Advances in Data sciences

Assignment 2

**Under the guidance of Professor Srikanth Krishnamurthy**

**Team 6:** Vaidehi Deshpande,

Puneet Kumar,

Yamini Sehrawat

**Table of Contents**

[Abstract 3](#_Toc479362735)

[Part 1: Data wrangling and Exploratory analysis 4](#_Toc479362736)

[Data Download and pre-processing 4](#_Toc479362737)

[Flow Chart: 5](#_Toc479362738)

[Data Download 6](#_Toc479362739)

[Missing Data Handling 7](#_Toc479362740)

[Feature Engineering 12](#_Toc479362741)

[Luigi 15](#_Toc479362742)

[Luigi Tasks: 15](#_Toc479362743)

[Summarizations 18](#_Toc479362744)

[Loan Dataset 18](#_Toc479362745)

[Power BI Graphs: 18](#_Toc479362746)

[Python Graphs 23](#_Toc479362747)

[Reject Dataset 28](#_Toc479362748)

[Power BI graph 28](#_Toc479362749)

[GitHub Links: 30](#_Toc479362750)

[Docker hub links: 30](#_Toc479362751)

[Run Docker Image 30](#_Toc479362752)

[Scheduling Cron Job to run the docker image: 32](#_Toc479362753)

**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 1: Data wrangling and Exploratory analysis

## Data Download and pre-processing

The first step here is to programmatically download the data from: <https://www.lendingclub.com/info/download-data.action>

Goal is to download the data programmatically from the website and create one dataset for the entire database.

* Missing data analysis
* Feature engineering: Variables we need to predict interest rates? Ensure users would be able to give that information to help predict rates
* Pipeline: Using Luigi/Pinball/Airflow automate the above 3 steps.
* We need to create one more pipeline to do this for the “Declined Loan data”. Repeat above steps

# Flow Chart:

Start

Download data using Python script

Missing Data Handling

Feature Engineering

Luigi Pipeline

Dockerize the task and upload the cleansed files to Amazon S3

Stop

# Data Download

As per the requirement, we have downloaded all the datafiles from lending club and created a single dataset.

Following CSV files are downloaded for Loan and Declined Loan Data:

2007-2011

2012-2013

2014

2015

2016 Q1

2016 Q2

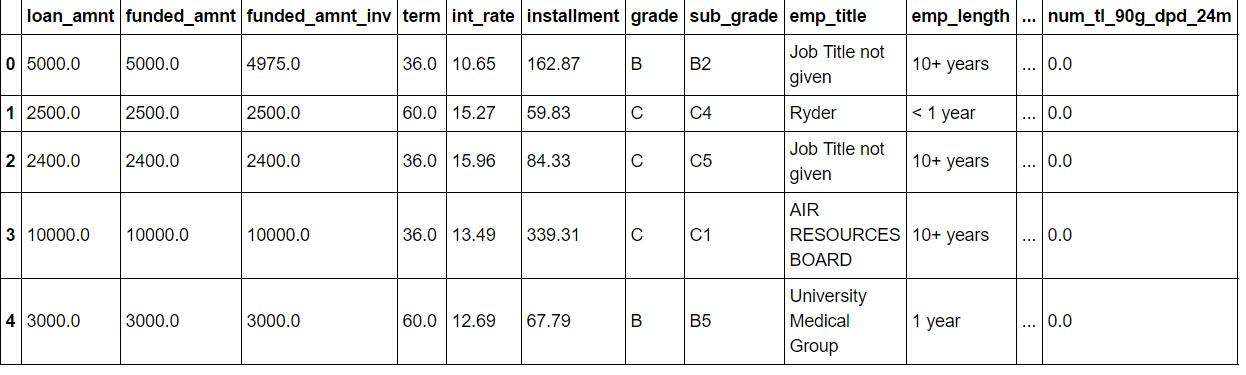
2016 Q3

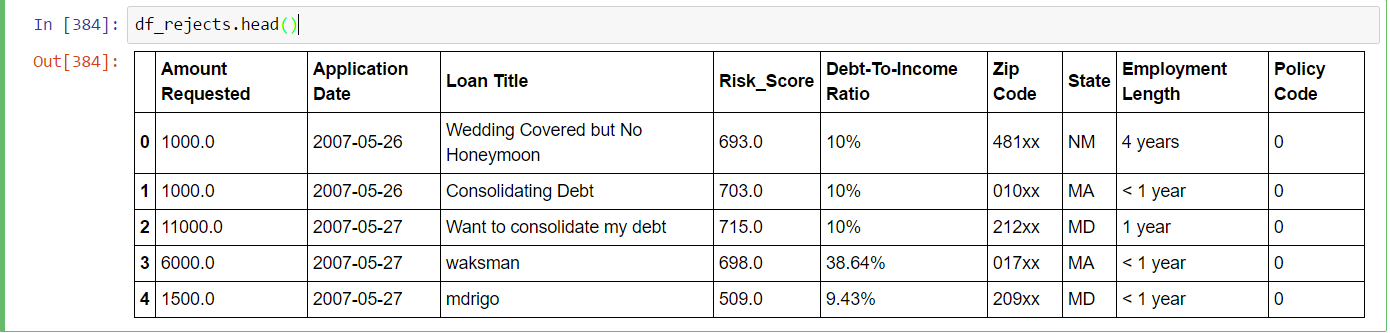
2016 Q4

The Dimensions for Loan Data: (1321848, 111)

Declined Loan Data: (11079386, 9)

For ex:





# Missing Data Handling

Loan Stats:

Missing data percentage ----

|  |  |
| --- | --- |
| id | 99 |
| member\_id | 100 |
| loan\_amnt | 0 |
| funded\_amnt | 0 |
| funded\_amnt\_inv | 0 |
| term | 0 |
| int\_rate | 0 |
| installment | 0 |
| grade | 0 |
| sub\_grade | 0 |
| emp\_title | 6.052146531 |
| emp\_length | 0 |
| home\_ownership | 0 |
| annual\_inc | 0 |
| verification\_status | 0 |
| issue\_d | 0 |
| loan\_status | 0 |
| pymnt\_plan | 0 |
| url | 100 |
| desc | 99 |
| purpose | 0 |
| title | 1.415502954 |
| zip\_code | 0 |
| addr\_state | 0 |
| dti | 0 |
| delinq\_2yrs | 0 |
| earliest\_cr\_line | 100 |
| inq\_last\_6mths | 0.000079039 |
| mths\_since\_last\_delinq | 49 |
| mths\_since\_last\_record | 83.88908034 |
| open\_acc | 0 |
| pub\_rec | 0 |
| revol\_bal | 0 |
| revol\_util | 0.058330514 |
| total\_acc | 0 |
| initial\_list\_status | 0 |
| out\_prncp | 0 |
| out\_prncp\_inv | 0 |
| total\_pymnt | 0 |
| total\_pymnt\_inv | 0 |
| total\_rec\_prncp | 0 |
| total\_rec\_int | 0 |
| total\_rec\_late\_fee | 0 |
| recoveries | 0 |
| collection\_recovery\_fee | 0 |
| last\_pymnt\_d | 0.09049924 |
| last\_pymnt\_amnt | 0 |
| next\_pymnt\_d | 46.13714468 |
| last\_credit\_pull\_d | 0.005374627 |
| collections\_12\_mths\_ex\_med | 0.018336964 |
| mths\_since\_last\_major\_derog | 74.86895394 |
| policy\_code | 0 |
| application\_type | 0 |
| annual\_inc\_joint | 99.30580365 |
| dti\_joint | 99.3061198 |
| verification\_status\_joint | 99.30580365 |
| acc\_now\_delinq | 0 |
| tot\_coll\_amt | 8.911843466 |
| tot\_cur\_bal | 8.911843466 |
| open\_acc\_6m | 71.81505907 |
| open\_il\_6m | 71.81498003 |
| open\_il\_12m | 71.81498003 |
| open\_il\_24m | 71.81498003 |
| mths\_since\_rcnt\_il | 72.57438326 |
| total\_bal\_il | 71.81498003 |
| il\_util | 75.48735224 |
| open\_rv\_12m | 71.81498003 |
| open\_rv\_24m | 71.81498003 |
| max\_bal\_bc | 71.81498003 |
| all\_util | 71.81632369 |
| total\_rev\_hi\_lim | 8.911843466 |
| inq\_fi | 71.81498003 |
| total\_cu\_tl | 71.81505907 |
| inq\_last\_12m | 71.81505907 |
| acc\_open\_past\_24mths | 7.311627216 |
| avg\_cur\_bal | 8.912791929 |
| bc\_open\_to\_buy | 8.217647115 |
| bc\_util | 8.271235311 |
| chargeoff\_within\_12\_mths | 0.018336964 |
| delinq\_amnt | 0 |
| mo\_sin\_old\_il\_acct | 11.64721262 |
| mo\_sin\_old\_rev\_tl\_op | 8.911922504 |
| mo\_sin\_rcnt\_rev\_tl\_op | 8.911922504 |
| mo\_sin\_rcnt\_tl | 8.911843466 |
| mort\_acc | 7.311627216 |
| mths\_since\_recent\_bc | 8.157182557 |
| mths\_since\_recent\_bc\_dlq | 76.86167606 |
| mths\_since\_recent\_inq | 16.97655082 |
| mths\_since\_recent\_revol\_delinq | 67.44216743 |
| num\_accts\_ever\_120\_pd | 8.911843466 |
| num\_actv\_bc\_tl | 8.911843466 |
| num\_actv\_rev\_tl | 8.911843466 |
| num\_bc\_sats | 7.988197951 |
| num\_bc\_tl | 8.911843466 |
| num\_il\_tl | 8.911843466 |
| num\_op\_rev\_tl | 8.911843466 |
| num\_rev\_accts | 8.911922504 |
| num\_rev\_tl\_bal\_gt\_0 | 8.911843466 |
| num\_sats | 7.988197951 |
| num\_tl\_120dpd\_2m | 12.48944834 |
| num\_tl\_30dpd | 8.911843466 |
| num\_tl\_90g\_dpd\_24m | 8.911843466 |
| num\_tl\_op\_past\_12m | 8.911843466 |
| pct\_tl\_nvr\_dlq | 8.923936377 |
| percent\_bc\_gt\_75 | 8.250606226 |
| pub\_rec\_bankruptcies | 0.211191239 |
| tax\_liens | 0.012013873 |
| tot\_hi\_cred\_lim | 8.911843466 |
| total\_bal\_ex\_mort | 7.311627216 |
| total\_bc\_limit | 7.311627216 |
| total\_il\_high\_credit\_limit | 8.911843466 |

We have removed the columns where values are nulls more than 50%. Those columns also don’t add much to our analysis.

For ex:

id

memberid

url

desc

mths\_since\_last\_record

After dropping, the dataset dimensions are**: (1321848, 86)**

For rest of the columns, the missing data percentage error is less except mths\_since\_last\_delinq. We can ignore those missing values but here we have imputed with mean, median or mode depending on data. We have also checked by creating models to predict the values.

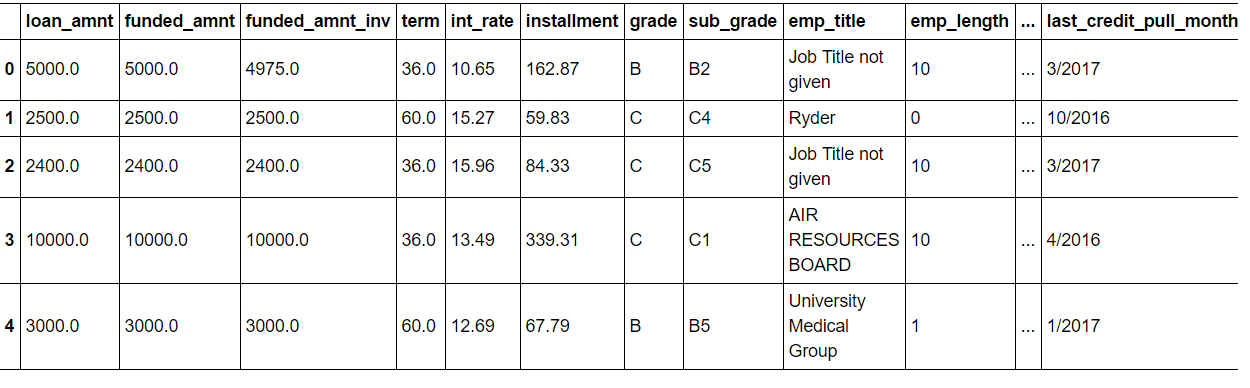
For ex:

For some date columns like--- issue\_d

df.issue\_d.fillna(df.issue\_d.mode(),inplace=True)

For mths\_since\_last\_delinq , its unknown whether it was delinquent anytime so just assume the max number of months since the account was delinquent. So we have imputed max for nulls in this column.

df['mths\_since\_last\_delinq'].fillna(df['mths\_since\_last\_delinq'].max(),inplace=True)



Rejects Stats:

|  |  |
| --- | --- |
| Amount Requested | 0 |
| Application Date | 0 |
| Loan Title | 0 |
| Risk\_Score | 57 |
| Debt-To-Income Ratio | 0 |
| Zip Code | 0 |
| State | 0 |
| Employment Length | 3 |
| Policy Code | 0 |

Risk\_Score: For applications prior to November 5, 2013 the risk score is the borrower's FICO score. For applications after November 5, 2013 the risk score is the borrower's Vantage score.

We tried building models for this, to see how the values varies.

But since this is a FICO score, we cannot impute it with any such value, so we will assume the credit history of those customers is not generated or somehow logged in the database.

For Employment Length: we have imputed 0 for nulls, as the missing data is very less, it won’t affect to change those few rows.

# Feature Engineering

The goal is to prepare and choose better features, to achieve better results. The idea is to “transform the data” from raw state to a state suitable for modelling where feature engineering fits in.

**Loan Stats:**

**Categorical features**

To handle categorical variables, we have not deleted any original column, but created derived columns for them. We have assignment numerical values as:

grade is categorical feature; we have derived grade\_num assigning integers to each grade so to have clarity when we want to build regression models with continuous features

#Cleaning Grade, df\_loan is the dataframe

df\_loan['grade\_num'] = df\_loan['grade'].map({'A':7,'B':6,'C':5,'D':4,'E':3,'F':2,'G':1})

Same logic is applied for some other columns like home\_ownership.

This way, we have created following columns in the dataset.

grade\_num

sub\_grade\_num

home\_ownership\_num

verification\_status\_num

pymnt\_plan\_clean

purpose\_num

application\_type\_num

loan\_status\_num

**Decompose Date-time**

The datatype of date is string here in the raw dataset.

Dec – 2011

We have changed the datatype to date format as: 12/2011

Next, we have derived two features out of it : Month & Year to give flexibility to user to build relationships.

Following columns have been created:

issue\_d\_year

issue\_d\_month

earliest\_cr\_line\_year

earliest\_cr\_line\_month

last\_pymnt\_month

last\_pymnt\_year

last\_pymnt\_monthYear

next\_pymnt\_month

next\_pymnt\_year

next\_pymnt\_monthYear

last\_credit\_pull\_month

last\_credit\_pull\_year

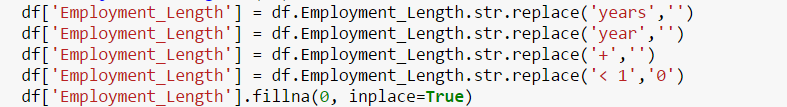
last\_credit\_pull\_monthYear

**Transform raw features to useful ones.**

Here we have cleaned some of the features to get more useful features.

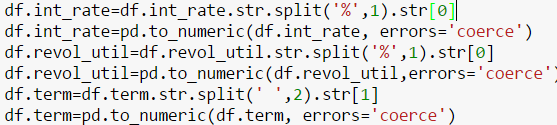
Cleaning **emp\_length** to remove +,<….

Employment length is in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years. So we have changed accordingly, But again created a new feature and original column is left as it is.



**int\_rate,** revold\_util we have removed % from the interest rate values.

In term column, we have only two values: 36 months and 60 months, we have removed months from that.



**Declined Loan Data : Rejects Loans**

First, we have renamed the columns to give meaningful and proper case names:

'Amount Requested': 'Amount\_Requested'

'Employment Length': 'Employment\_Length'

'Loan Title': 'Loan\_Title'

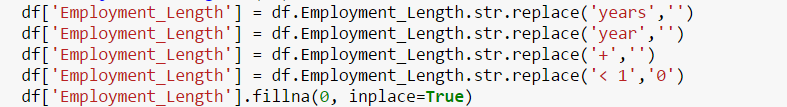
'Application Date': 'Application\_Date'

'Policy Code': 'Policy\_Code'

'Zip Code': 'Zip\_Code'

'Debt-To-Income Ratio': 'Debt-To-Income-Ratio'

Next, transform the features to get meaningful values:



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

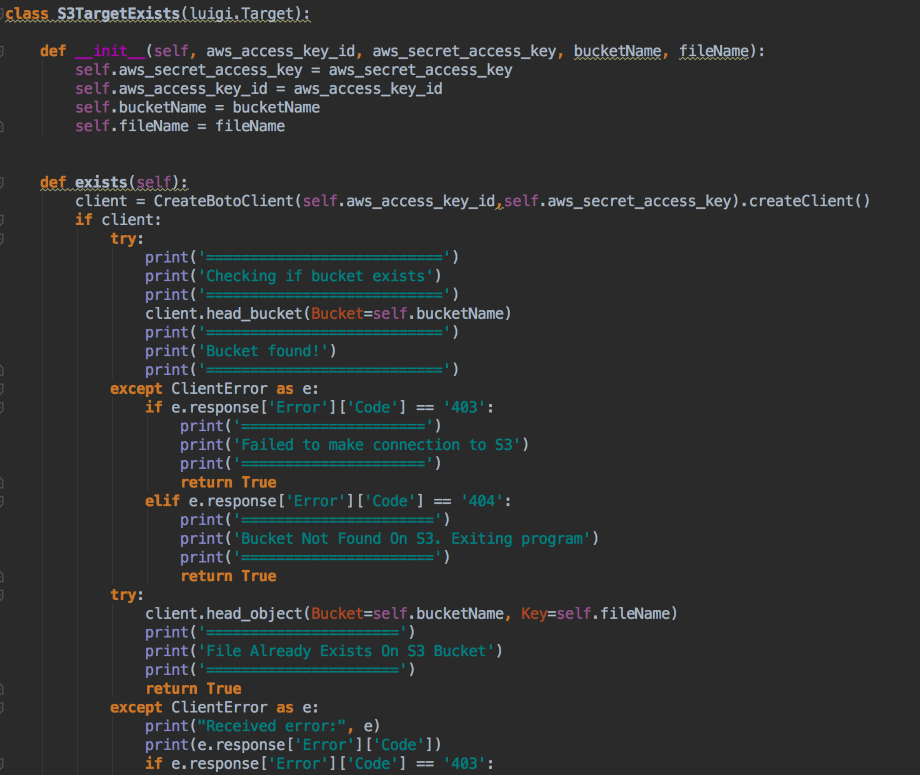
# Luigi

# Luigi Tasks:

We have created the below task with targets on S3 bucket:

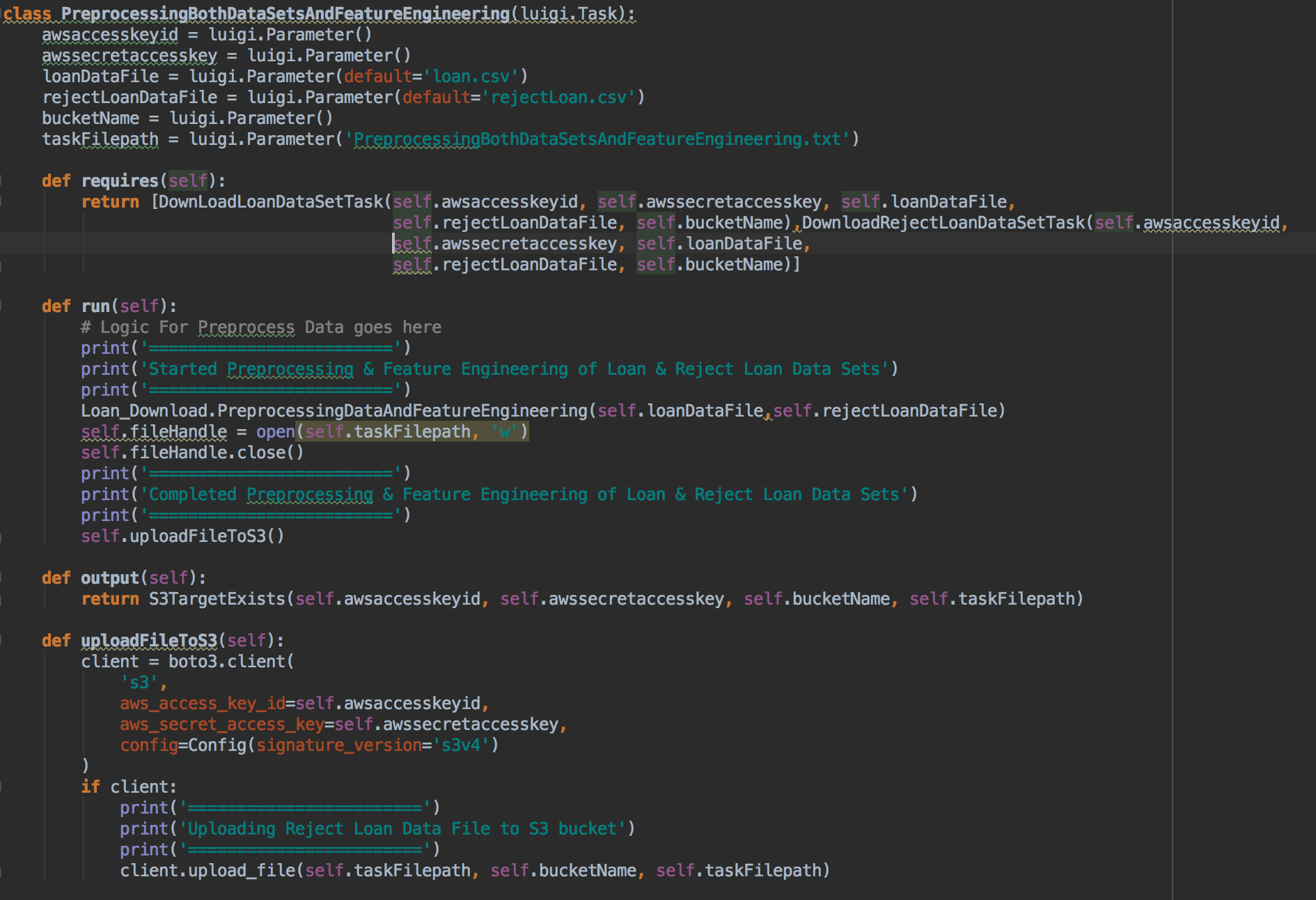
1. DownLoadLoanDataSetTask
2. DownloadRejectLoanDataSetTask
3. PreprocessingBothDataSetsAndFeatureEngineering
4. SummarizationTask
5. UploadLoanDataToS3
6. UploadRejectLoanDataToS3
7. UploadSummaryFilesToS3

Create a custom target class to check Target on S3(This will check for connection errors, bucket errors):



Tasks:

Almost all the tasks are similar except changes in run method of each task which runs its respective methods:



1. The above task first checks if target exists, if yes this task will not run.
2. If target doesn’t exist on S3 then this task will run requires method and calls the dependent tasks and makes sure that they are complete.
3. Finally run method is called, in run method task related main code will be executed and a flag file will be uploaded into S3.

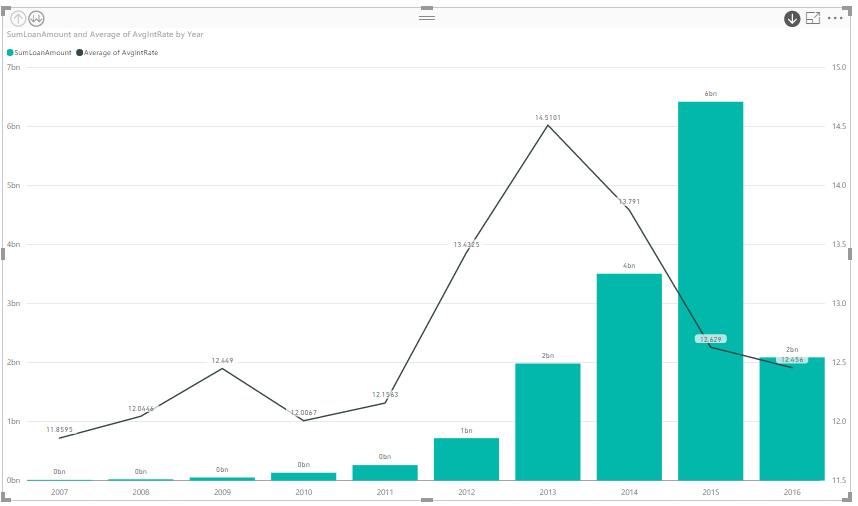
# Summarizations

## Loan Dataset

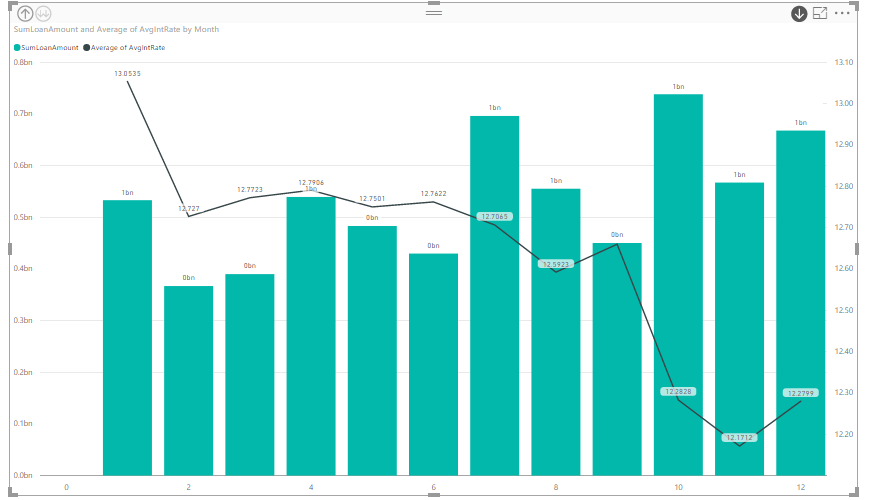
### Power BI Graphs:

##### Summarization by Year and Month

Total Loan Amount and Average Interest Rate per year:



This can be further drilled down to monthly analysis for selected year:

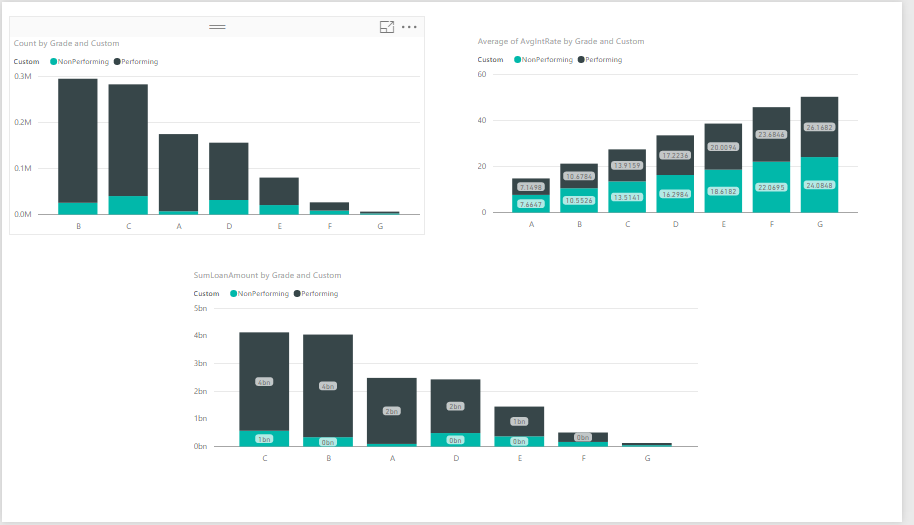


##### Summarization by Grade and Loan Status

Here we have done following mapping:

|  |  |
| --- | --- |
| Loan Status | Mapping |
| Fully Paid | Performing |
| Current | Performing |
| All other loan status | Non-Performing |

After this mapping we found the total loan amount, total count of loans and average interest rate by grade and Mapping:



##### Summarization by Home Ownership

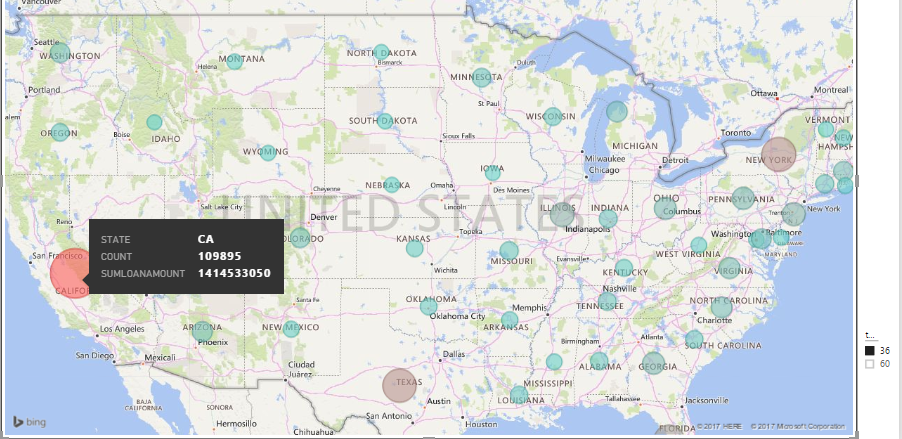
In this graph, the percentage contribution of each type of home ownership as well as total loan amount and average interest rate is displayed:



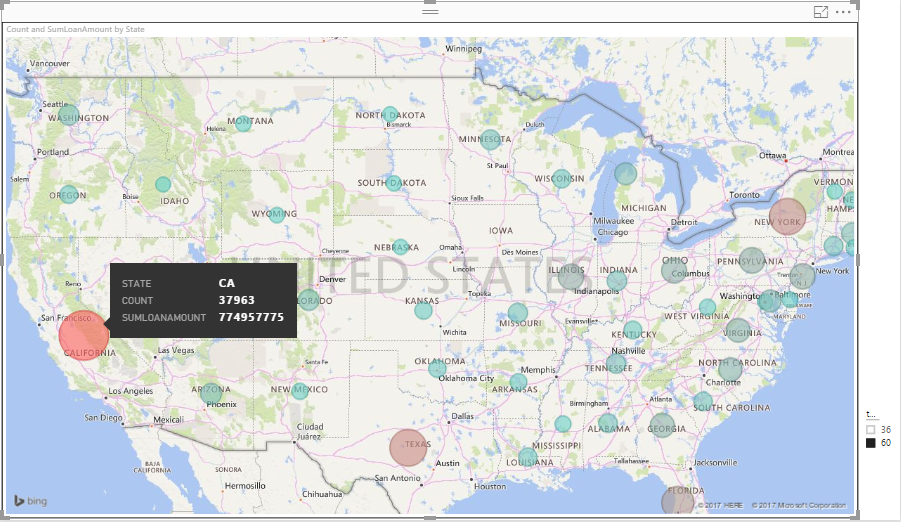
##### Summarization by location

Total count of loans and loan amount by state, filter is set on “Term”.

Term = 36



Term = 60

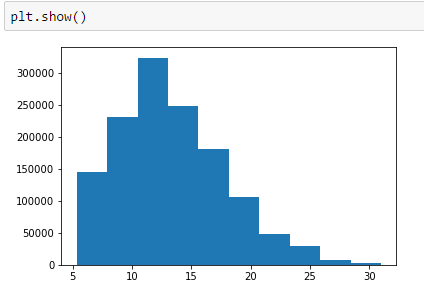


In both cases, highest is noticed in California.

### Python Graphs

##### Frequency distribution of interest rate

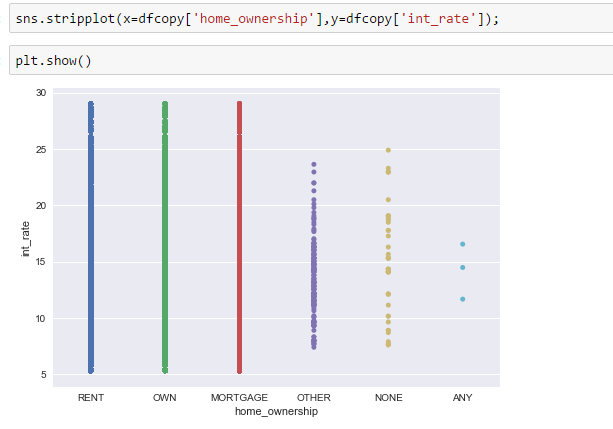




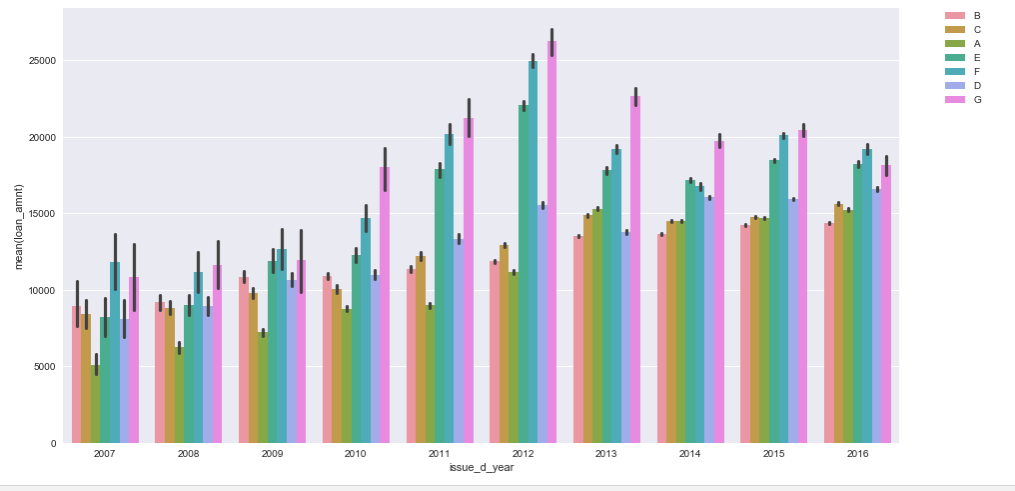
##### Interest rate by grade and term



##### Distribution of interest as per type of home ownership

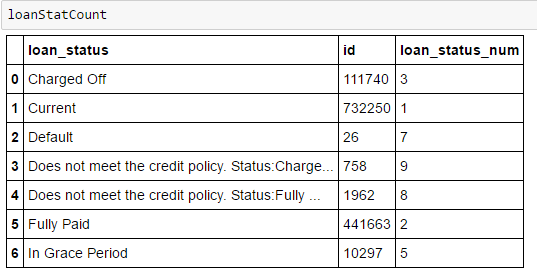


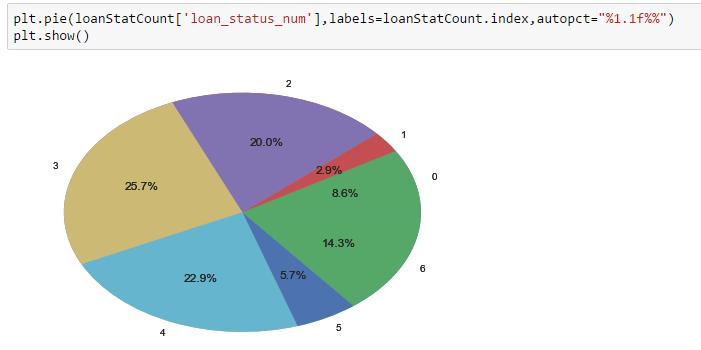
##### Average loan amount by year and grade



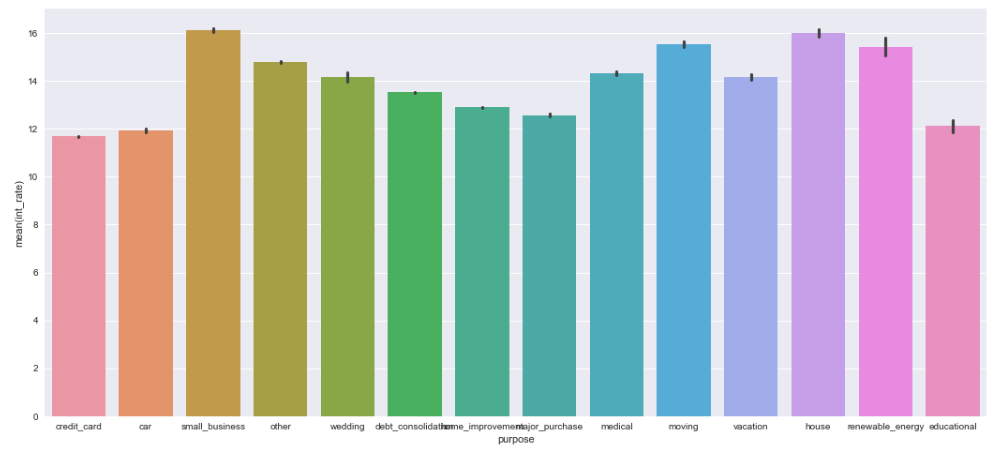
##### Percentage of different types of loan status

We have factorized the loan status and then plotted the pie chart.

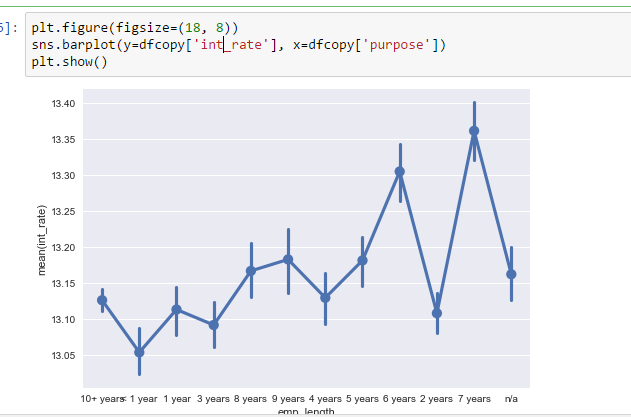




##### Variation of interest rate by purpose of loan



##### Mean Interest Rate by employee length



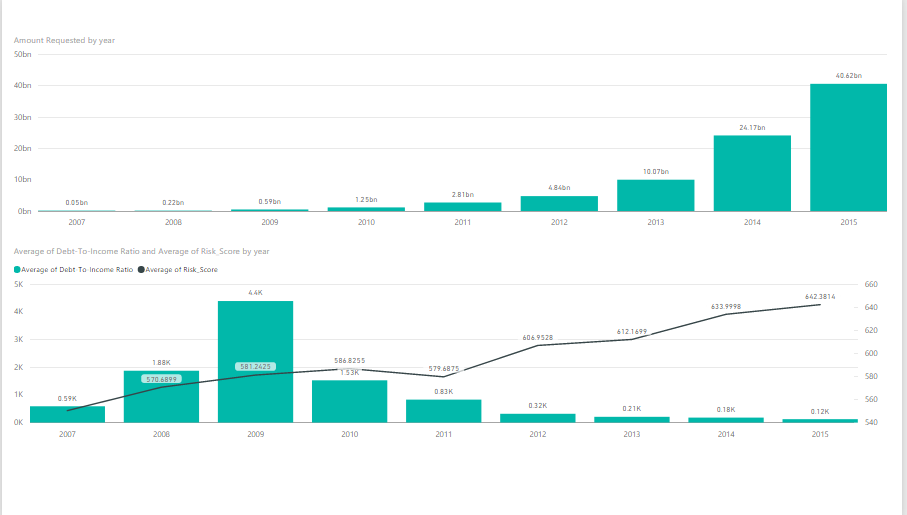
## Reject Dataset

### Power BI graph

##### Total Amount Requested, Average DTI ratio and average risk score by year and month

Here we have created hierarchy with year and month:

By Year:



On selecting a particular year and drill down the hierarchy, data will be displayed for all months of that year:



As we can see the one of the reason behind rejecting these loan requests can be very high DTI ratio and moderate Risk Score.

# GitHub Links:

Github Repo: <https://github.com/Yamini-S/ADS/tree/master/Assignment02>

# Docker hub links:

**Link:** <https://hub.docker.com/r/melavoyrajap/ads_assignment2/>

**Repo:** melavoyrajap/ads\_assignment2

**Tag:** latest

# Run Docker Image

Commands:

1. **Pull the docker image from docker hub**:

docker pull melavoyrajap/ads\_assignment2

1. **Run the Image:** Copy the command below and make necessary changes as instructed:

|  |
| --- |
| docker run -it -e "accesskey=<enter your accesskey>" -e "secretkey=<enter your secretkey>" -e "bucket=<enter your bucket>" -e "loanDataFile=<enter filename>"  -e "rejectLoanDataFile=<enter filename>" melavoyrajap/ads\_assignment2:latest |

Please enter your Amazon S3 credentials: **accesskey, secretkey and bucket**.

For **loanDataFile** and **rejectLoanDataFile** are filenames to be given by user of his choice, and the files will be downloaded by this name to S3 bucket.

For ex:

docker run -it -e "accesskey=XXXXXXXXXXXXXXXXXXX" -e "secretkey=XXXXXXXXXXXXXXX" -e "bucket=adsdoc" -e "loanDataFile=LoanData.csv" -e "rejectLoanDataFile=rejectsLoan.csv" melavoyrajap/ads\_assignment2:latest

Here, the accesskey and secretkey are just used for example, these are not valid, and bucket is also just a random name to use in this example. Use your credentials in the command.

For environmental variable: **accesskey, secretkey, bucket**, **loanDataFile and rejectLoanDataFile**. These are case sensitive, just use as given in the command.

**Note**: There should be no space in your credentials, otherwise the code will give error message to enter correct credentials.

Please do not remove double quotes (“accesskey=<enter your accesskey>”) also.

For example, following command is invalid:

docker run -it -e accesskey= XXXXXXXXXXXX -e secretkey= XXXXXXXXXXXXXX -e bucket= TEST loanDataFile=LoanData.csv -e rejectLoanDataFile=rejectsLoan.csv melavoyrajap/ads\_assignment2:latest

Following files are downloaded:

DownLoadLoanDataSetTask.txt

DownloadRejectLoanDataSetTask.txt

PreprocessingBothDataSetsAndFeatureEngineering.txt

RejectSummary.csv

SummarizationTask.txt

SummaryStatistics.csv

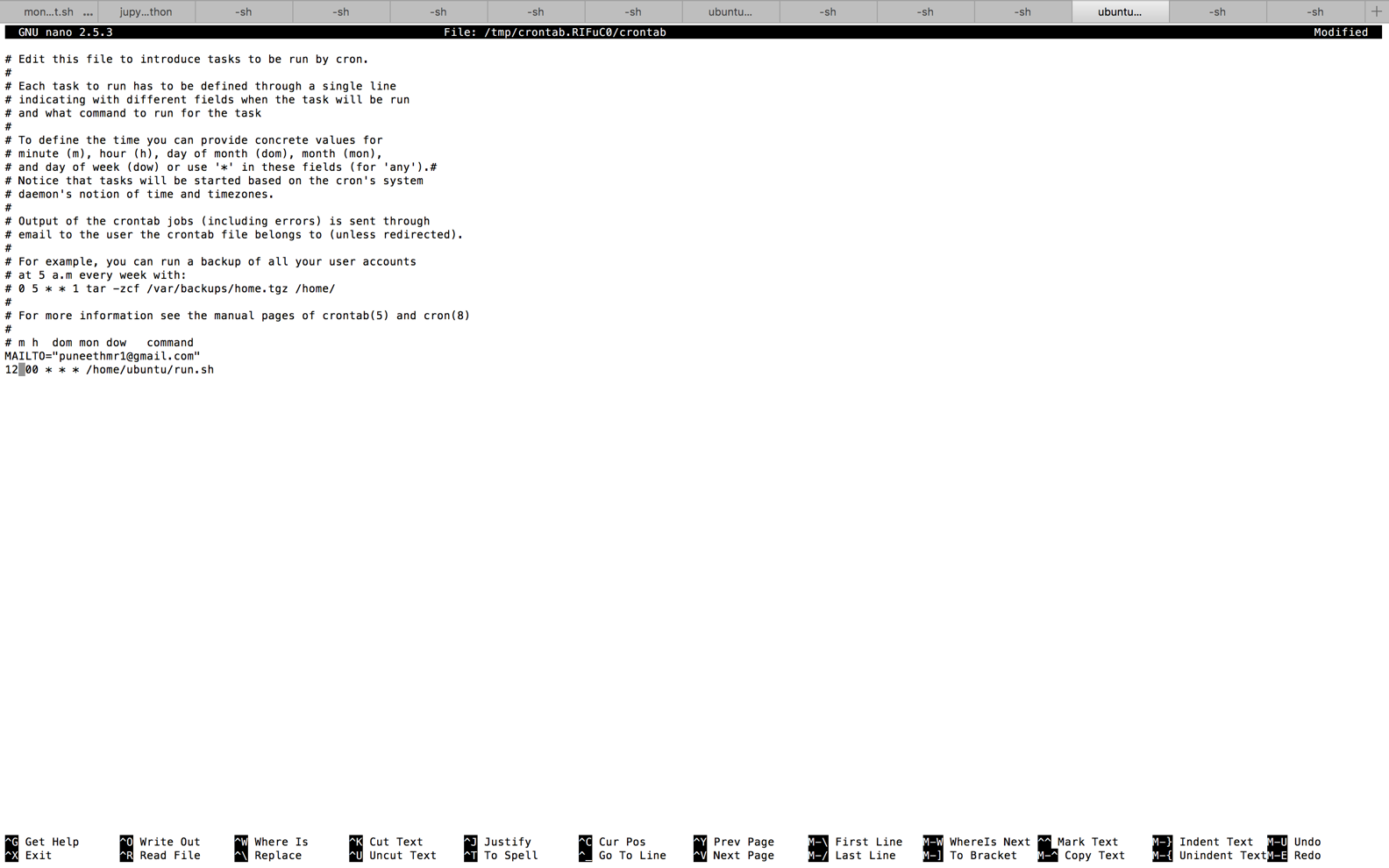
loanDataFileDocker2.csv

rejectLoanDataFileDocker2.csv

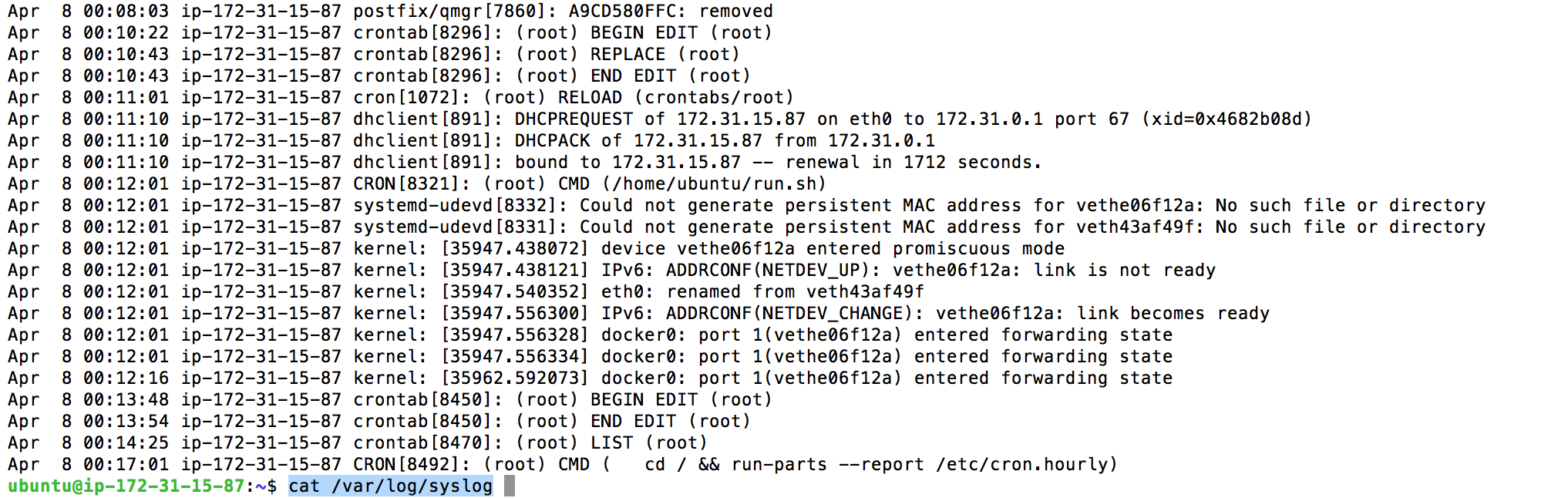
**NOTE**: Please remove all the downloaded files from your bucket to run the docker image for the second time.

# Scheduling Cron Job to run the docker image:

* Created a bash script to be executed every day at particular time.
* Install POSTFIX (sudo apt-get postfix)
* Below is the job scheduling config (sudo crontab -e):



* Output of the cron job is configured to go into out.log on the home directory.
* Need to change permissions on run.sh and out.log (chmod -777 run.sh out.log)
* Run.sh and out.log are added in the git repo.
* Please find the screen shot of log entries(cat /var/log/syslog):
  + In the below screen shot at 00:12:01 cron job got executed



* Please find screen shot of file uploaded to S3 after running CRON Job:

