algorithm (that can factor both numerical and categorical variables) and segment data into k clusters. We will then build prediction models for each cluster.

3. No clusters; Just use data as is

Once we do the clustering use t-sne to visualize your clusters for some sample test data.

**Prediction**

Write a prediction script in a Jupyter notebook in R/Python that builds a Regression model for the interest rate using data from the 3 clustering methodologies you worked

* Try variable selection and build the best model for each segment/cluster (Note: You may have many segments and each model may have different coefficients based on the clusters used to train. You should automate it. Try parallel computing libraries to make things go faster).
* Compute MAE, RMS, MAPE for training and testing datasets o Repeat this using Random Forest, Neural Network models and KNN algorithms.
* Choose the best model amongst the 4 types of algorithms.
* Deploy the best algorithm/algorithms on Azure ML studio
* You will have a bunch of Rest APIs you should be able to choose from based on the cluster the record belongs to

**Deployment**

Design the following workflow:

* Given a record, use a pre-trained clustering model to cluster the record to a segment.
* You will have 3 cluster assignments (1-manual, 2-based on your clustering algorithm, 3-default 1cluster for all data)
* For each cluster, there should be a RestAPI which is linked to a chosen prediction model. Look up that API and use it to predict the 3 distinct interest rates.
* Select the highest interest rate and return it as your prescribed interest rate.

# Classification

Here we have first removed the rows with loan\_status ‘Does not meet credit policy’ and added them in Rejects dataset.



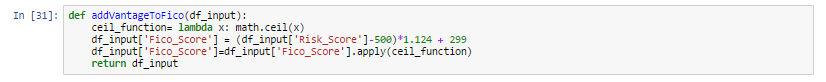


We have combined both accepted and rejects dataset and added flag to show if the loan is accepted or rejected:





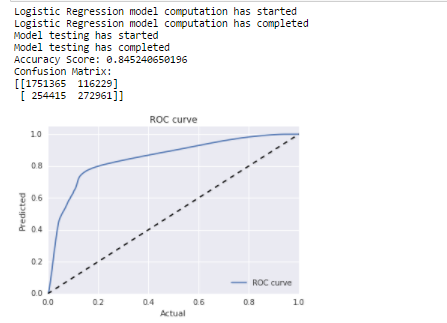
Using the below shown formula we have the set range of Risk Score in in Rejects dataset:

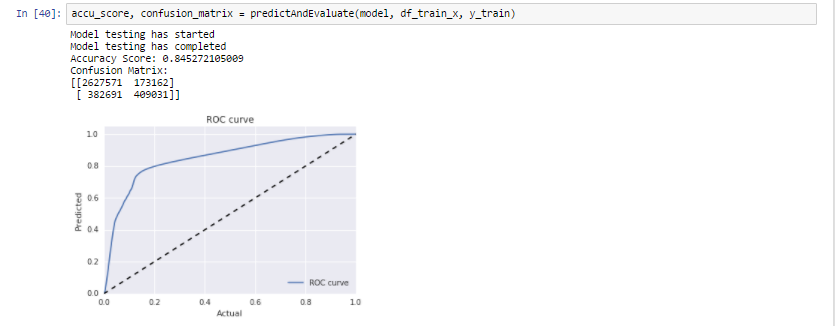


## Logistic Regression:

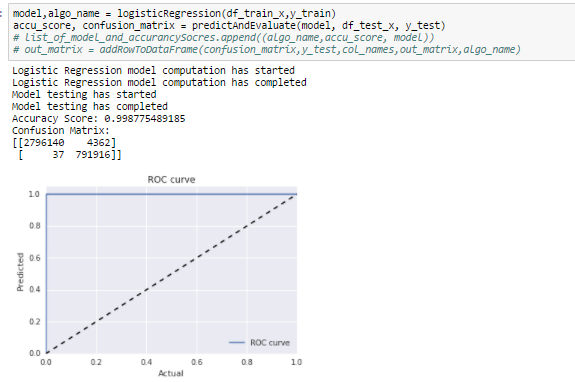


ROC curve for Logistic Regression





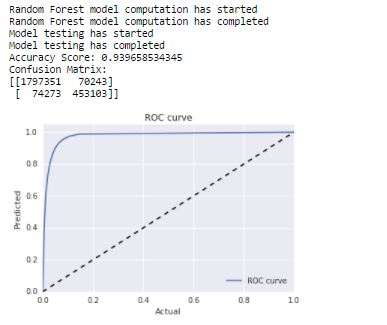
Good ROC curve

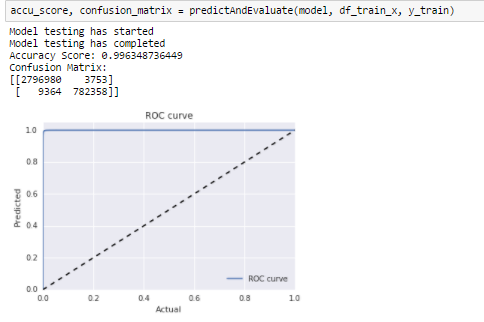


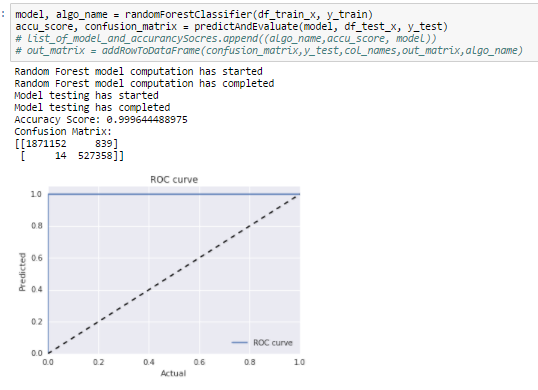
## Random Forest:



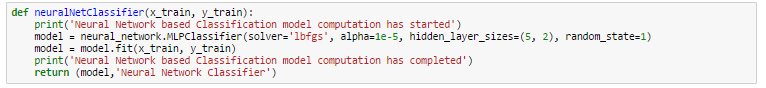
ROC curve for Random Forest

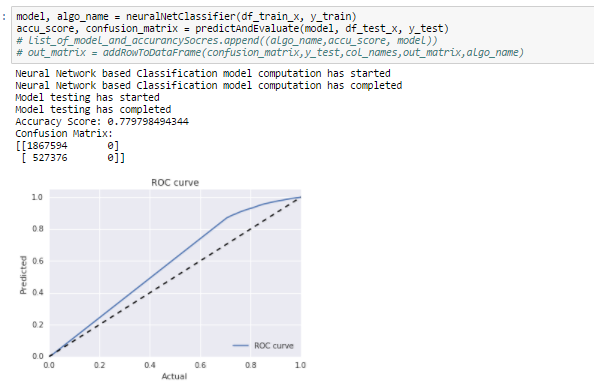


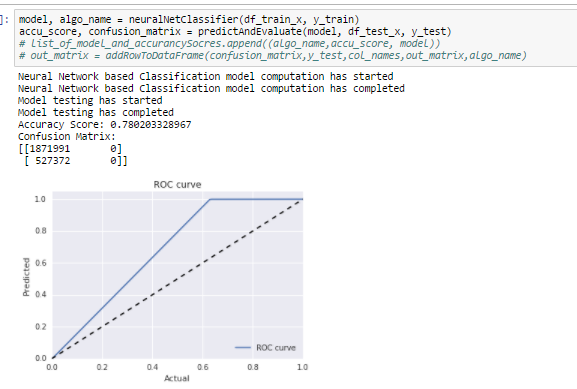




## Neural Net Classifier



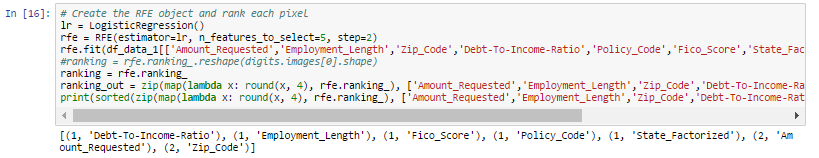




## SVC



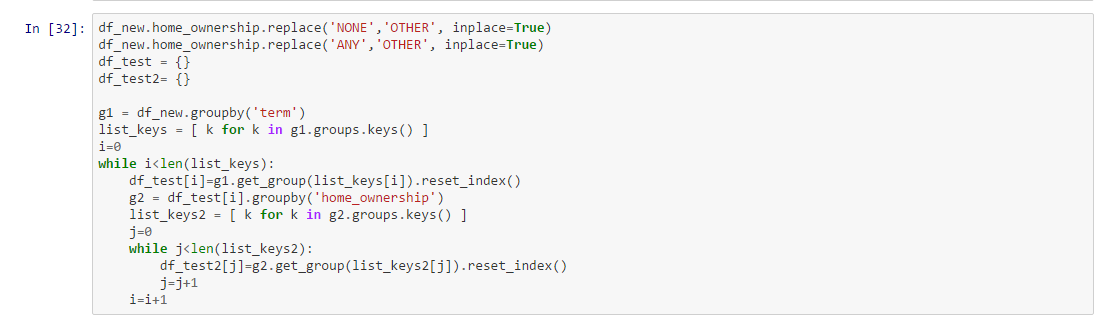
## Feature Selection:



# Clustering

## Manual Division

We have divided the accepted loan dataset manually into 8 bins using following logic:



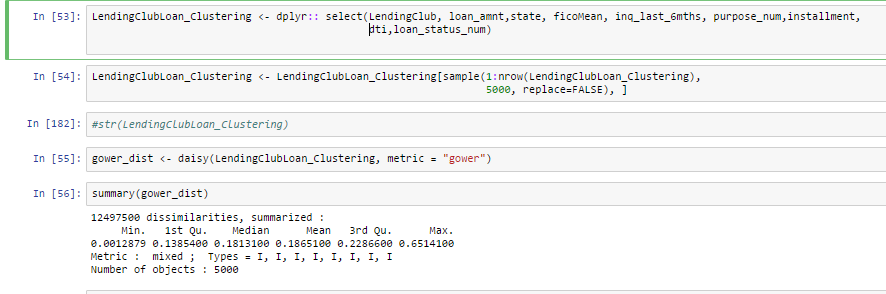
The bins are based on combinations of unique values in “Term” and “Home\_Ownership” columns. Since the percentage of data in “Home\_Ownership” of type “Any” and “None” is very less we have changed it to “Other”.

This is further sent to prediction algorithm for predicting individual interest rates for each bin.

## Clustering Algorithm

### Python Notebook

We are using Gower distance to calculate dissimilarities between the data points. Since we have a combination of continuous, binary and categorical variables we have chosen this method to calculate the distances.





We tried with various combinations of variables for getting the minimum mean gower distance and finally chose the following set for executing the clustering algorithm:

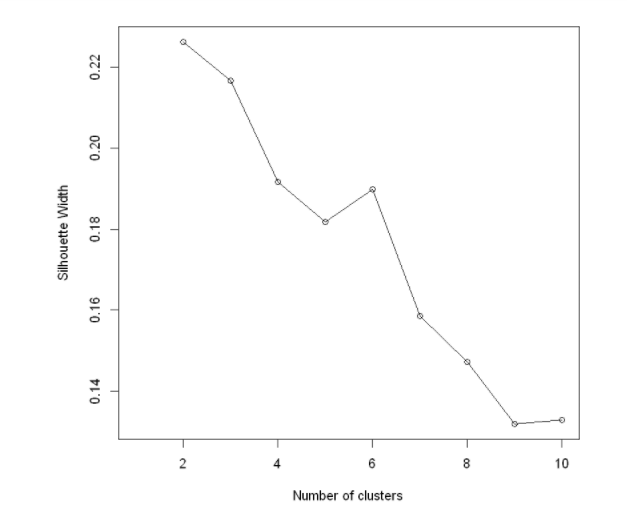
* Loan\_amnt
* State
* FicoMean
* Inq\_last\_6\_mths
* Purpose
* Installment
* Dti
* Loan\_status

With this set of variables we got Mean Gower distance = 0.1865100

Further we calculated the silhouette width using PAM for finding the perfect number of clusters to be chosen. This technique provides a succinct graphical representation of how well each object lies within its cluster. The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from -1 to 1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. Hence here we choose the number of clusters (k=2) where silhouette width is maximum using the following graph:

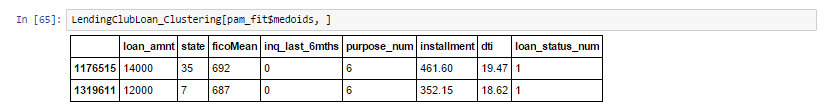




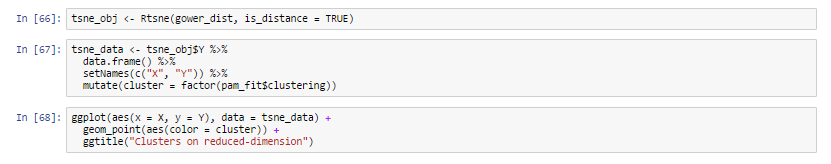


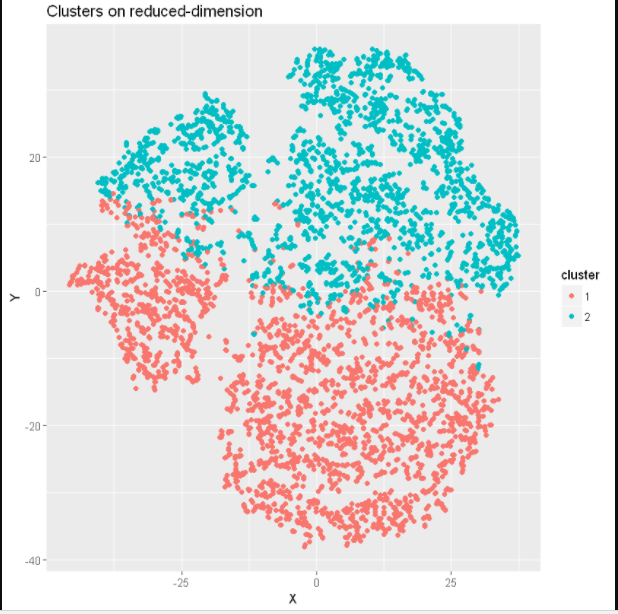
We further form the clusters using k=2 and PAM algorithm and plot them using t-sne which is used for dimensionality reduction:





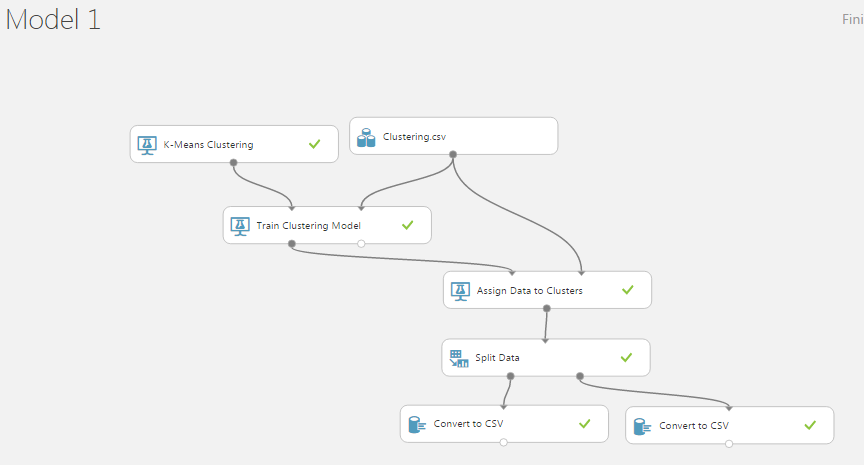
T-sne:





### Clustering Deployment on Azure

We implemented the clustering algorithm on azure with the best found set of variables and value of k.



The original dataset was thus split into two using the cluster assignments which was further used for prediction.

# Prediction

Remove extra columns not required for analysis: Following columns have been dropped as not required for prediction.

['earliest\_cr\_line\_month','last\_credit\_pull\_month','mo\_sin\_rcnt\_rev\_tl\_op','title','id',

'mths\_since\_recent\_bc','num\_accts\_ever\_120\_pd','num\_actv\_bc\_tl',

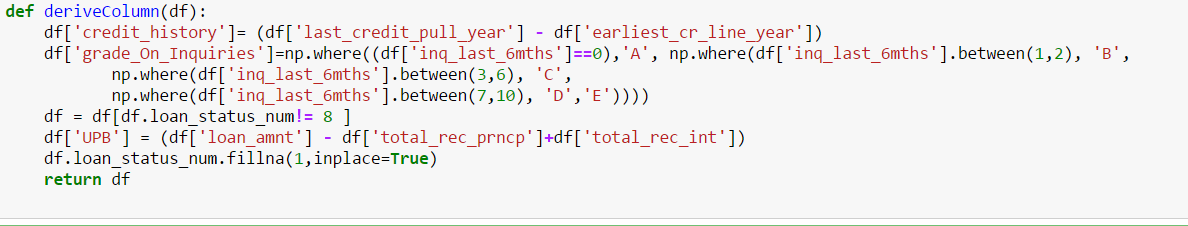
'num\_actv\_rev\_tl','num\_bc\_sats','num\_bc\_tl','num\_il\_tl','num\_op\_rev\_tl','num\_rev\_accts', 'num\_rev\_tl\_bal\_gt\_0','num\_tl\_120dpd\_2m','num\_tl\_30dpd','num\_tl\_90g\_dpd\_24m','num\_tl\_op\_past\_12m',

'pct\_tl\_nvr\_dlq','emp\_title','mo\_sin\_rcnt\_tl','id','percent\_bc\_gt\_75',

'total\_rec\_late\_fee','zip\_code']

Derive following columns which will add to prediction:

1. Credit\_history
2. grade\_On\_Inquiries
3. UPB



## Feature selection

**Variable Selection using RFE:**

We have used Recursive feature selection in Python to obtain variables based on their ranking

The Recursive Feature Elimination (or RFE) works by recursively removing attributes and building a model on those attributes that remain.

It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute.

And the result is:

lmReg = LinearRegression()

rfe = RFE(estimator=LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=1, normalize=False),

n\_features\_to\_select=12, step=1, verbose=0)

rfe.fit(df\_X,df\_y)

ranking = rfe.ranking\_

We got following variables with their ranks:

print(sorted(zip(map(lambda x: round(x, 2), rfe.ranking\_), df\_X)))

[(1, 'credit\_history'), (1, 'emp\_length'), (1, 'ficoMean'), (1, 'home\_ownership\_num'), (1, 'inq\_last\_6mths'), (1, 'installment'), (1, 'loan\_amnt'), (1, 'loan\_status\_num'), (1, 'open\_acc'), (1, 'purpose\_num'), (1, 'revol\_util'), (1, 'term'), (2, 'addr\_state'), (3, 'annual\_inc')]

In [114]:

Prediction on Full dataset:

Use the full dataset – Lending club accepted loan dataset.

-----Jupyter Script----

Split the data randomly into Training and Testing dataset:



**------------Below algorithms are run on full datasets-------------**

**Regression**

The regression model is first trained on a cleaned dataset, then run the model on complete dataset. On fitting the regression model, testing against the complete test dataset measures of predictive accuracy are:

lm.coef\_, lm.intercept\_, R-Squared

Linear Regression model computation starts

Coefficient is: [ 4.62297842e-01 -3.16846012e-02 -1.36464427e-03 4.47008055e-02

-2.09305350e-01 -3.79871032e-06 -1.09434443e-02 1.89022022e-01

-1.94375285e-02 -3.18253477e-02 1.16685630e-02 6.31845010e-01

2.28157726e-04 9.39023705e-01]

Intercept is: 14.5688203328

R-Square Training 0.614847567625

R-Square Testing 0.615626946082

Completed Regression

Model Prediction and evaluation metrics:

**AcceptedLoan\_Model MAE\_test MAE\_train RMSE\_test MSE\_test MSE\_train**

Regression 2.201350 2.204143 2.834695 8.035495 8.057798

**Ordinary Least Squares Assumptions**

OLS measures the accuracy of a linear regression model.

OLS is built on assumptions which, if held, indicate the model may be the correct lens through which to interpret our data. If the assumptions don't hold, our model's conclusions lose their validity.

We build our model on testing various variables and following features gives suitable results.

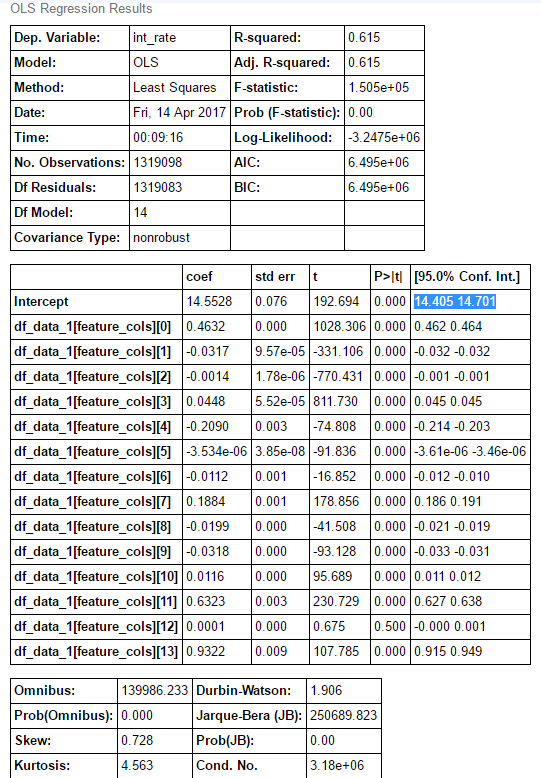
ols\_model = ols("int\_rate ~df[feature\_cols] ", data=df\_data\_1).fit()

From the results below we can see **Adj. R-squared** is 61%.

The **regression coefficient (coef)** represents the change in the dependent variable resulting from a one unit change in the predictor variable, all other variables being held constant.

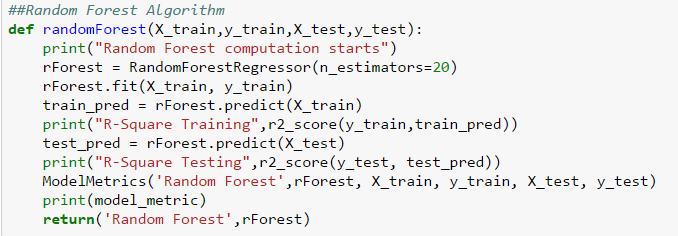
The **standard error** measures the accuracy of the variable’s coefficient by estimating the variation of the coefficient if the same test were run on a different sample of our population. Our standard error is low in case of variables and therefore appears accurate.

The **confidence interval** is a range within which our coefficient is likely to fall. The coefficients we can see will be within our confidence interval, [14.405 14.701]



**Random Forest**

The model is built on following variables as:



And the metrics are:

Random Forest computation starts

R-Square Training 0.989169716933

R-Square Testing 0.935300506296

Completed Random Forest

**AcceptedLoan\_Model MAE\_test MAE\_train RMSE\_test MSE\_test MSE\_train**

Random Forest 0.540625 0.217846 1.163002 1.352573 0.226581

We calculated the feature importance for the features we selected:

**feature importance**

13 loan\_status\_num 0.002237

4 home\_ownership\_num 0.002242

6 emp\_length 0.005063

12 addr\_state 0.008302

8 open\_acc 0.008909

9 credit\_history 0.011561

10 revol\_util 0.019524

11 inq\_last\_6mths 0.024004

7 purpose\_num 0.028539

5 annual\_inc 0.036976

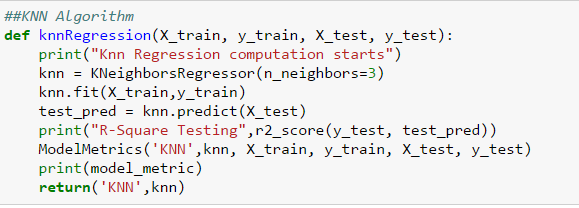
1 ficoMean 0.154819

0 term 0.169335

2 loan\_amnt 0.224339

3 installment 0.304150

**KNN**



Knn Regression computation starts

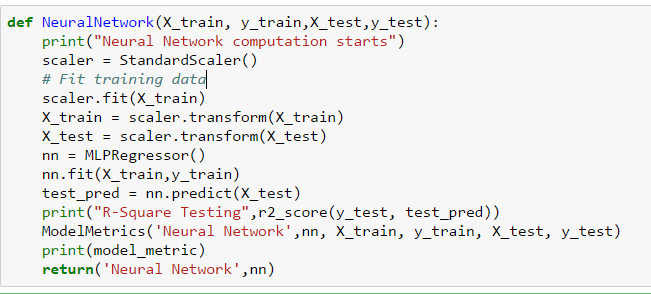
R-Square Testing 0.624362133807

Completed KNN

**AcceptedLoan\_Model MAE\_test MAE\_train RMSE\_test MSE\_test MSE\_train**

KNN 1.672386 1.137260 2.802299 7.852882 3.667681

**Neural Network**



Neural Network computation starts

R-Square Testing 0.511358041902

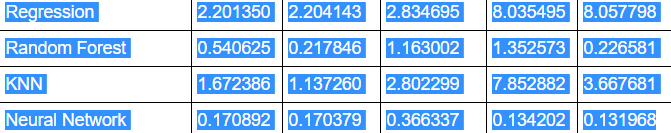
Completed Neural Network

**AcceptedLoan\_Model MAE\_test MAE\_train RMSE\_test MSE\_test MSE\_train**

Neural Network 0.170892 0.170379 0.366337 0.134202 0.131968

**Final Metrics:**

**Model MAE\_Test MAE\_train RMSE\_Test MSE\_Test MSE\_Train**



Choosing the Best Model, we are considering the RMSE to select the best mode

Though, Neural Network comes out to be the best model after comparing the RMSE, but we will go with Random Forest which is very near to Neural.

As with Neural it requires high computation burden and often hard to train.

## Prediction on Clusters – Manual and based on clustering algorithms

To parallize the tasks we have used Luigi to run regression algorithms for all the clusters created:

CSV files belonging to each cluster are place in bin folder.

Below are the tasks:

1. LinearRegressionTask
2. RandomForestTask
3. NeuralNetworkTask
4. SummarizationTask
5. KNNTask
6. FinalTriggerTask

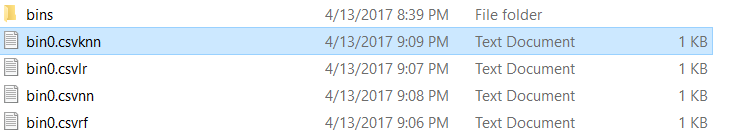


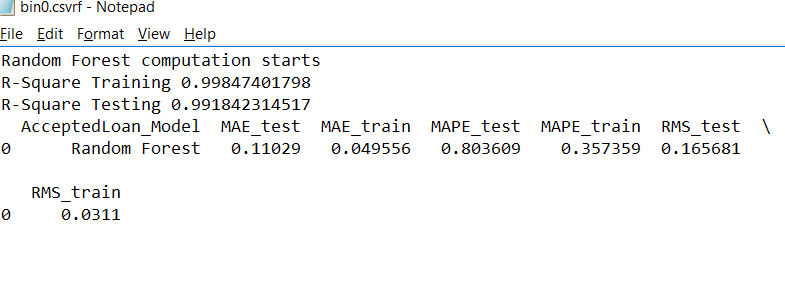
FinalTriggerTask will call each task one by one, executing all the algorithms on each cluster file and storing the results for each algorithm in TEXT files. We have created separate TEXT file for each algorithms result.

For example, following files are created for first segment/cluster:

Term:60

Home\_ownership: MORTGAGE



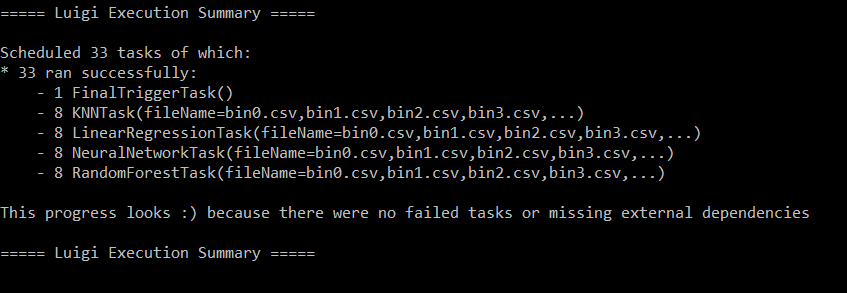


======================================================================

Run the script:

C:\Users\Yamini\AppData\Local\Microsoft\Windows\INetCacheContent.Word\run.png

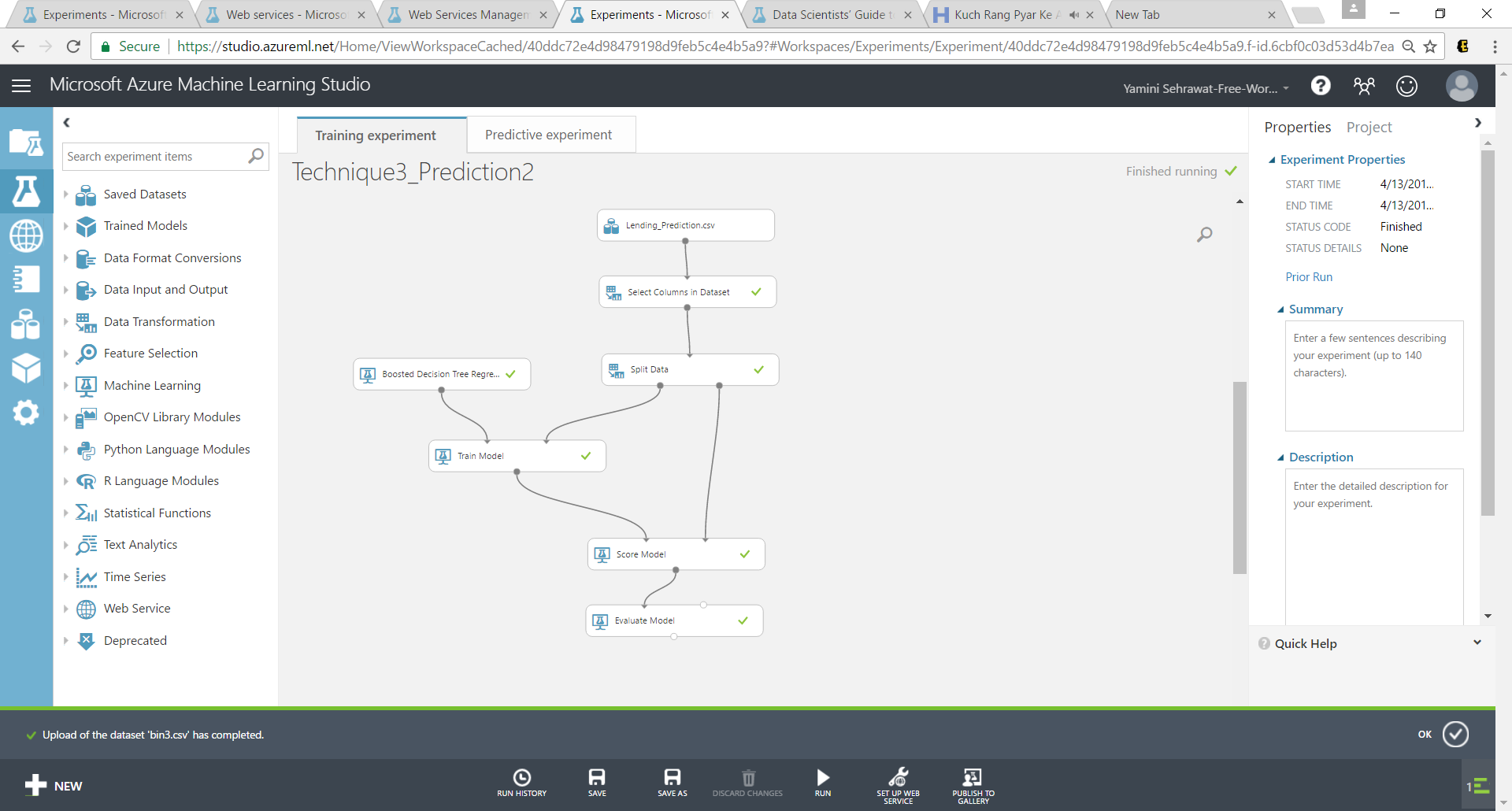
Success Result

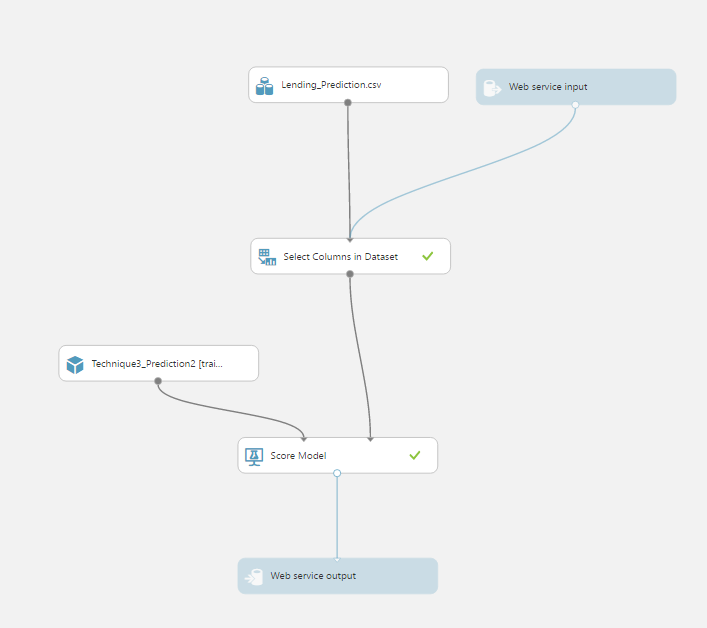


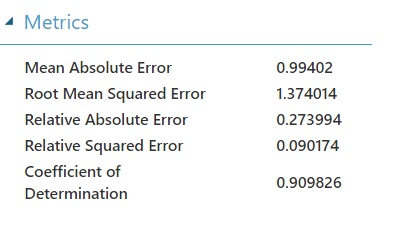
## Deployment on Azure

### Prediction with full dataset on Azure

1. No cluster, Full dataset







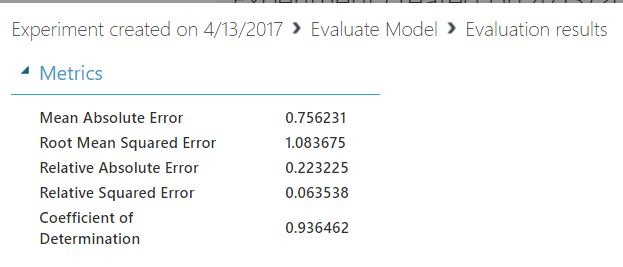
Then deployed the web service.

##### Results

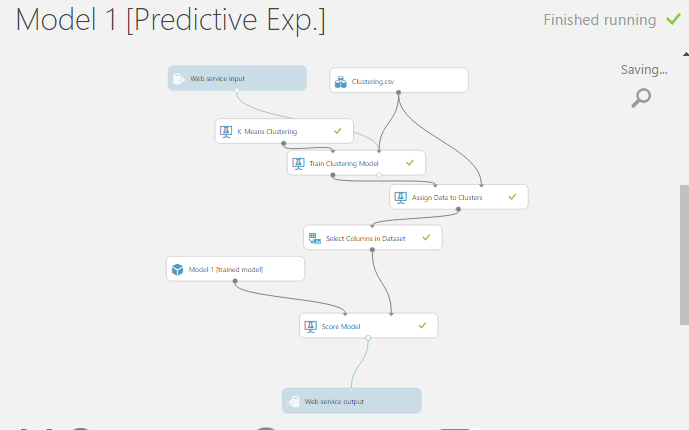
|  |  |
| --- | --- |
| **RMSE** | **R square** |
| **1.374014** | **0.909826** |

### Prediction with clustering algorithm clusters on Azure

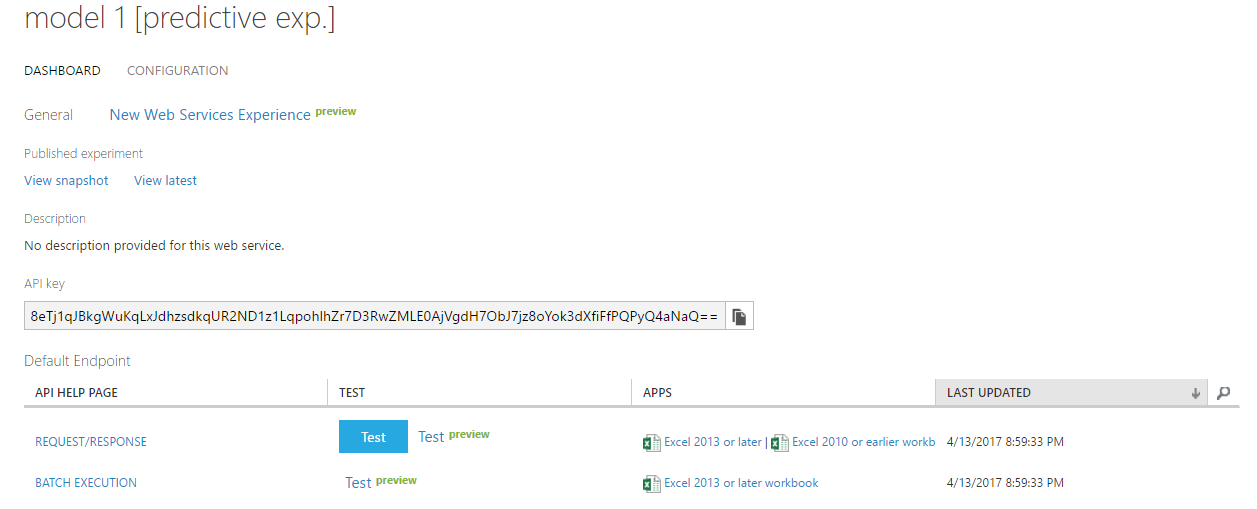
##### Output of cluster 1



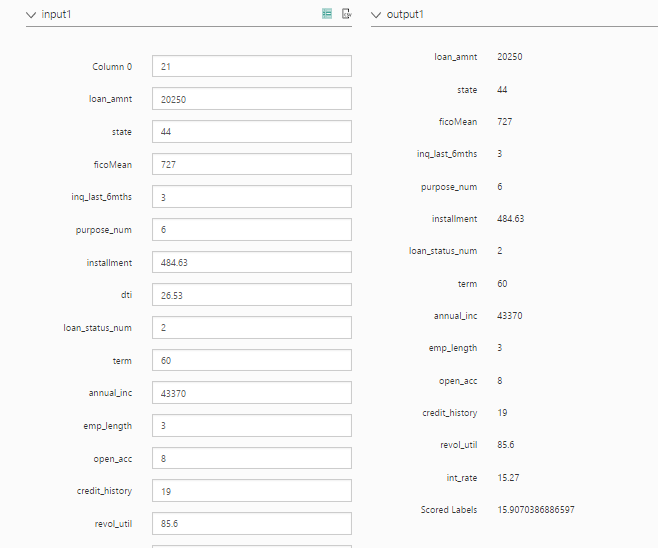
Web service:



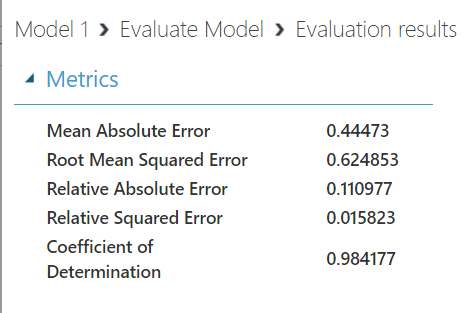
API generated:



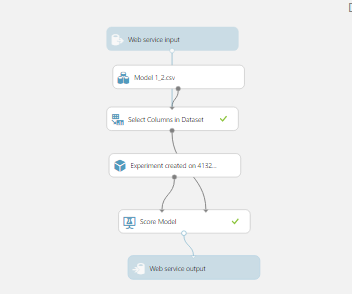
Deploy web service and generate API:



##### Output of cluster 2



Deploy Web Service and generate API:



##### Results:

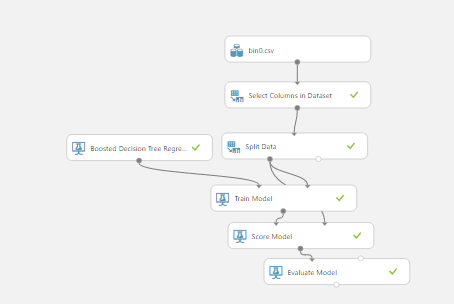
|  |  |  |
| --- | --- | --- |
| **Cluster Number** | **RMSE** | **R square** |
| **1** | **1.083675** | **0.936462** |
| **2** | **0.624853** | **0.984177** |

### Prediction using manual clusters in Azure

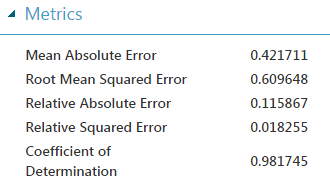
We ran the prediction algorithms on all the bins created by manual division of dataset generating interest rate for each bin.

##### Bin 0

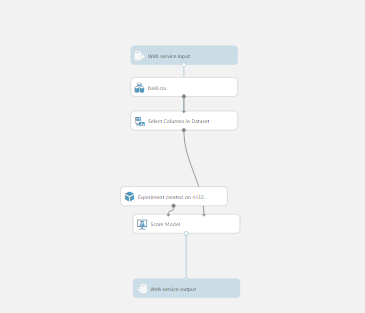
Term : 60 and Home\_ownership: Mortgage



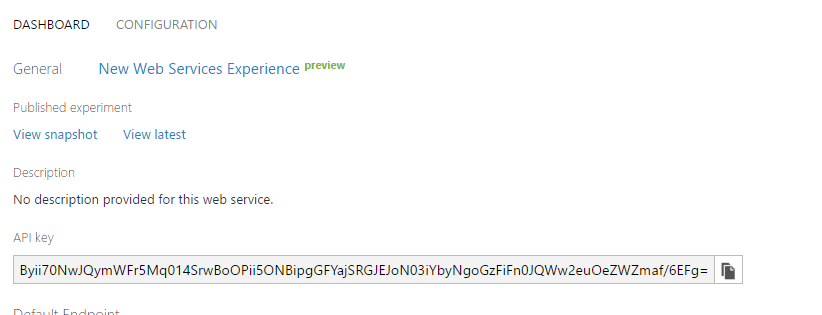
Output:



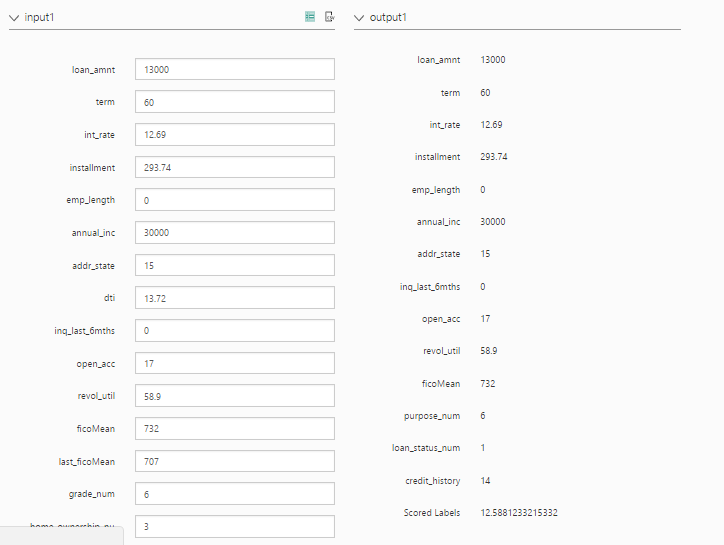
Predictive Model



Deploy Bin 0 web service and generate API:

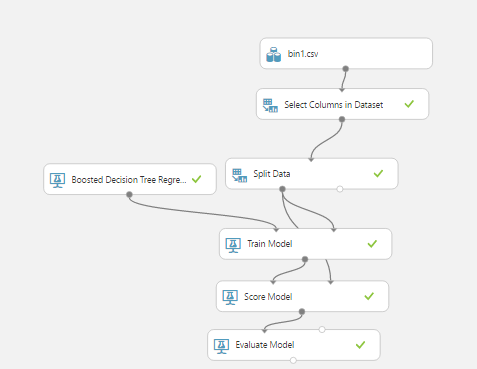


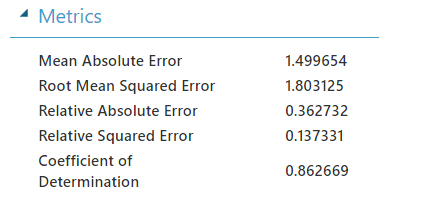
Prediction Output:



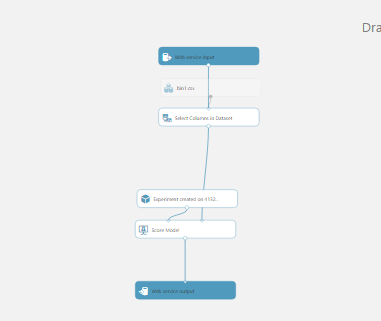
##### Bin 1

Term: 60, Home\_Ownership: OTHER

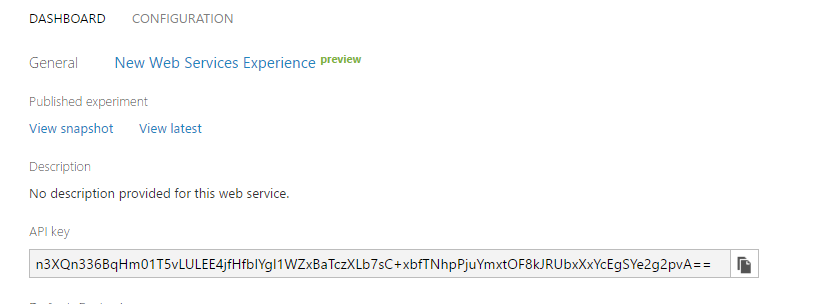




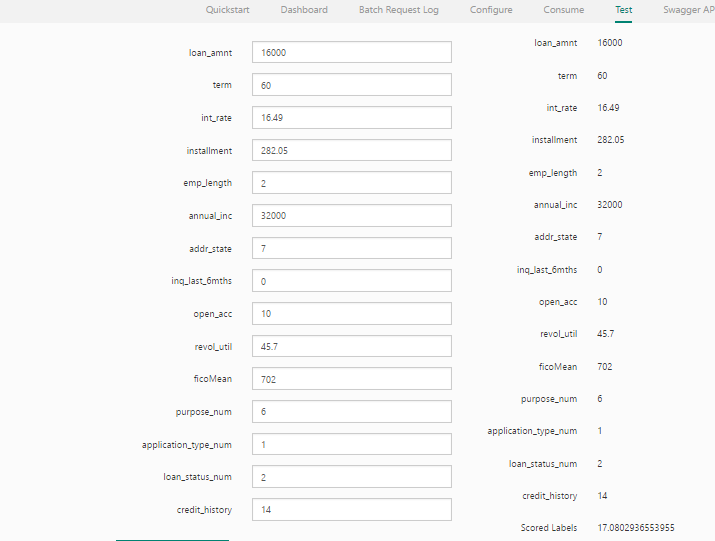
Web service:

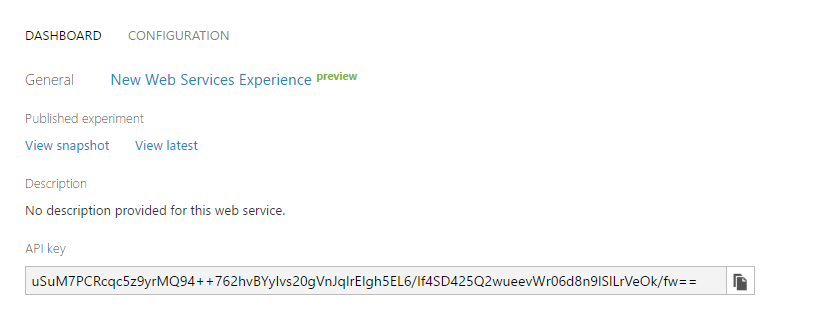


Deploy Web service:

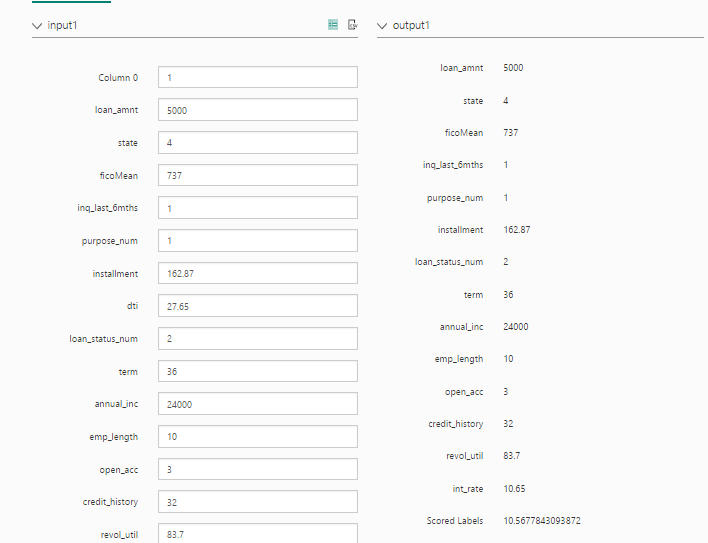


Prediction output:



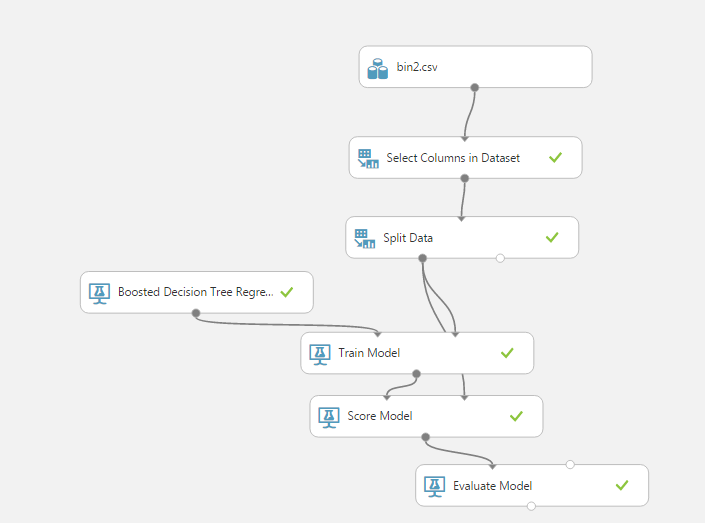


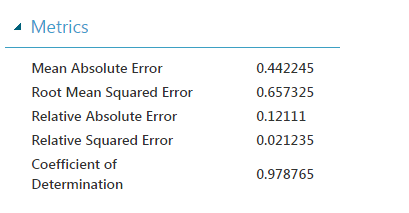
Output of prediction:

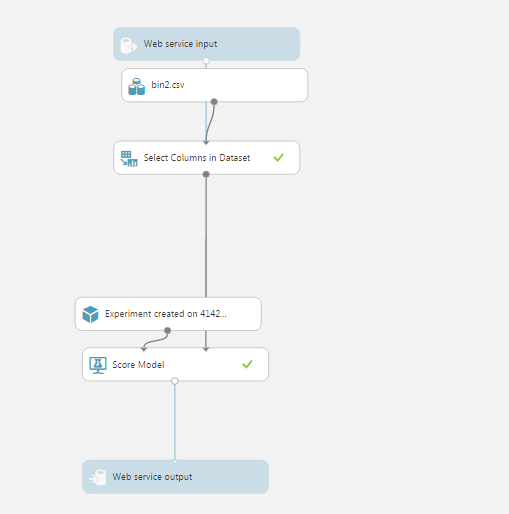


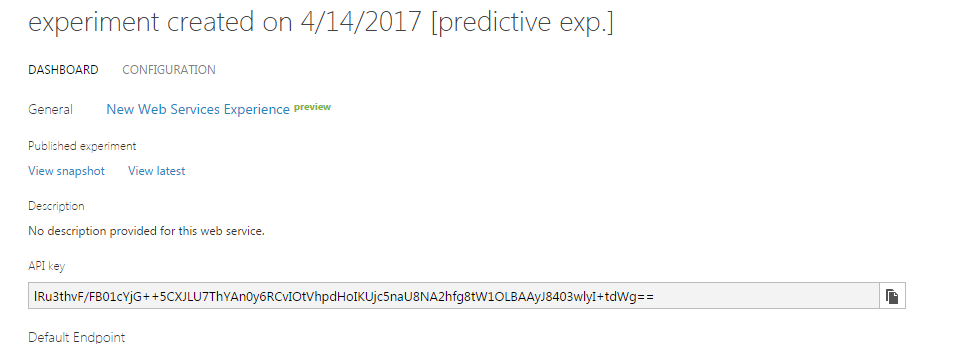
##### Bin2

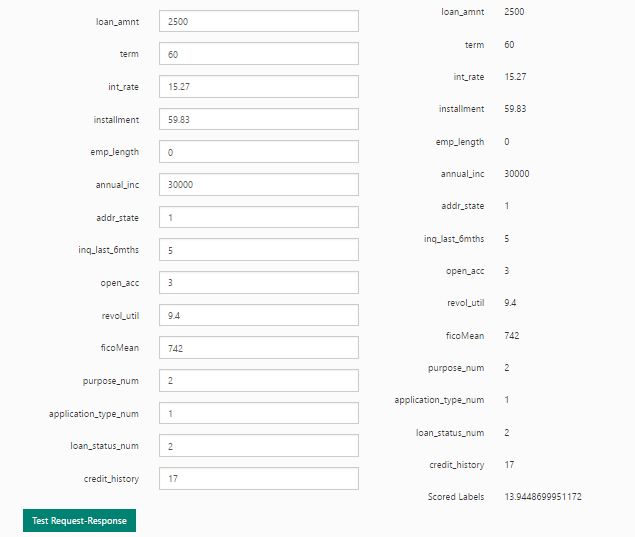
Term: 60 Home\_ownership: OTHER











##### Results for above 3 bins:

|  |  |  |
| --- | --- | --- |
| **Bin Number** | **RMSE** | **R square** |
| **0** | **0.609648** | **0.981745** |
| **1** | **1.803125** | **0.862669** |
| **2** | **0.657325** | **0.978765** |

Similarly, we have created models on azure for remaining 5 bins.

# Web Page

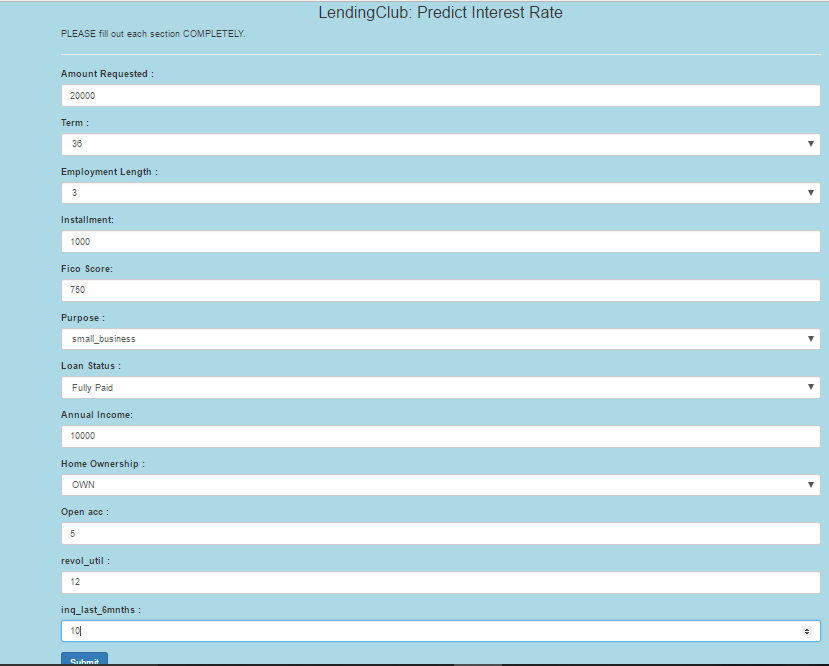
Form 1:

User will enter data on this page based on which it will be decided whether a loan is accepted or not using classification algorithm:



Form 2:

If the loan is accepted, user will be directed to next page where he will enter details related to securing loan:



After all details are entered, the predicted interest rate will be displayed.

At the backend we have used Spring boot which will accept input from web page and trigger corresponding APIs and give the final result of prediction.

# GitHub Links:

Github Repo: <https://github.com/vaidehi1305/ADSAssignment2>