Advanced Data Science & Architecture

Loan Data Analysis

* *Under the guidance of Sri Krishnamurthy*

Compiled By,

Mohit Mittal

Sneha Ravikumar

Taj Poovaiah

Contents

[About Application: 3](#_Toc479974428)

[STEPS TO RUN THE Pipeline 3](#_Toc479974429)

[DOCKER ON AMAZON AWS SCHEDULING: 4](#_Toc479974430)

[Downloading Data: 7](#_Toc479974431)

[Cleaning Data: LOAN DATA SET 8](#_Toc479974432)

[SUMMARY OBSERVATIONS AFTER DOWNLOAD 8](#_Toc479974433)

[SUMMARY OBSERVATIONS POST CLEANING 11](#_Toc479974434)

[Handling Missing Data: LOAN DATA 13](#_Toc479974435)

[ClEANING: Declined Loan Data Set 15](#_Toc479974436)

[Summary Observations made after replacing the missing values of the risk score with median and mean. Replaced the missing values with 999 16](#_Toc479974437)

[Microsoft Azure ML STUDIO to cross validate our findings and visualize Cleaning, Summary, and Feature Engineering 17](#_Toc479974438)

[Checking for missing values using Summarize Data using Microsoft Azure ML Studio 17](#_Toc479974439)

[Cleaning data and checking for missing values using Microsoft Azure ML Studio 18](#_Toc479974440)

[Exploratory Data Analysis: LOAN DATA 19](#_Toc479974441)

[USING JUPYTER NOTEBOOK AND R Published Link: http://rpubs.com/Palecanda 19](#_Toc479974442)

[ANALYSIS for Loan Data: GRADES 19](#_Toc479974443)

[LOAN DATA SET GRADE FREQUENCY 19](#_Toc479974444)

[Exploring interest rates based on Grades assigned by Lending Club 20](#_Toc479974445)

[Paid Vs. Unpaid loan amount over the Grades 21](#_Toc479974446)

[INTEREST RATES AGAINST GRADES 22](#_Toc479974447)

[ANALYSIS for Loan Data: LOAN AMOUNT 23](#_Toc479974448)

[LOAN AMOUNT AGAINST LOAN STATUS 23](#_Toc479974449)

[COUNT OF LOAN AMOUNT AGAINST GRADE 24](#_Toc479974450)

[EXPLORATORY ANALYSIS: DECLINED LOAN DATA 25](#_Toc479974451)

[REJECTED LOAN AMOUNT AGAINST EMPLOYMENT LENGTH VS RISK SCORE 25](#_Toc479974452)

[COUNT OF ACCEPTED AND REJECTED LOAN AMOUNT AGAINST STATES 27](#_Toc479974453)

[Totals Funded By State 29](#_Toc479974454)

[Feature Engineering: PART 1 31](#_Toc479974455)

[**Deployment** 33](#_Toc479974456)

[**Classification** 34](#_Toc479974457)

[**ROC CURVES** 35](#_Toc479974458)

[FEATURE SELECTION FOR CLUSTERING USING R 37](#_Toc479974459)

[**Clustering** 38](#_Toc479974460)

[**Clustering using k-means for numeric and categorical values** 38](#_Toc479974461)

[MANUAL CLUSTERING USING PYTHON 40](#_Toc479974462)

[**PREDICTION** 42](#_Toc479974463)

[PREDICTION ON THE ENTIRE DATASET 42](#_Toc479974464)

[**Web Application: HOW IT WORKS** 42](#_Toc479974465)

# About Application:

**Programming Language used : Python**

**Workflow Manager User: Luigi**

**Tasks**

1. **Downloading Loan Data (GetData.py)**
2. **Clean Loan Data (CleanData.py)**
3. **Upload Preprocessed data to Amazon S3 (LoanData.py)**

# STEPS TO RUN THE Pipeline

1. Pull the Docker image (mohit914/test:LCv1.05) from the dockerhub

**docker pull mohit914/test:LCv1.05**

1. Run the docker image. It will take you into the bash terminal

**docker run -ti mohit914/test:LCv1.05**

1. Inside the bash terminal run the python application by using the following command

**python LoanData.py Start –local-scheduler**

**python RejectLoanData.py Start –local-scheduler**

This program will run the following tasks automatically:

1. **GetData()** – It will download the LendingClub.com loan dataset
2. **ClenaData()** – It will clean and preprocess the LendingClub loan dataset
3. **Start()** – It will upload the preprocessed data to Amazon S3 Bucket. ***You will need to provide your Amazon Access key and Secret Key to complete this task.***

***Docker Trouboulshooting:***

* + - 1. *If running docker gives No space on the device error, then remove all the images and follow the steps to run the pipeline again. If problem still exists, remove all the images and containers on your device and follow the steps to run the pipeline again.*
      2. *If the problem still exists, please email us to let us know the problem so that we can try to troubleshoot the problem.*

# DOCKER ON AMAZON AWS SCHEDULING:

**Docker on AWS – Scheduling**

1. Connect to Amazon Linux AMI instance
2. Update the installed package cache on the instance

**sudo yum update -y**

1. Install Docker

**sudo yum install -y docker**

1. Start the docker service

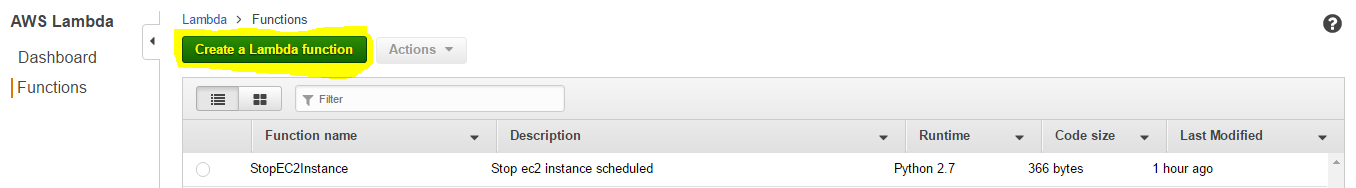
**sudo service docker start**

1. Add the ec2-user to the docker group so that you can execute Docker commands without using **sudo usermod -a -G docker ec2-user**
2. Log out and log back in again to pick up the new docker group permissions
3. Pull the docker image from dockerhub

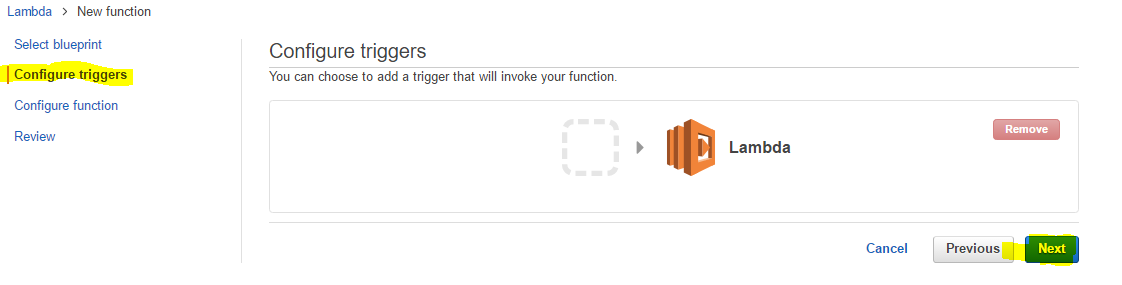
**docker pull mohit914/test:LCv1.05**

**To schedule the task to start and stop the ec2 instance**

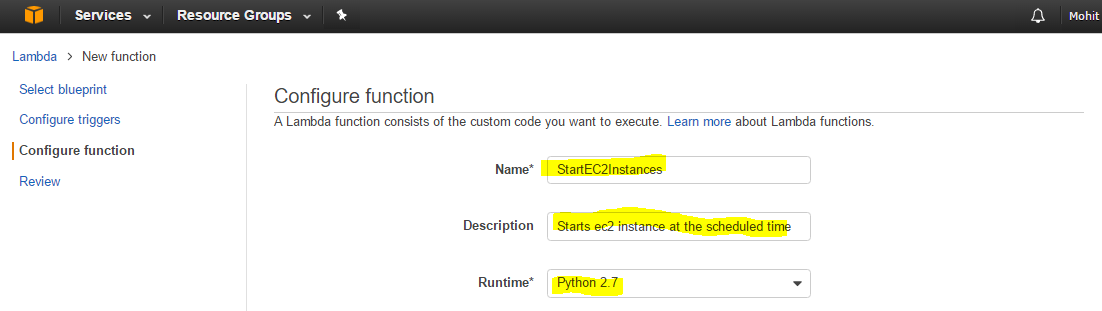
1. Create a Lambda function by going on the AWS Lambda console and clicking on Create Lambda function.



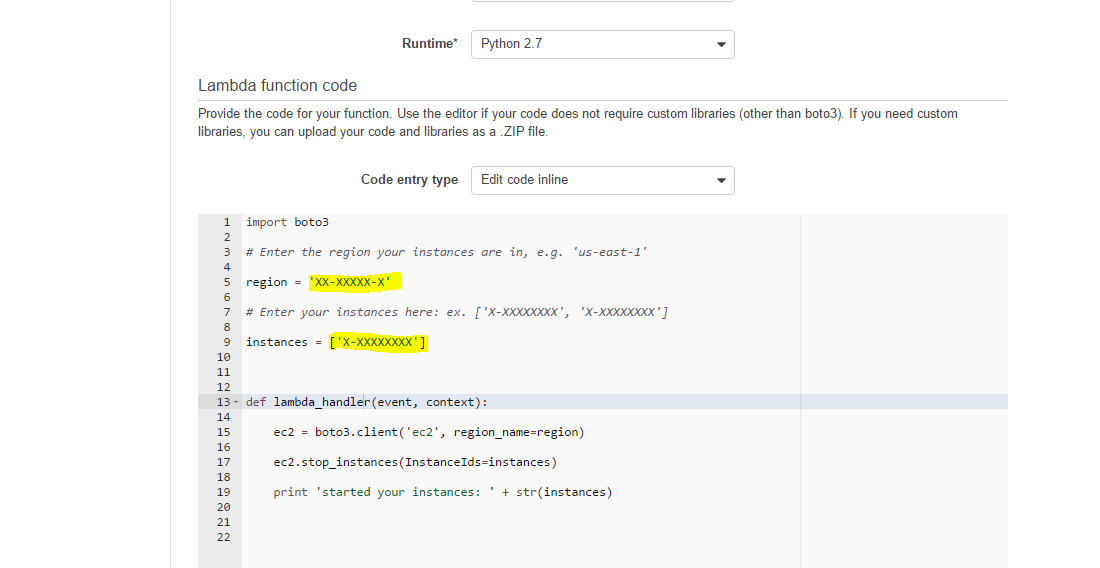
1. Choose Configure triggers and then choose next



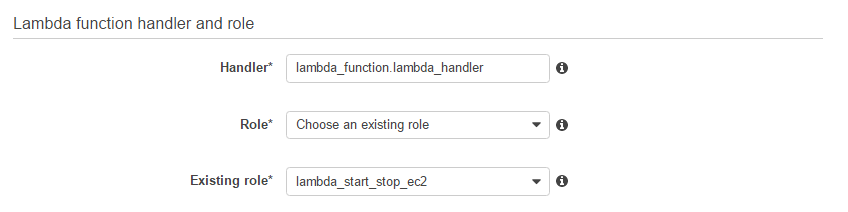
1. Enter the values for name and description of the function and select python 2.7.



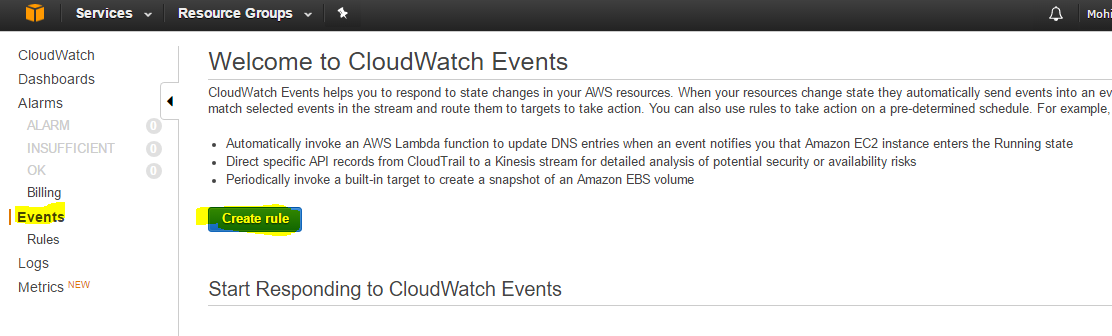
1. Enter the code to start the instance in the “Lambda function code” section below. Replace the highlighted values with your region and instance id respectively.



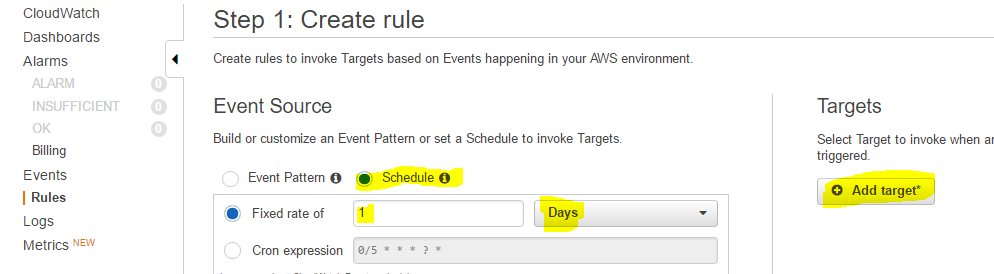
1. Select the role (create if it doesn’t exist to start and stop the ec2 instance)



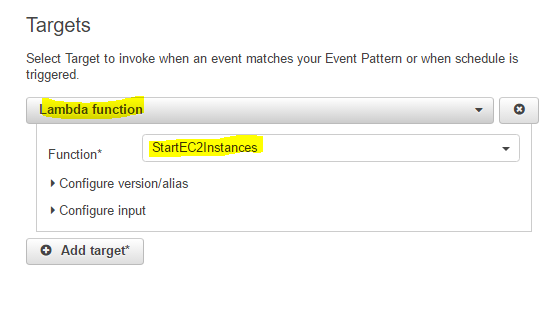
1. Click next and click on Create function
2. Open CloudWatch console and select Events.



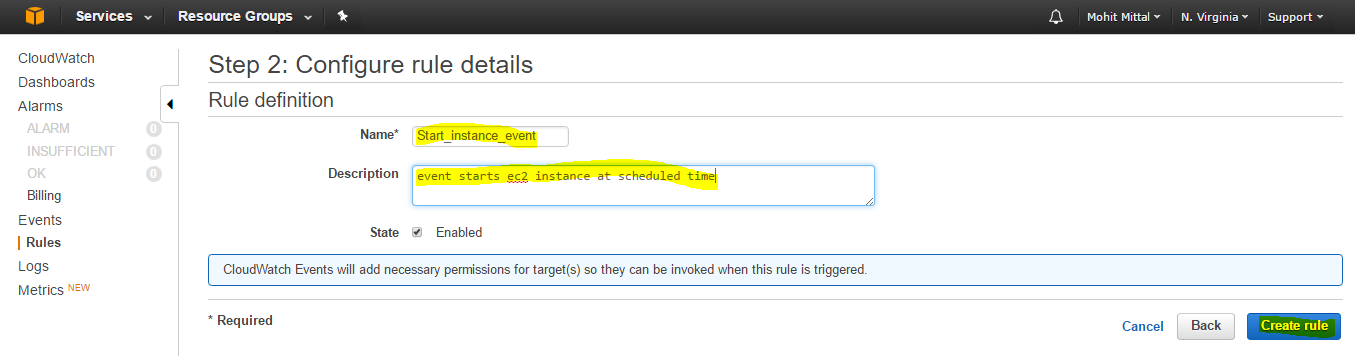
1. Select Schedule under Event Source and select fixed rate of 1 Days. And then click on add target.



1. Select Lambda Functions and the function name.



1. Click on configure details. And enter the name and description for the event.



1. Follow the same steps to create an event to stop the instance on scheduled time.

# Downloading Data:

File Location: Classes/LoanData/GetData.py

Task Requires no prior tasks to be completed.

Output of the task are all the Loan Data files.

Process:

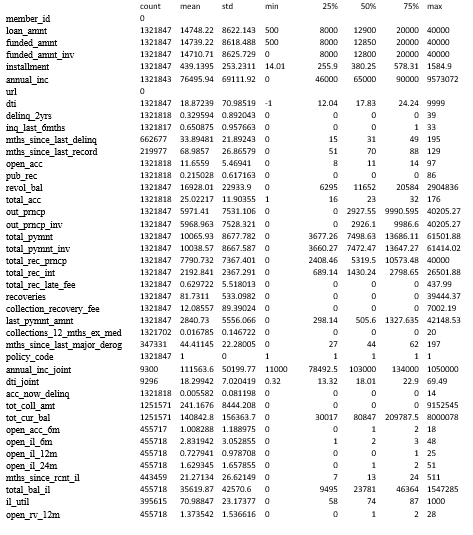
* Asking user for username and password.
* Creating a browser agent (using the mechanicalsoup library) to store and pass the cookies
* Logging in with the user’s credentials
* Checking if the user is successfully logged in or not.
* Landing to the page that contains the list of files and download links
* Putting the table of files in a dataframe
* Iterating through the rows in dataframe for the links that contain sample files and downloading them to a newly created (if it doesn’t already exist) “Downloads” directory
* The program also checks if the files are already present in the “Downloads” directory. It skips the downloading if the file already exists.
* Unzipping the downloaded file.

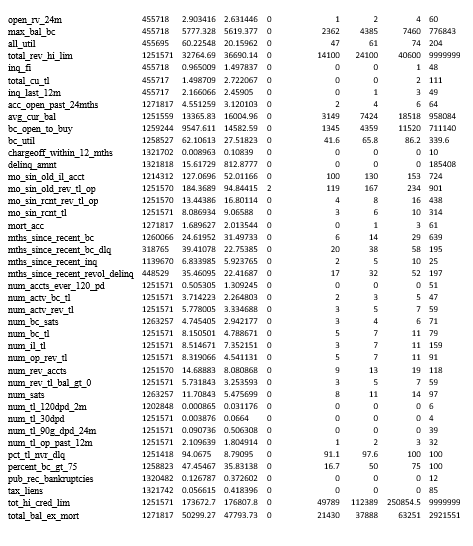
The Downloaded data can be found in the directory : “Data/Downloads/”

# Cleaning Data: LOAN DATA SET

## SUMMARY OBSERVATIONS AFTER DOWNLOAD

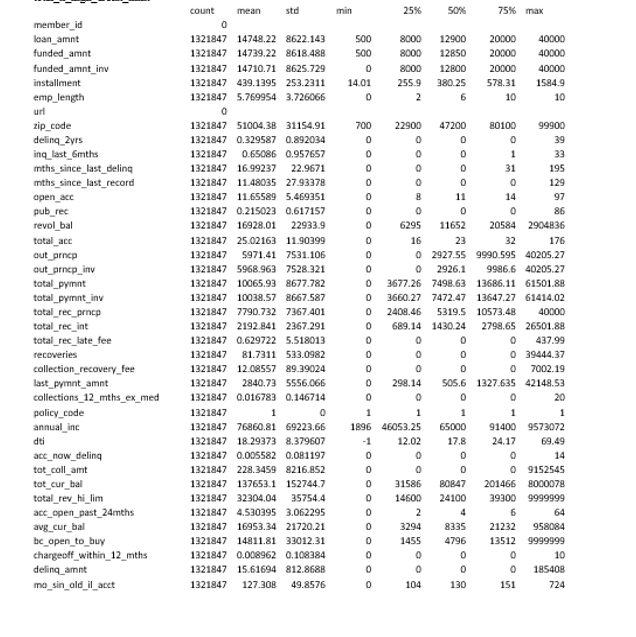
We obtained the summaries for Loan data after download and after cleaning. We see that the count of missing values have been changed. All the columns after cleaning have the same number of values as seen by the count value. The details of how we are handling missing data in each column is given after “summary of data after cleaning”

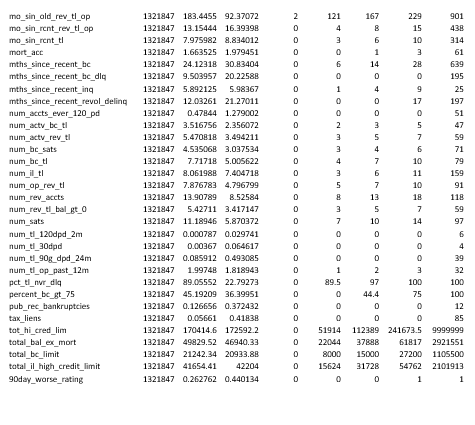






## SUMMARY OBSERVATIONS POST CLEANING





## Handling Missing Data: LOAN DATA

Missing Data analysis was undertaken in two phases:

1. Handling Numeric columns

2. Handling text columns

Handling Numeric Columns:

**Step 1:**

For those columns whose distribution doesn’t change mush on replacing with zero, the same was done.

**delinq\_2yrs** has 29 missing observations and, we can replace those with zero, giving lendors the benefit of the doubt they wouldn't forget someone delinquent.

**inq\_last\_6mths** was done in a similar manner.

**mths\_since\_last\_delinq** is also replaced by 0 because of the same reason.

Other columns imputed with zeroes are - **open\_acc, pub\_rec, total\_acc, collections\_12\_mths\_ex\_med, acc\_now\_delinq**

**Step 2:**

Numeric columns where missing values can be replaced by median without changing the distribution of the data, we can replace it by median.

Example:

annual\_inc has only 4 missing observations so we do a median value imputation with this feature.

tot\_coll\_amt will involve a median value imputation.

fullData ['tot\_coll\_amt'] = fullData ['tot\_coll\_amt'].fillna(fullData ['tot\_coll\_amt'].median())

Other columns for median replacement:

**Step 3**:

We also included a few derived columns to get some necessary insights and information:

mths\_since\_last\_major\_derog will be changed to a new variable where missing values = 0

for no derogs and non-missing = 1 for atleast 1 derog. feature will be named 90day\_worse\_rating

fullData ['90day\_worse\_rating'] = np.where(fullData ['mths\_since\_last\_major\_derog'].isnull(), 0,

1)Joint Account Type and Individual Account Type were mutually exclusive. So were the incomes. Hence they were put together to form a single column

open to buy = credit limit - (sum of holds and outstanding balance) assuming that sum of holds and outstanding balance is zero in this case fullData['bc\_open\_to\_buy'].fillna(fullData['tot\_hi\_cred\_lim'], inplace=True)

With the high & low credit ranges, we were able to calculate the risk score

fullData['risk\_score'] = fullData[['fico\_range\_low', 'fico\_range\_high']].mean(axis=1)

**Step 4**: Delete those insignificant columns where most of the column is empty and will not let us analyze anything

features below are being dropped due to their significantly high proportion of

missing values or they are date values.

fullData = fullData.drop(['earliest\_cr\_line', 'last\_pymnt\_d', 'next\_pymnt\_d', 'last\_credit\_pull\_d',

'annual\_inc\_joint','dti\_joint', 'verification\_status\_joint', 'open\_acc\_6m', 'open\_il\_6m',

'open\_il\_12m', 'open\_il\_24m', 'mths\_since\_rcnt\_il',

'total\_bal\_il', 'il\_util', 'open\_rv\_24m', 'open\_rv\_12m', 'max\_bal\_bc', 'all\_util','total\_cu\_tl',

'mths\_since\_last\_record', 'mths\_since\_last\_major\_derog']

Handling Missing & Cleaning for Text Columns:

**STEP 1**: Check all the text columns that are categorical. We noticed that there weren't any missing values

->Split Issue Date to obtain Month and Year columns.

->Home ownership had about 7 categories. Merged 'other', 'none' and 'any' into a single category called none.

->Employment length was changed to contain numeric values

->The same was done with term and interest rate columns

**STEP 2**: A lot of columns seemed like they couldnt give any important information.

These were dropped - url,dech, emp title

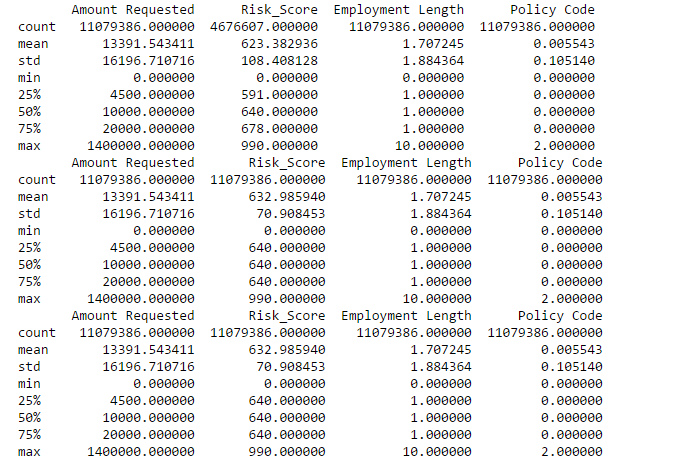
## ClEANING: Declined Loan Data Set

After downloading the Declined Loan Data Set, we observed that there were 9 columns:

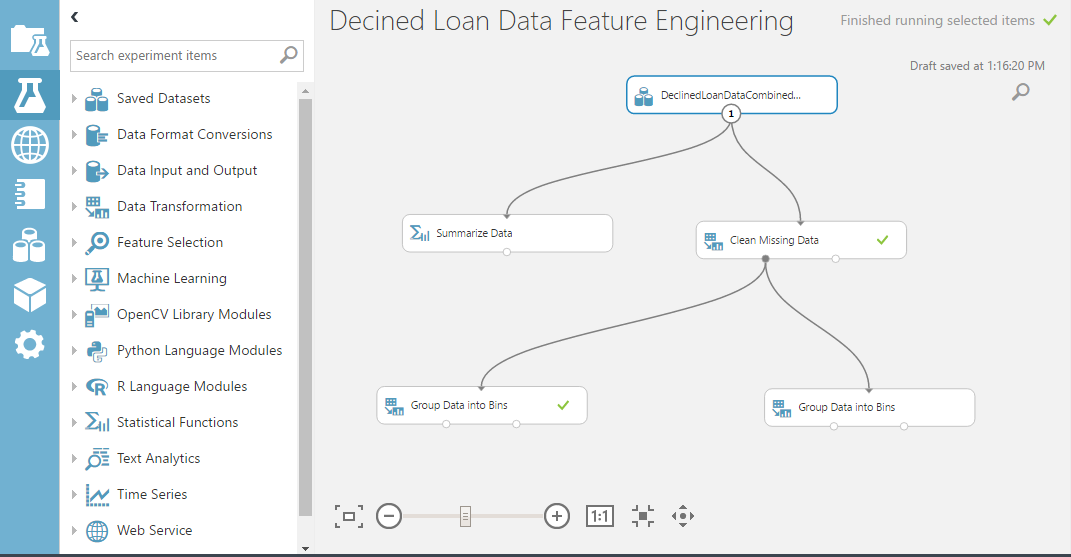
|  |  |
| --- | --- |
| Amount Requested |  |
| Application Date |  |
| Loan Title |  |
| Risk Score |  |
| Debt-To-Income Ratio |  |
| Zip Code |  |
| State |  |
| Employment Length |  |
| Policy Code |  |

1. **Loan Title**: We removed the Loan Title Column since we found a lot of titles that didn’t make any sense and realized that it wouldn’t help with any of our further analysis
2. **Application Date**: We split the application date into Year, Month and Day since we plan on using these columns for future Analysis
3. **State**: We replaced all the missing State values with ‘NA’
4. **Zip Code**: We replaced the the XX’s in Zip Code with 00’s since there’s no way of knowing the last 2 digits. We plan on removing the column in future since it doesn’t make much sense
5. **Risk Score**: We replaced the missing values with ‘999’ since more than 50% of the data was missing and replacing the values with either a mean or median would change the distribution by a large margin

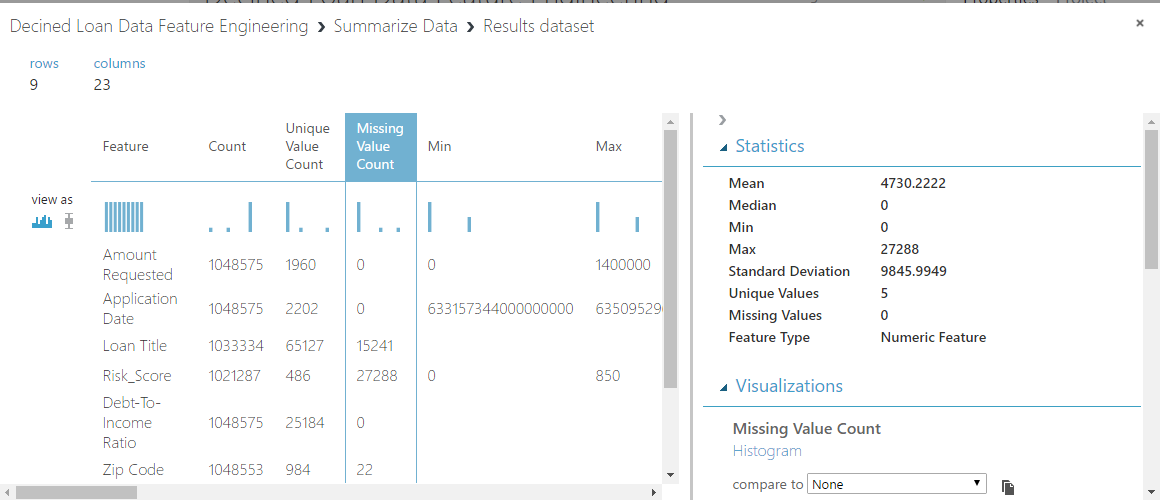
## Summary Observations made after replacing the missing values of the risk score with median and mean. Replaced the missing values with 999



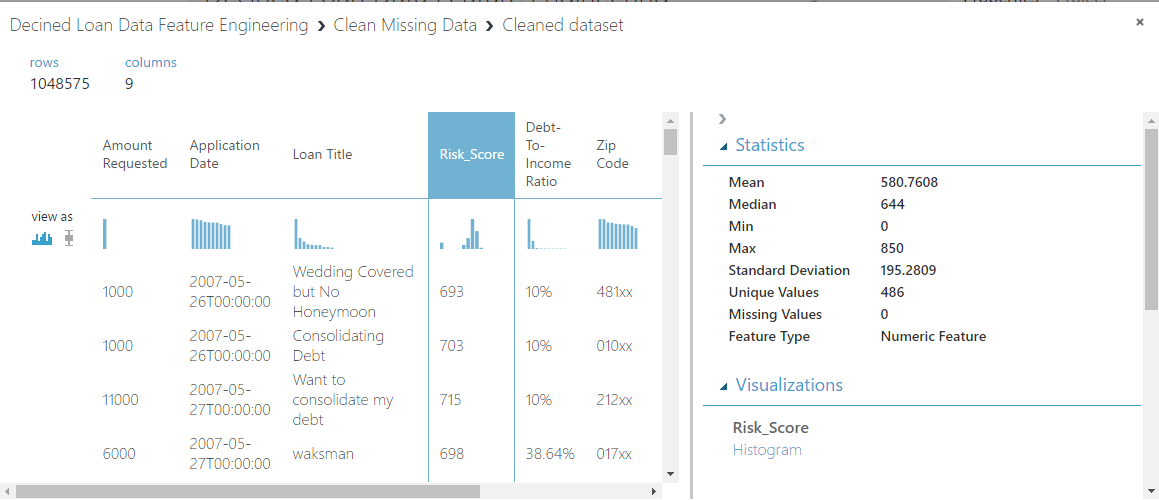
## Microsoft Azure ML STUDIO to cross validate our findings and visualize Cleaning, Summary, and Feature Engineering



## Checking for missing values using Summarize Data using Microsoft Azure ML Studio



## Cleaning data and checking for missing values using Microsoft Azure ML Studio

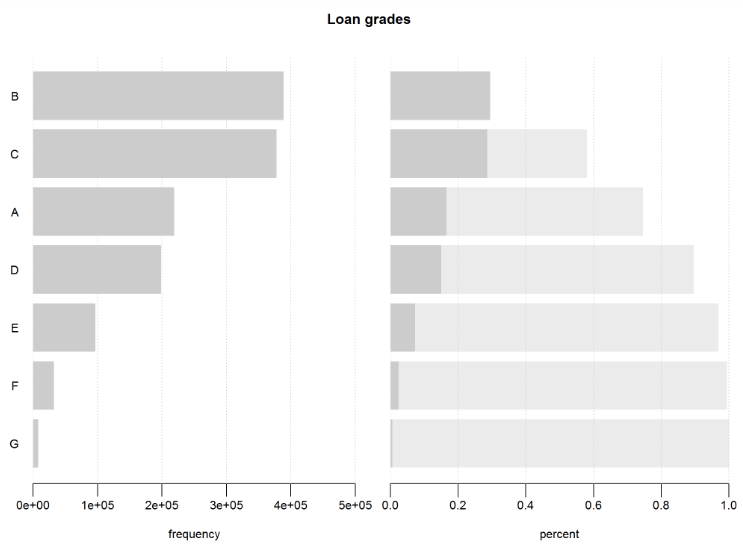


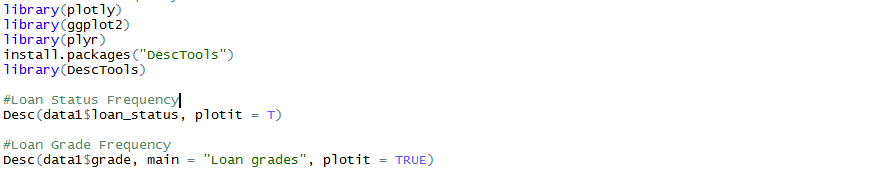
# Exploratory Data Analysis: LOAN DATA

## USING JUPYTER NOTEBOOK AND R Published Link: <http://rpubs.com/Palecanda>

## ANALYSIS for Loan Data: GRADES

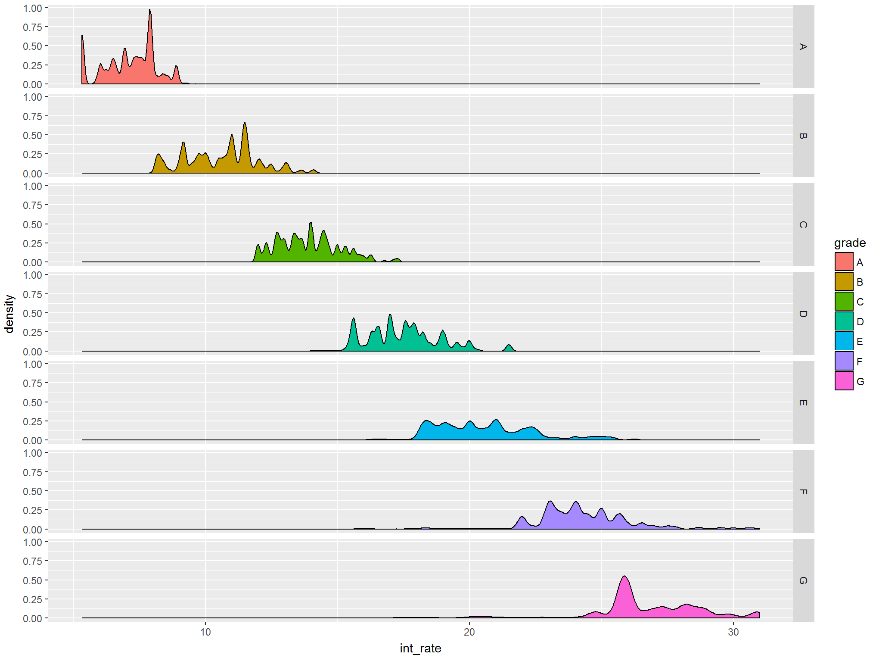
## LOAN DATA SET GRADE FREQUENCY





**By checking the frequency of the grade, we see that people with lower grades (F/G) are very less as compared to people with Higher grades (A,B,C). Hence, people with a good credit score are more in number**

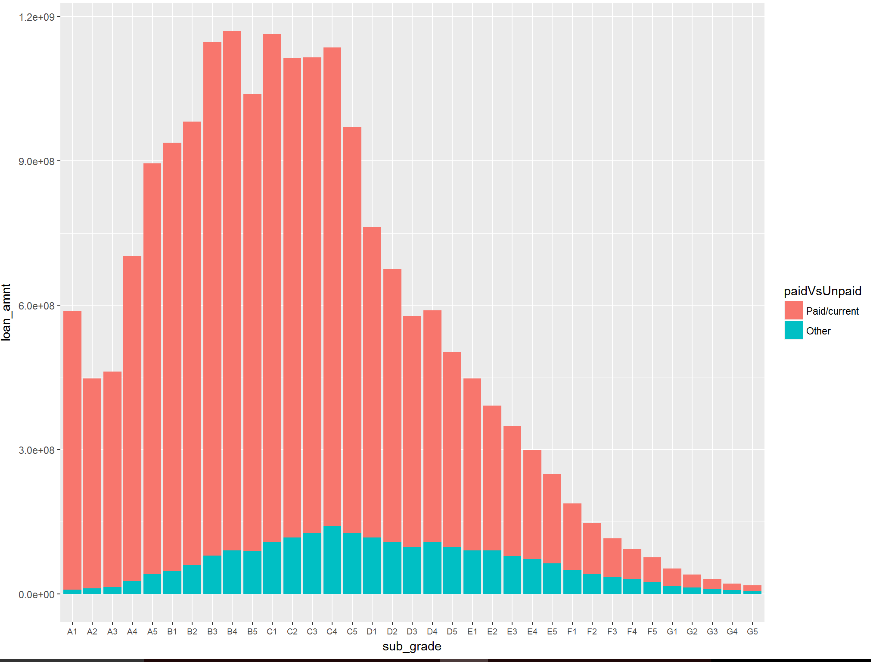
## Exploring interest rates based on Grades assigned by Lending Club

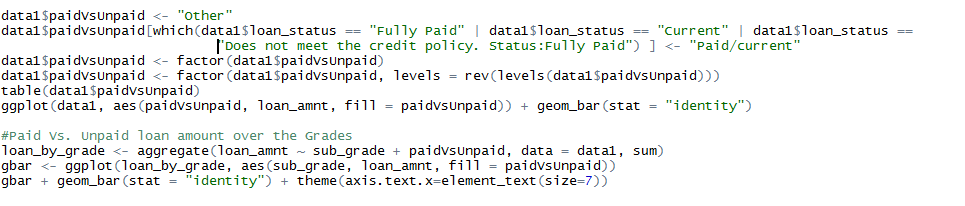




**Grades are assigned based on risk, and so interest rates go up as the risk goes up. Here we see that the interest rate is the highest for grades F and G, since their credit score, hence risk score is low**

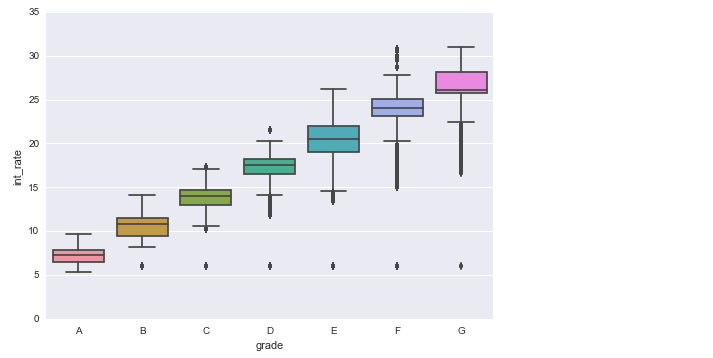
## Paid Vs. Unpaid loan amount over the Grades

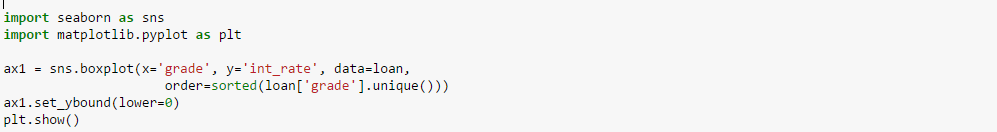




Here, we have created new column with 2 factor levels. 1) “Paid/current” - Represents the status is Current or Fully Paid. 2) “Other” - Represents defaults, charger-off and other status

## INTEREST RATES AGAINST GRADES

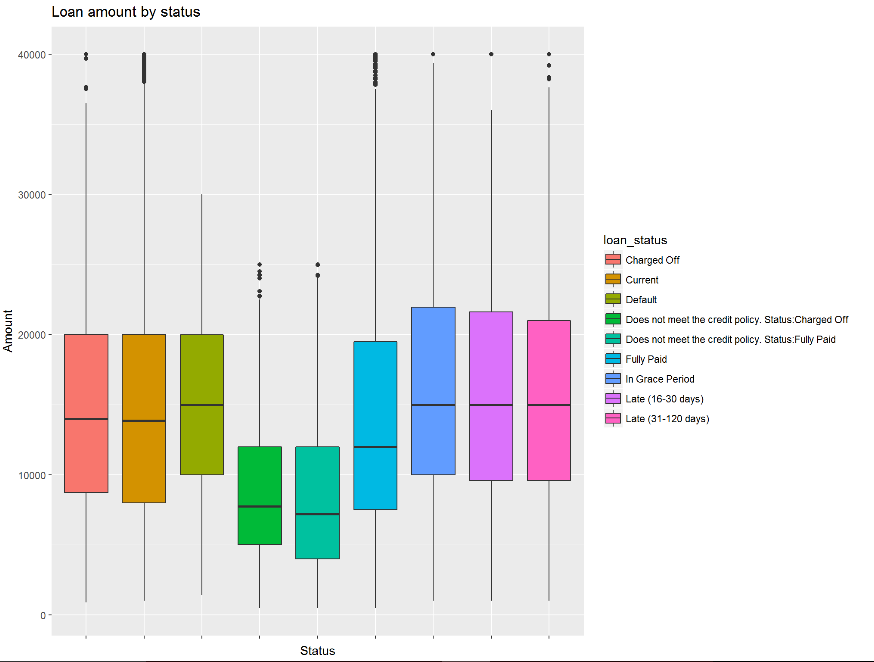


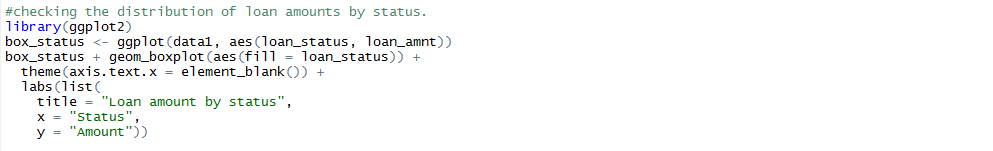


**We observe that the loans have a higher rate of interest until one gets to the higher grades such as A and B**

## ANALYSIS for Loan Data: LOAN AMOUNT

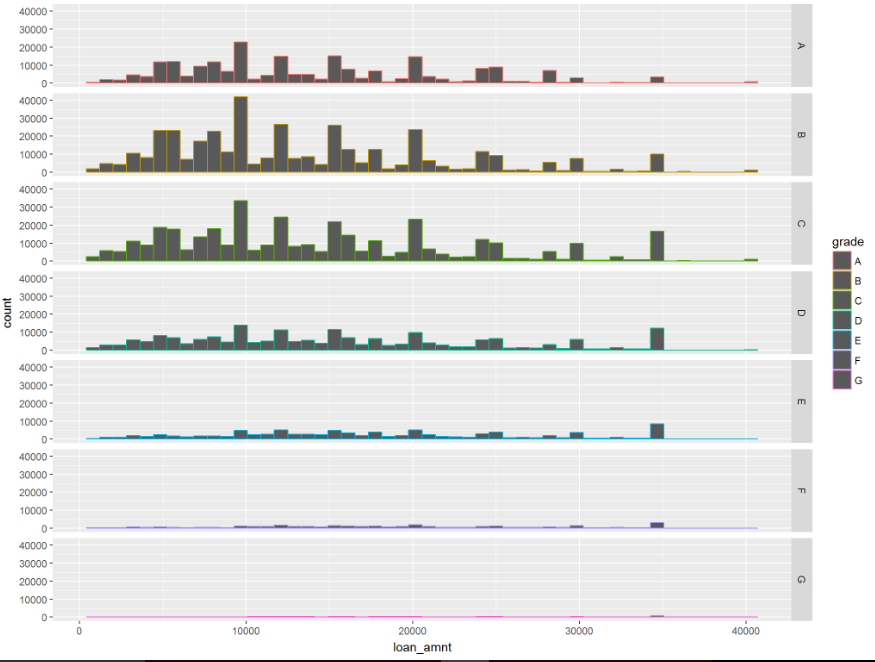
## LOAN AMOUNT AGAINST LOAN STATUS

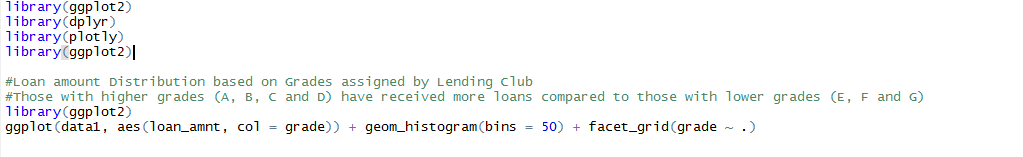




By plotting the loan amount against the loan status, we see that most of the loans are in grace period, or are late

## COUNT OF LOAN AMOUNT AGAINST GRADE

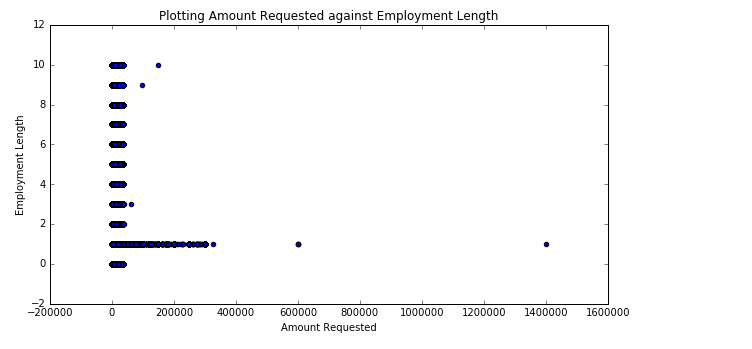


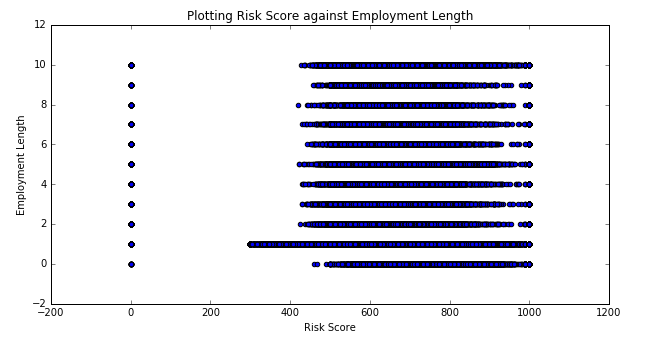


**We observe that, those with higher grades (A, B, C and D) have received more loans compared to those with lower grades (E, F and G)**

## EXPLORATORY ANALYSIS: DECLINED LOAN DATA

## REJECTED LOAN AMOUNT AGAINST EMPLOYMENT LENGTH VS RISK SCORE





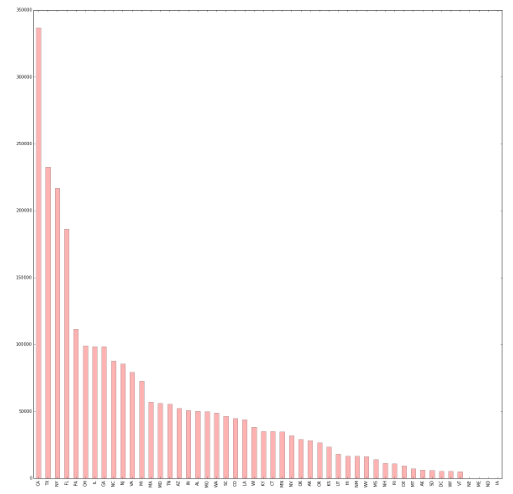
In this Analysis for Declined Loan Data, we have analyzed Employment length against the Amount Requested and Employment length against the Risk Score. We can see that the Loan Requested has mostly been declined for people with an Employment Length between 0-2 years. This may be due to various reasons

1. Their credit scores are low since they just started working
2. After plotting the employment length against the risk score, the above point is pretty evident since the risk score is calculated using the credit scores (FICO scores/2)





## COUNT OF ACCEPTED AND REJECTED LOAN AMOUNT AGAINST STATES



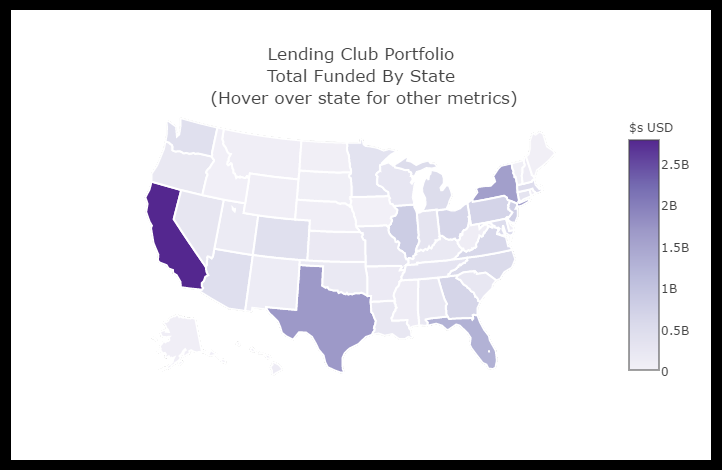




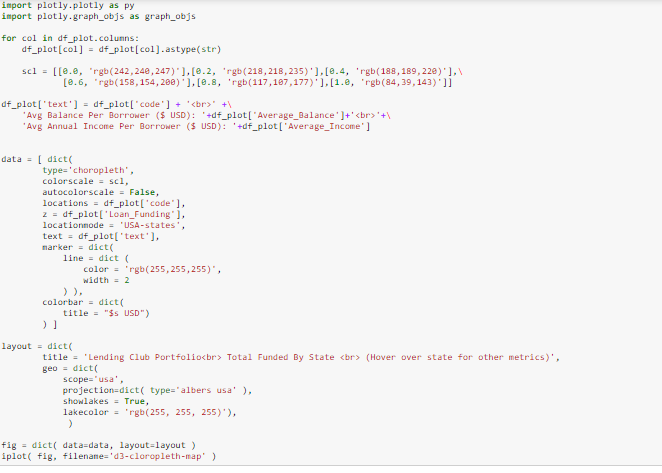
The above 2 graphs shows that the count of the accepted and rejected loan are high in CA, TX, NY and FL. T the reason for this could be that these states were the first to be eligible to get a loan from lending club.



## Totals Funded By State



The above graph gives a detailed analysis of the Average Balance per Borrower and Average Income per borrower in each state. We see that the Averages are the highest in CA



# Feature Engineering: PART 1

We plan on building models and predicting the best features that we can use to calculate the interest rates.

As of now, we plan on selecting the following features:

1. Minimum Credit Score (FICO) – To calculate the Risk Score
2. Maximum Credit Score (FICO) – To calculate the Risk Score
3. Employment Length – Since we’ve seen that the employment length affects the credit score
4. DTI (Debt-to-Income) – Since we have this in both Loan and Declined Loan Data Set and since the DTI is a vital deciding factor
5. State – Since we have this information in both the data sets, and since we’ve seen that the interest rates, income, etc varies with state (May or May not be included)

PART 2

# **Deployment**

Web Interface (Frontend)

Web Application (Backend)

Classification REST API

Prediction REST API 1

Prediction REST API 2

Prediction REST API 3

1. **Web Interface** (**HTML, Javascript, JQuery, CSS, Bootstrap**)
2. Takes the input and shows the result that if we can provide loan or not (if yes, then at what interest rate).
3. Link to PowerBi Visualizations, Jupyter notebook visualization
4. Link to the results of interest rates from all three APIs as professor said in class.
5. Link to PPT and Report
6. **Web Application (Backend: Java)** –

Application (Using Java to interacts with the frontend and the 4 rest apis (1 Classification, 3 Prediction using Python and R). Hosting on AWS

1. **REST API on Azure ML Studio**

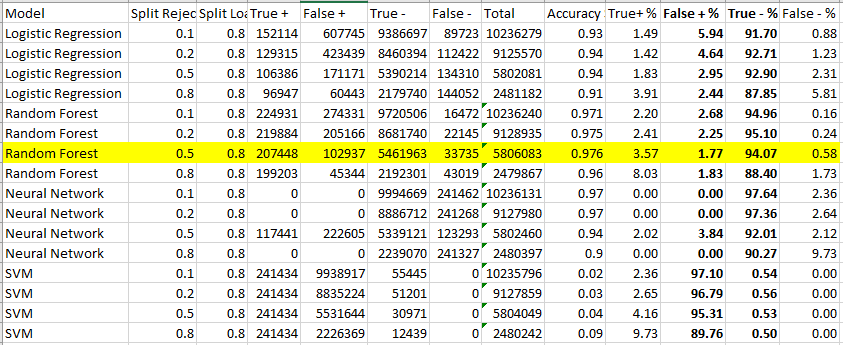
# **Classification**

1. Created a “**Issue\_Loan\_Flag**” column on the Loan Data and the Rejected Loan Data.

Value **= 0 for declined loan** data, **1 for accepted loan** data ***(except the rows which are marked as policy changed, chargedoff and default - we put 0 for those rows)***

1. We ran Logistic Regression, Random Forest, Neural Network and SVM. We selected different percentage of split data for reject loan as the number of rows flagged as reject loan was higher than the number of rows flagged as accepted loan.

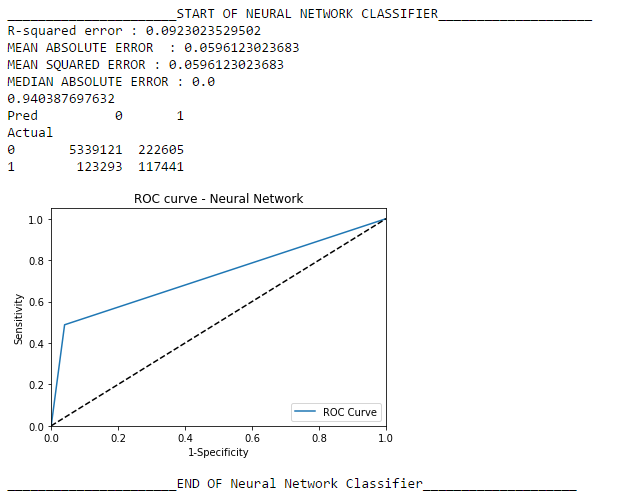
We used 4 features (Loan Amount, Risk score (we assume it is Fico Score), Employment Length and Debt-To-Income ratio.

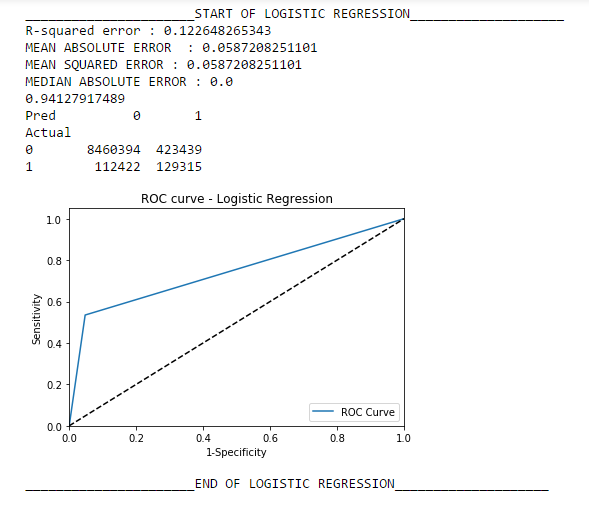
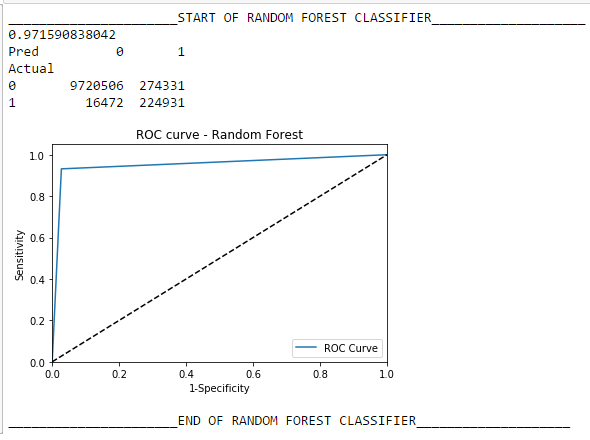
We selected Random Forest (with Split ratio = .5 for reject loan data) as our best model because

1. It has Accuracy of .976
2. Lowest False Positive Rate (1.77 %)
3. Higher True Negative Rate (94.07%)
4. We created Azure ML web service with the selected model and interacted with it using Java – Spring Web Application.

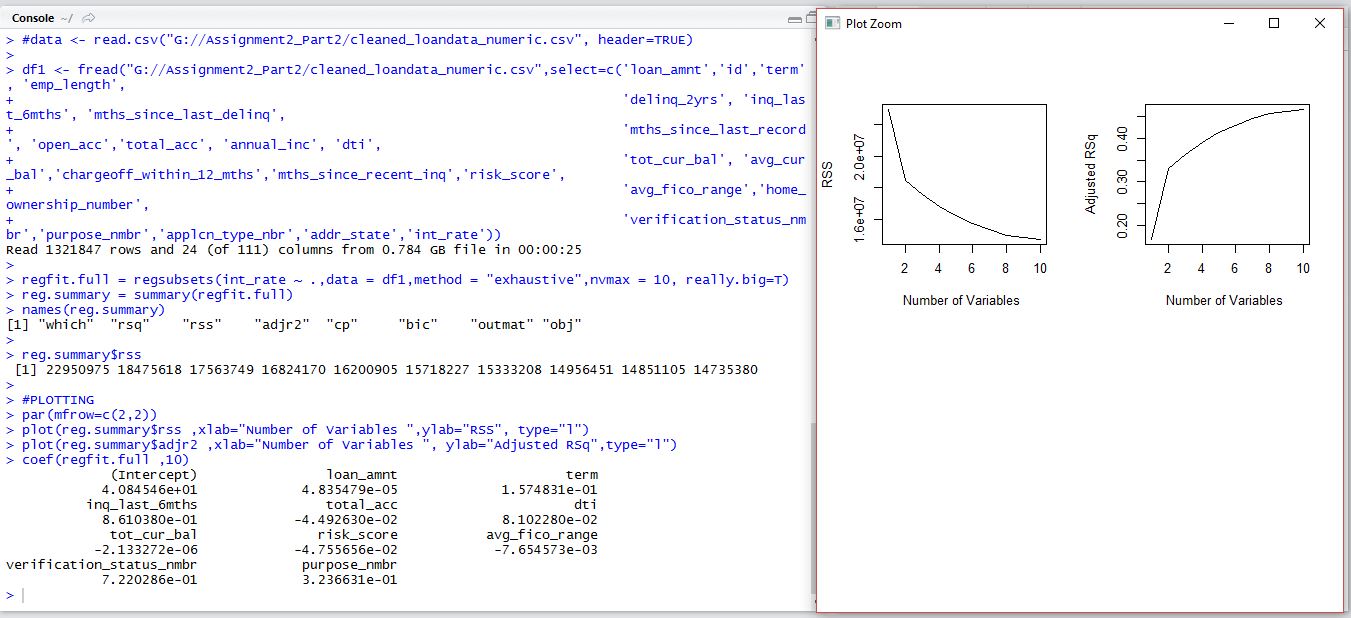
# **ROC CURVES**

* Neural Network Classification Link: <https://nbviewer.jupyter.org/github/pptaj/Assignment2_Final/blob/master/Classification/NeuralNetClassification.ipynb>
* SVM Classification Link:
* <https://nbviewer.jupyter.org/github/pptaj/Assignment2_Final/blob/master/Classification/SVMClassification.ipynb>
* Logistic Regression Link:
* <https://github.com/pptaj/Assignment2_Final/blob/master/Classification/LogisticRegression.ipynb>
* Random Forest Link: <https://github.com/pptaj/Assignment2_Final/blob/master/Classification/RandomForest.ipynb>



# FEATURE SELECTION FOR CLUSTERING USING R

* We first factorized the categorical columns in the entire Loan Data Set
* We tried running a feature selection on 45 columns based on correlations. Since we were facing issues with the algorithms, we dilled it down to 25 columns based on correlation, and then ran an exhaustive feature selection on those columns
* We then selected the 10 columns, including the factorized states, id, and interest rate columns for prediction

|  |
| --- |
|  |
| Loan Amount |
| Loan Term |
| Inquiry Last 6 Months |
| Total number of credit lines |
| DTI |
| Total Current Balance of All Accounts |
| Verification Status Number |
| Risk Score |
| Average FICO Range |
| Purpose Number |
| State Address Factors |

# **Clustering**

**Language Used: Python**

# **Clustering using k-means for numeric and categorical values**

* For Clustering , we excluded interest rate column, and selected the below 11 columns to find the clusters
* We then produced the Ben graph which suggested we run the K-means Clustering for 4 Clusters



1. We the ran the K-means Clustering algorithm, to detect the centroids, and the labels



1. These values, were appended along with the other selected columns on to a csv file, which was used for prediction

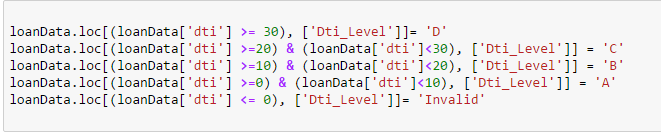
# MANUAL CLUSTERING USING PYTHON

Manual clustering was done on the basis of two derived columns

1. Dti\_Level
2. Credit\_Score\_Level

These two are buckets of Risk\_Score and Dti and are calculated as shown below:

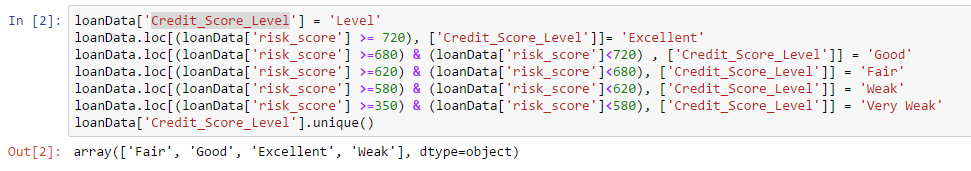
**Dti\_Level:**



There are four classes based on the DTI:

Negative DTI (Ignoring this use case)

1. Between 0 and 10 – A
2. Between 10 and 20 – B
3. Between 20 and 30 – C
4. Dti greater than 30

**Credit Score Level:**

There are five buckets based on the risk score:

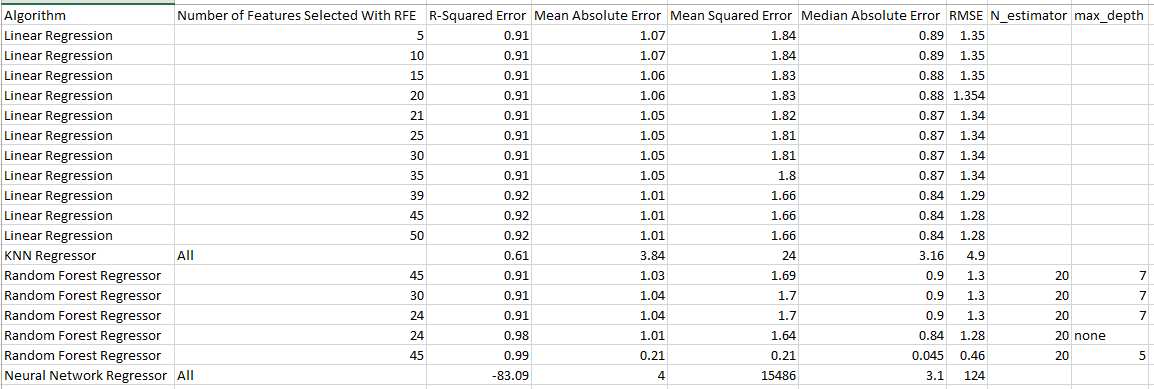
1. Credit Score > 720 -> Excellent
2. Risk Score between 689 and 720 -> Good
3. Risk Score between 620 and 680 -> Fair
4. Risk Score between 580 and 620 -> Weak
5. Risk Score between 350 and 580 -> Good

The Clusters are then assigned based on the combinations available with these columns as below:

Although we have 12 clusters, our data belongs to only the following clusters:

4, 2, 1, 6, 5, 3, 7

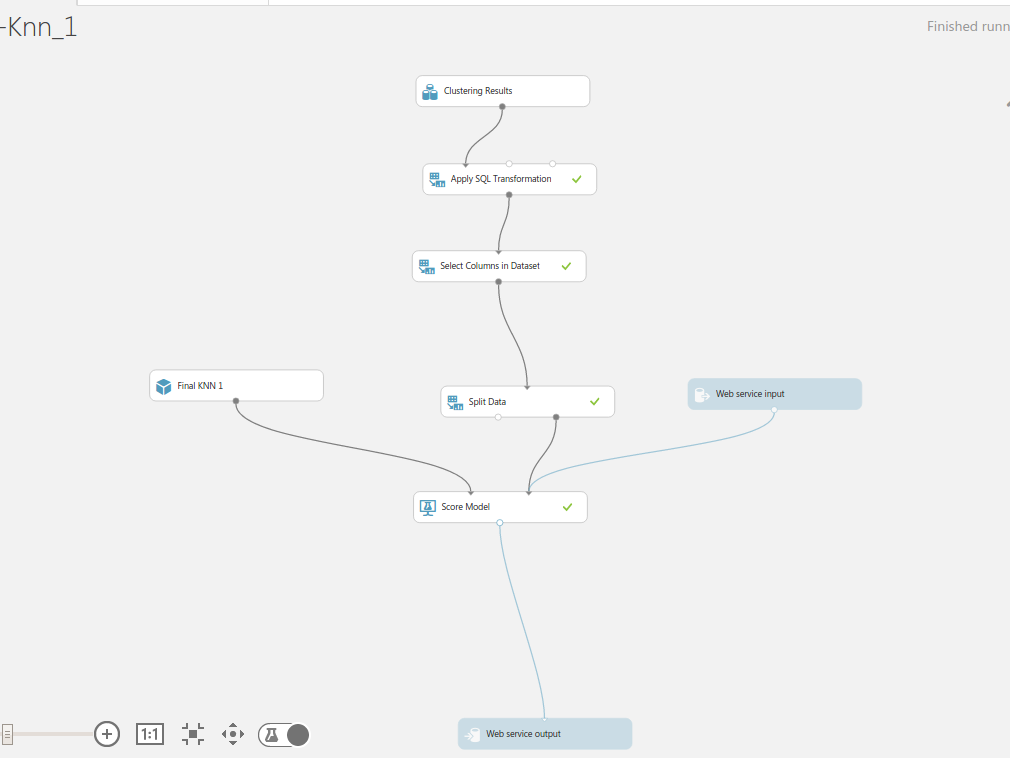
# **PREDICTION**

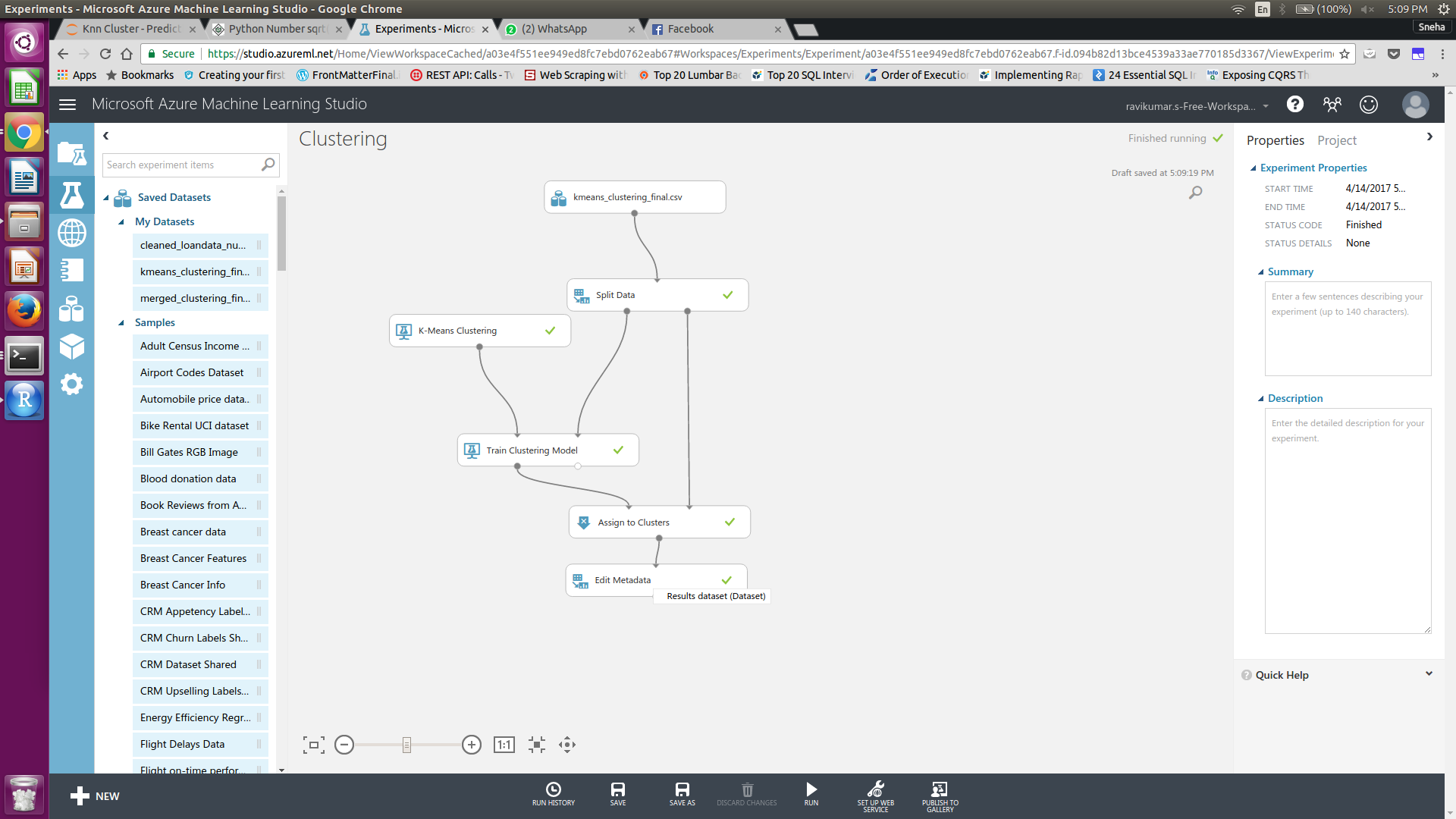


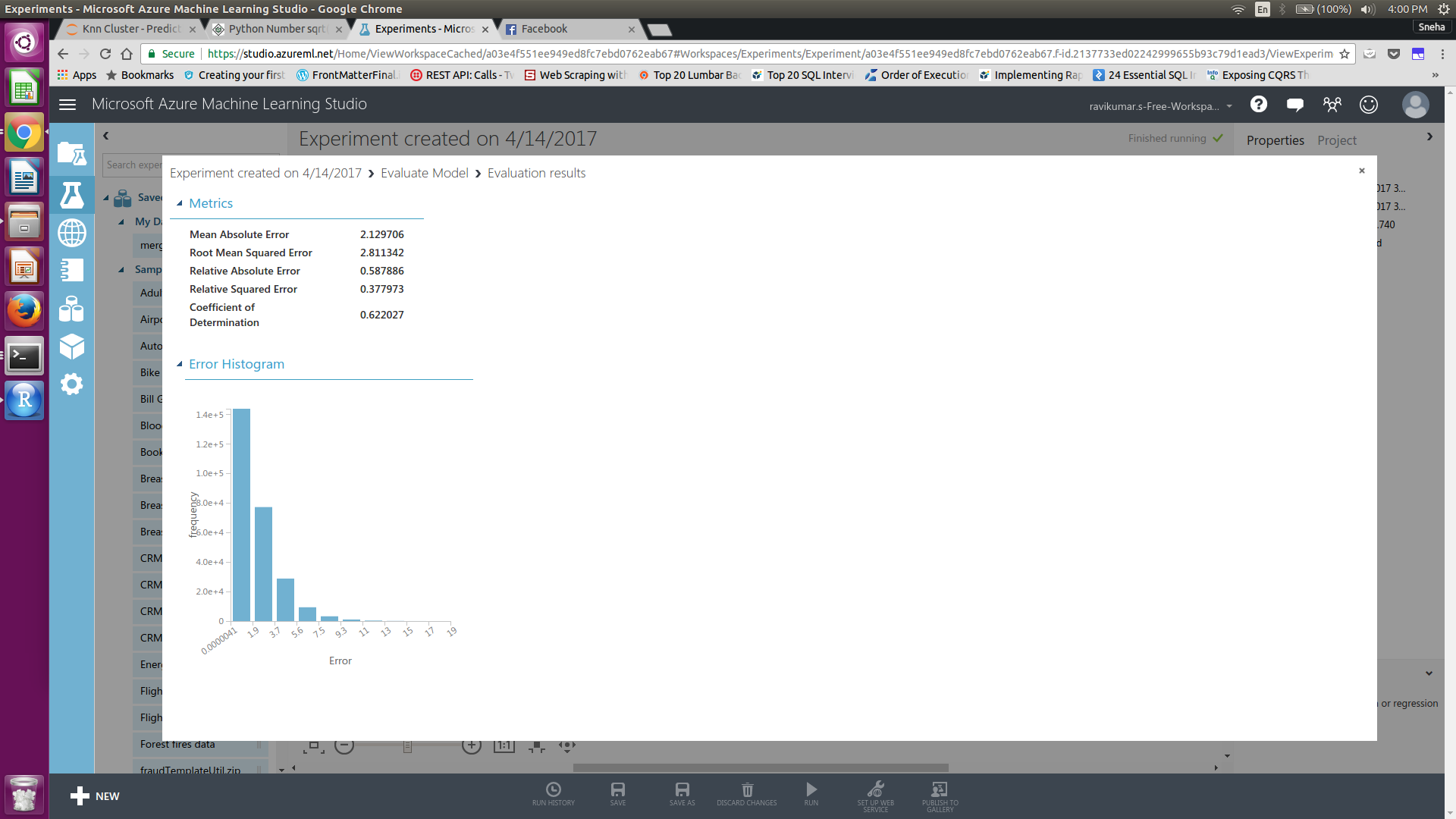
## PREDICTION ON THE ENTIRE DATASET

* We ran the Prediction models using the 4 algorithms, Linear Regression, Random Forest, KNN, and Neural Networks first on the entire dataset
* We got the best results for Random Forest algorithm
* We selected the features using backward, forward, and exhaustive selection and correlations

## PREDICTION MODELS USING MICROSOFT AZURE

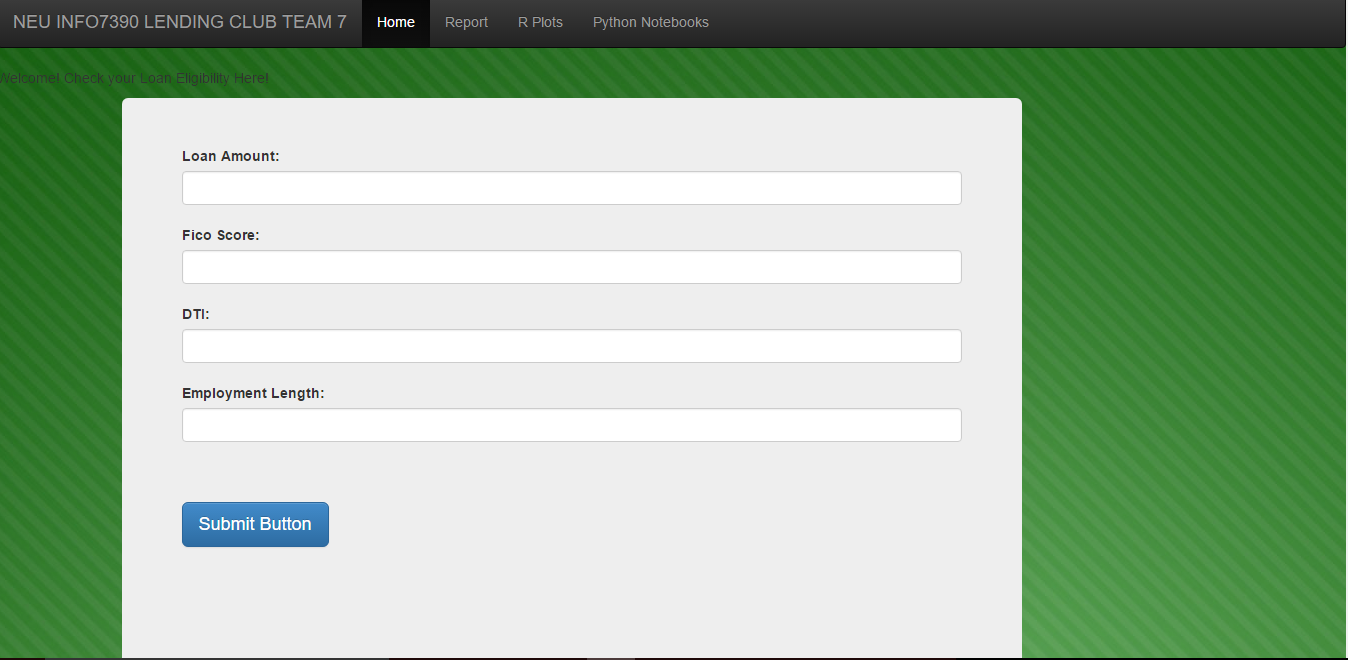






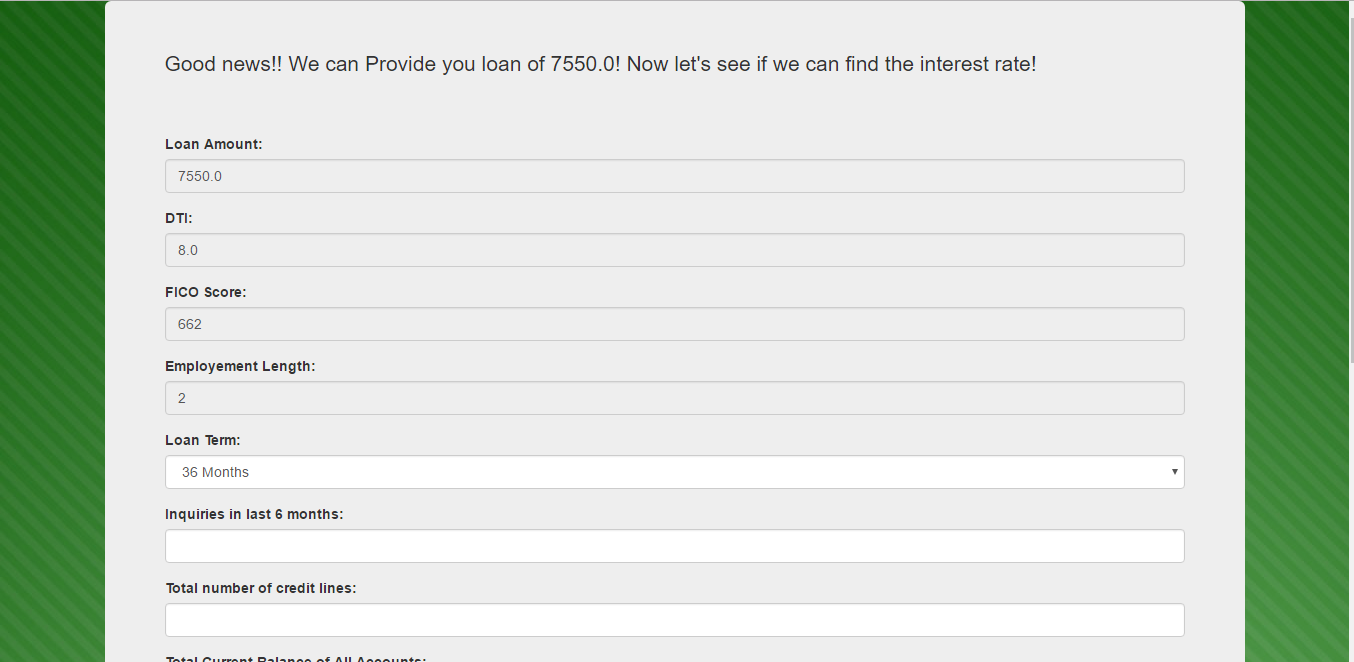
# **Web Application: HOW IT WORKS**

* **Web Application Link**: Sample-env.dvtp6mv4cj.us-east-1.elasticbeanstalk.com
* The Web application takes in the values from the user, predicts the interest rate, and informs the user if he/she is eligible for a loan or not

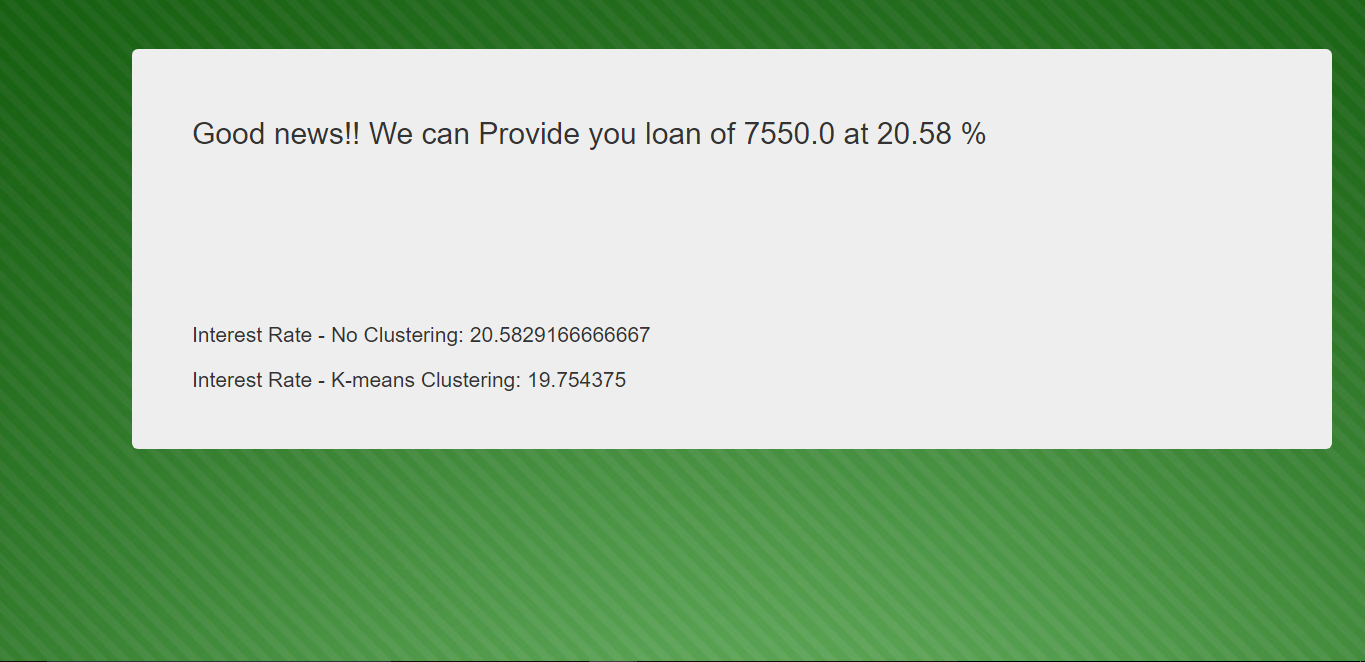


* Once we enter the values above, it takes us to the next page and asks for more information

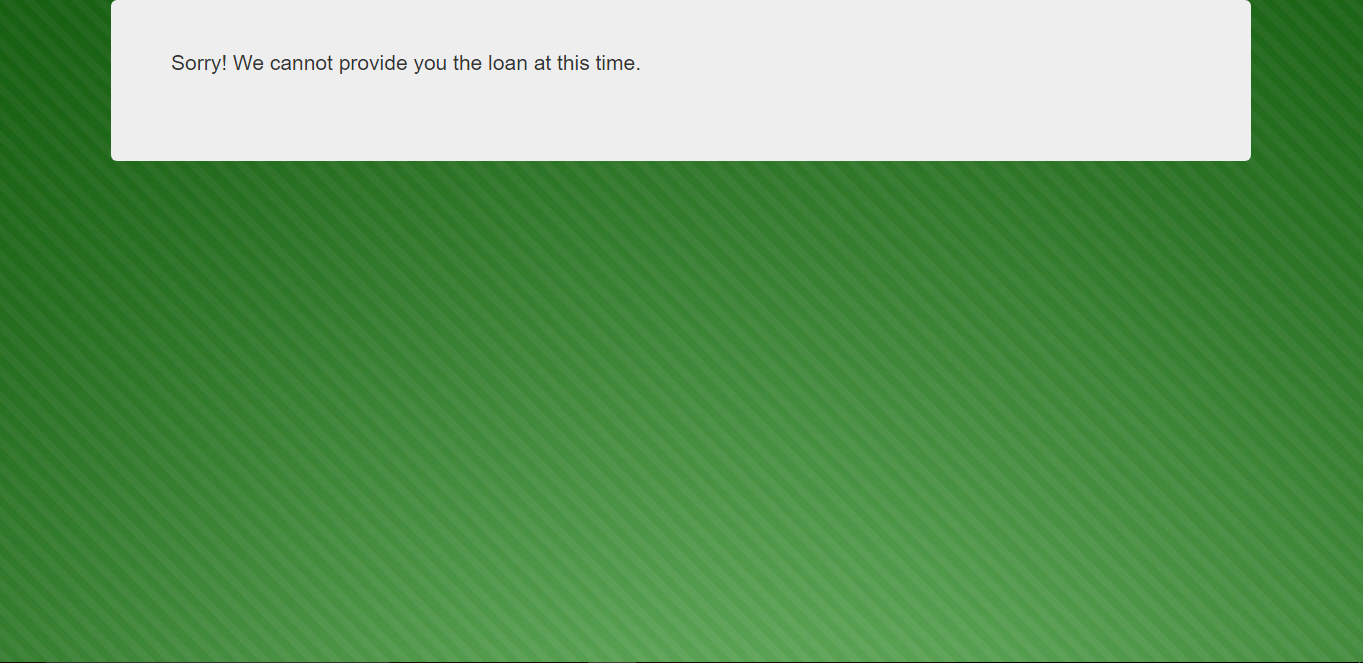
After classifying the values and approving the amount based on the classification analysis that we’ve provided above



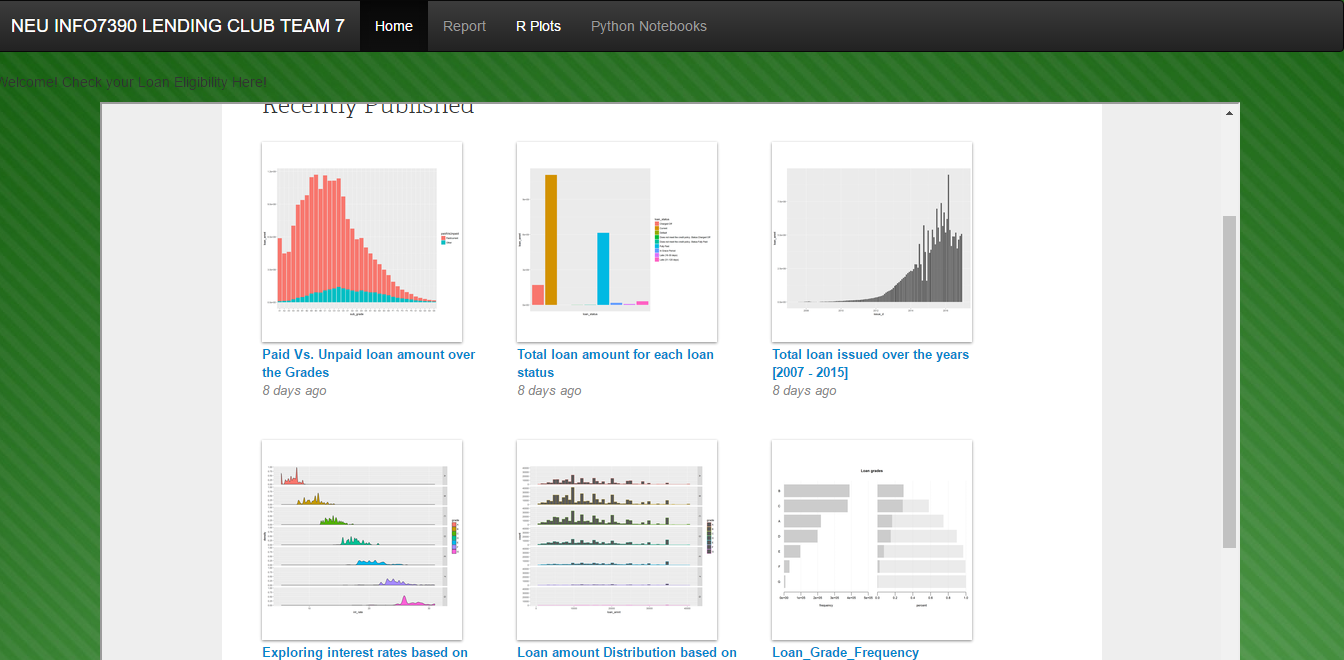
* Once the classification is done, we provide more features, for the prediction of interest rate



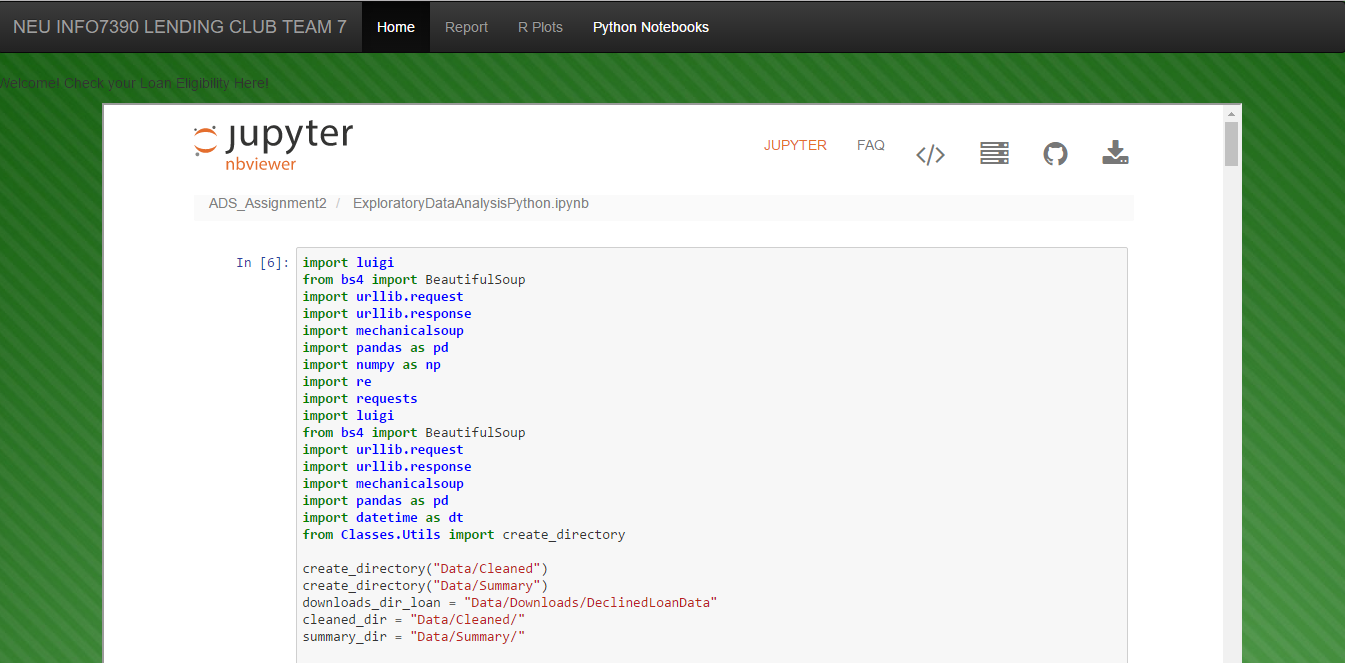
* If the predicted rates, are good, the loan is approved, and the customer is notified



* If the classification says that the values are bad, it informs the customer that the loan cannot be provided
* The web application consists of a bunch of apis that take in the values for the entire data set, and predicts the best interest rates and confirms whether or not one is eligible for loan
* We have 4 clusters for the K-means Clustering, 12 Clusters for manual, and one entire data set for which, we have built apis to predict the interest rates using the Random Forest algorithm as explained earlier
* We have a RestAPI that predicts which cluster each record belong to, and accordingly redirects it to the required cluster’s prediction model
* We have one RestAPI to predict the interest rate for each cluster
* The site opens up the Data Analysis on the RPlots



* It also displays the Analysis on Python Notebooks



END OF REPORT