FLIP ROBO

Micro Credit Loan Prediction

Submitted By-
Akanksha Amarnani

Acknowledgement

I would like to thank Almighty for giving me the confidence to pursue this project. Further, the concepts from DataTrained Academy guided me to complete the project.
In addition I would like to thank my mentor from Flip Robo Technology, Ms Khushboo Garg for clarifying my doubts and queries.

The references used for the completion of this project are-

- A Machine Learning Approach for Micro-Credit Scoring; MDPI., Apostolos Ampountolas , Titus Nyarko Nde , Paresh Date and Corina Constantinescu
- Predicting Default loans using Machine Learning; 27th Telecommunications forum TELFOR 2019, Serbia, Belgrade, November 26-27, 2019.; Zoran Ereiz, Member, IEEE
- Rural Micro Credit Assessment using Machine Learning in a Peruvian microfinance institution; Henry Ivan Condori-Alejoa , Miguel Romilio Aceituno-Rojoa , Guina Sotomayor Alzamoraa; Science Direct, Procedia Computer Science 187 (2021) 408–413

INTRODUCTION

Business Problem Framing-

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. Many microfinance institutions (MFI), experts and donors are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient, and cost saving, than the traditional high-touch model used since long for the purpose of delivering microfinance services.

We are working with one such client that is in Telecom Industry. They are a fixed wireless telecommunications network provider. They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah).

Conceptual Background of the Domain Problem-

The domain related concepts which help us in a better understanding are-

- Exploratory Data Analysis (EDA)- By conducting explanatory data analysis, we obtain a better understanding of our data. This yields insights that can be helpful later when building a model, as well as insights that are independently interesting.

- Splitting the data- The dataset is split into train and test sample using the train test split

- Downsampling the data- As the dataset is imbalanced, downsampling it is required

- Modeling- We apply Logistic Regression, KNeighbor Classifier, Decision Tree Classifier, Random Forest Classifier, Adaboost Classifier, Gradient Boosting Classifier and SVC to ckeck is the user ia a defaulter or not

- Regularization- Models are regularized and the parameters are hypertuned to enhance the efficiency of the models

Review of Literature-

Machine learning is a subfield of Artificial Intelligence (AI) that works with algorithms and technologies to extract useful information from data. Machine learning methods are appropriate in big data since attempting to manually process vast volumes of data would be impossible without the support of machines. Machine learning in computer science attempts to solve problems algorithmically rather than purely mathematically. Therefore, it is based on creating algorithms that permit the machine to learn. However, there are two general groups in machine learning which are supervised and unsupervised. Supervised is where the program gets trained on pre-determined set to be able to predict when a new data is given. Unsupervised is where the program tries to find the relationship and the hidden pattern between the data

The performance of the model build will be measured upon its accuracy to determine whether a user is a defaulter or not, whereby, Label '1' indicates that the loan has been paid i.e. Non- defaulter, while, Label '0' indicates that the loan has not been paid i.e. defaulter. The features assessed will be such to track down the history of the user, his/her mobile service utilization, his/her credibility and his/her credit balance. We implement and evaluate various learning methods on the provided dataset. However, proper EDA is to be kept in mind and the balancing of dataset is a necessity to avoid unbiased predictions. The data used in the experiment will be handled by using a combination of pre-processing methods to improve the prediction accuracy

Motivation for the Problem Undertaken-

The project involves a Microfinance Institution (MFS). MFS have turned out be very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on. Microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients. Such a utility of microfinance which become accessible to every class of the society, automatically brings in the motivation to continue with such a socio-financial inclined project. However, predicting the status if a user is a defaulter or not by random is not possible. Hence a data analyst, it seems to be my responsibility to add the element of possibility in every random scenario, to solve the cumbersome unsure process into a more reliable, dependent matter. Further, every project has a lot to offer as well. The project and its attributes imparted a lot of knowledge about the finance and telecommunication system.

EDA Steps and Visualization

- The provided sample data from the client database and saved in a dataframe
- The shape of the dataframe is checked-

There are 209593 rows and 37 columns

- **The columns are as follows-**
  - Unnamed: 0
  - Label
  - Msisdn
  - Aon
  - daily_decr30
  - daily_decr90
  - Rental30
  - Rental90
  - last_rech_date_ma
  - last_rech_date_da
  - last_rech_amt_ma
  - cnt_ma_rech30
  - fr_ma_rech30
  - sumamnt_ma_rech30
  - medianamnt_ma_rech30
  - Medianmarechprebal30
  - cnt_ma_rech90
  - fr_ma_rech90
  - sumamnt_ma_rech90
  - medianamnt_ma_rech90
  - Medianmarechprebal90
  - cnt_da_rech30
  - fr_da_rech30
  - cnt_da_rech90
  - fr_da_rech90
  - cnt_loans30
  - amnt_loans30
  - maxamnt_loans30
  - medianamnt_loans30
  - cnt_loans90
  - amnt_loans90
  - maxamnt_loans90
  - medianamnt_loans90
  - Payback30
  - Payback90
  - Pcircle
  - pdate

The data type of each column is-

- Unnamed: 0 - int64
- label - int64
- msisdn - object
- aon - float64
- daily_decr30 - float64
- daily_decr90 - float64
- rental30 - float64
- rental90 - float64
- last_rech_date_ma - float64
- last_rech_date_da - float64
- last_rech_amt_ma - int64
- cnt_ma_rech30 - int64
- fr_ma_rech30 - float64
- sumamnt_ma_rech30 - float64
- medianamnt_ma_rech30 - float64
- medianmarechprebal30 - float64
- cnt_ma_rech90 - int64
- fr_ma_rech90 - int64
- sumamnt_ma_rech90 - int64
- medianamnt_ma_rech90 - float64
- medianmarechprebal90 - float64
- cnt_da_rech30 - float64
- fr_da_rech30 - float64
- cnt_da_rech90 - int64
- fr_da_rech90 - int64
- cnt_loans30 - int64
- amnt_loans30 - int64
- maxamnt_loans30 - float64
- medianamnt_loans30 - float64
- cnt_loans90 - float64
- amnt_loans90 - int64
- maxamnt_loans90 - int64
- medianamnt_loans90 - float64
- payback30 - float64
- payback90 - float64
- pcircle - object
- pdate - object

The null values are checked. The whitespaces, and dashes ('—') are replaced by null values-

There are no null values

As Unnamed: 0 is the index value, it is safe to drop it

As msisdn is the mobile value, and every person has a unique mobile number, it is recommended to drop the column

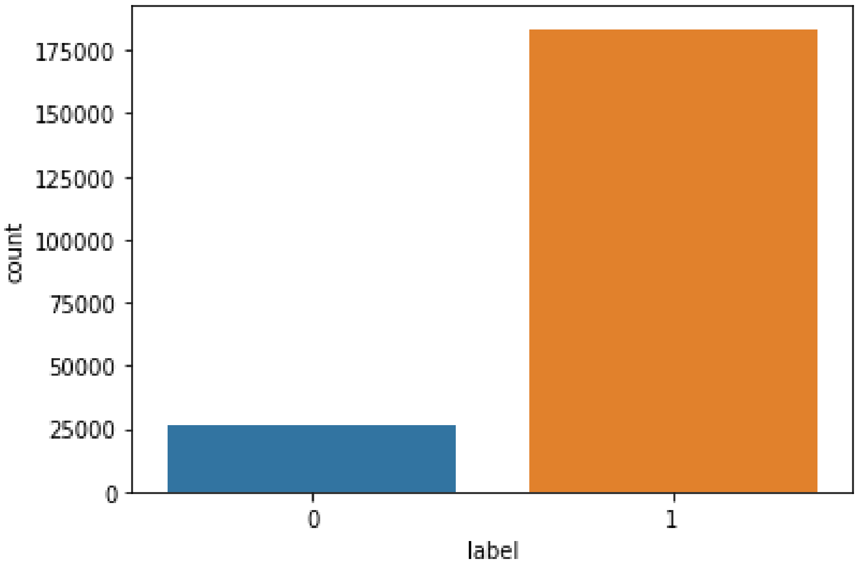
As telecom circle is same for all, i.e, UPW, it is recommended to drop the column

As pdate lies between 3 months of 2016, it is recommended to drop the column

The data visualization, value counts and encoding for each column

Label

Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}



- 183431 customers are Non-defaulters
- 26162 customers are Defaulters

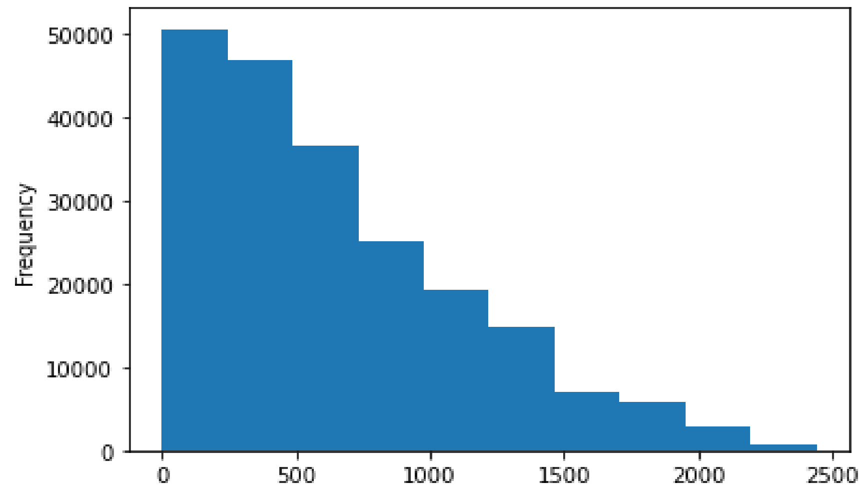
The dataset is imbalanced and will be down-sampled later

Aon

age on cellular network in days

Considering the company is 25years old, the age on cellular network in days should be below 9130 days. Hence replacing age >9130 with the mean of the column

As number of days should be preferably in whole numbers, hence converted to int datatype

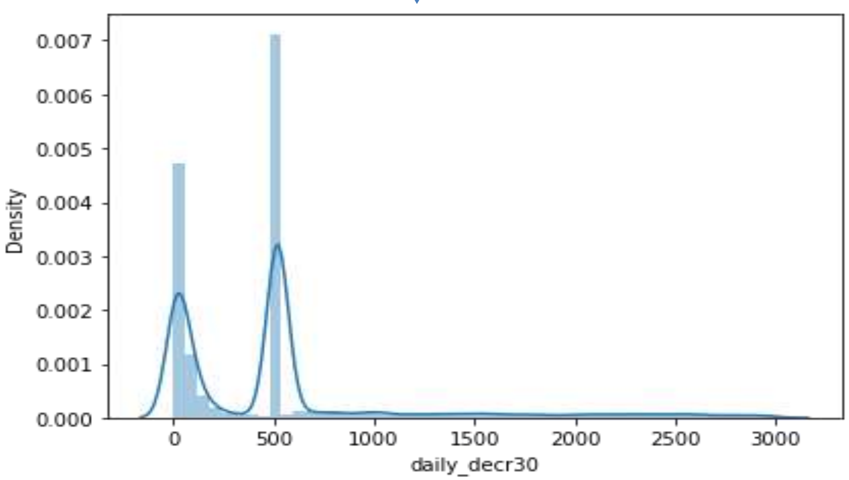


- Majority customers have 660 days age on cellular network

daily_decr30

Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)

Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah) should not be more than 3000. Hence replacing amount >3000 with the mean of the column

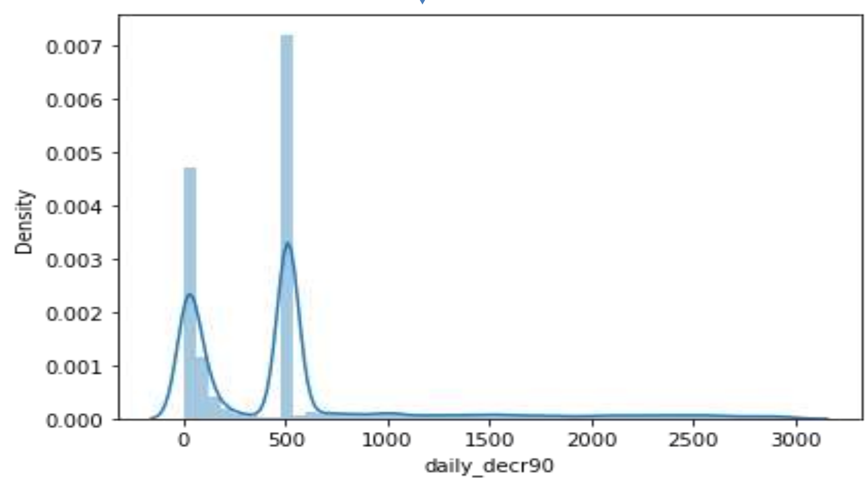


- Majority customers have daily spent amount as 0 or 520

daily_decr90

Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)

Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah) should not be more than 3000. Hence replacing amount >3000 with the mean of the column

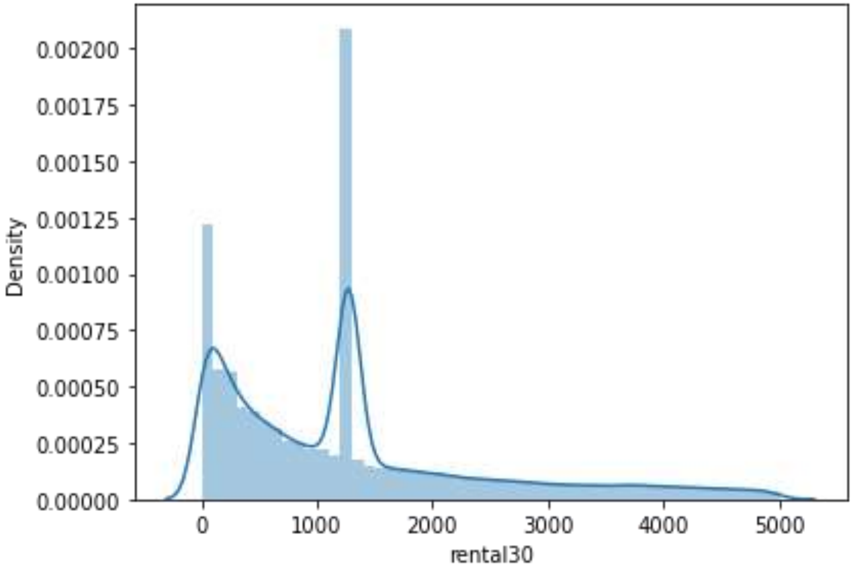


- Majority customers have daily spent amount as 0 or 511

Rental30

Average main account balance over last 30 days

Average rental should not be <0 and should not be greater than 5000. Hence replacing absurd values with the mean of the column

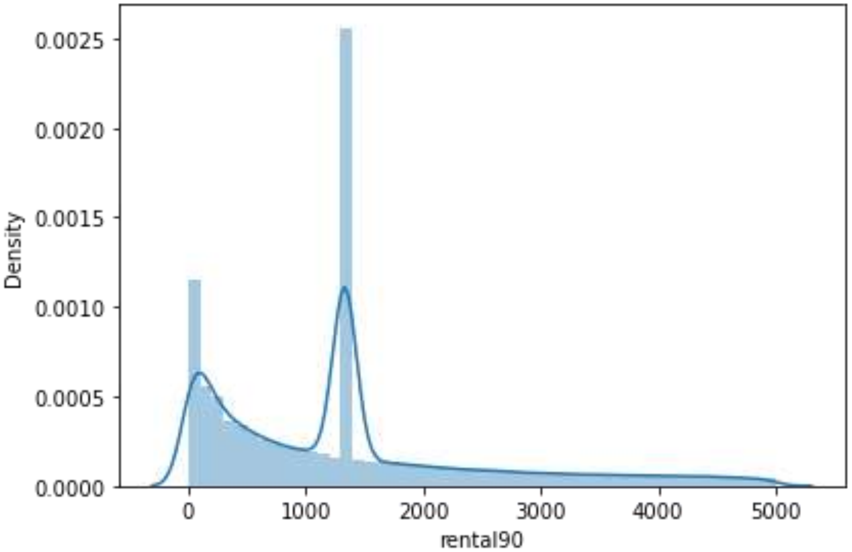


- Majority customers have average rental as 0 or 1272

Rental90

Average main account balance over last 90 days

Average rental should not be <0 and should not be greater than 5000. Hence replacing absurd values with the mean of the column



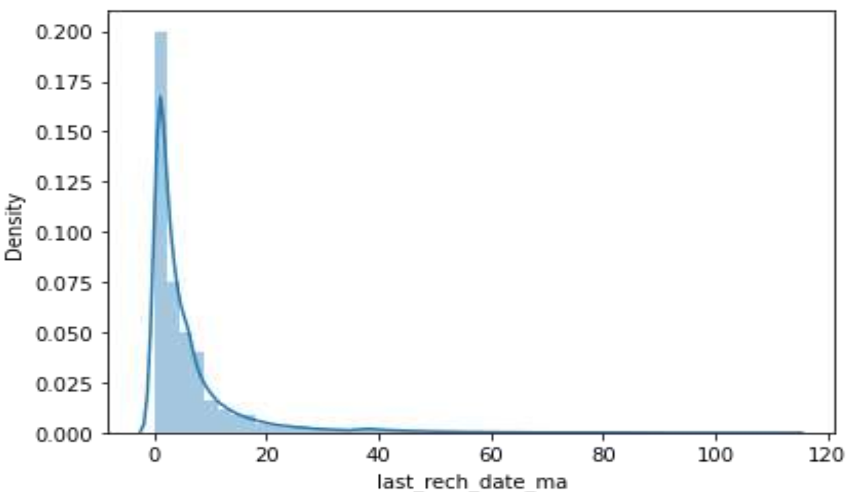
- Majority customers have average rental as 0 or 1332

last_rech_date_ma

Number of days till last recharge of main account

Last recharge date should be as old as the company is, approx, 25 years, i.e., 9130 days. Hence replacing age >9130 with the mean of the column

As number of days should be preferably in whole numbers, hence converted to int datatype



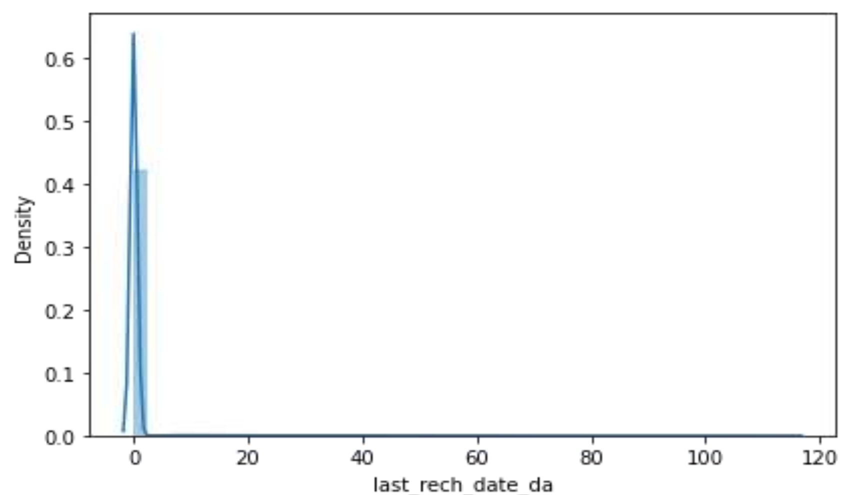
- Majority customers have 0-4 days till the last recharge

last_rech_date_da

Number of days till last recharge of data account

Last recharge date should be as old as the company is, approx, 25 years, i.e., 9130 days. Hence replacing age >9130 with the mean of the column

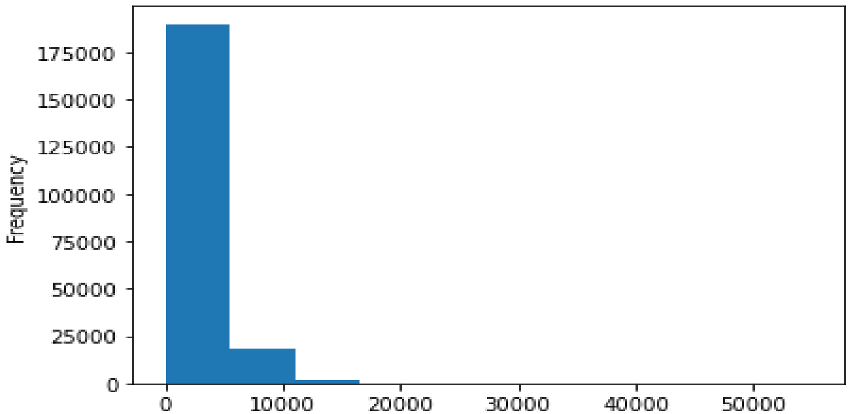
As number of days should be preferably in whole numbers, hence converted to int datatype



- Majority customers have 0 days till the last recharge

last_rech_amt_ma

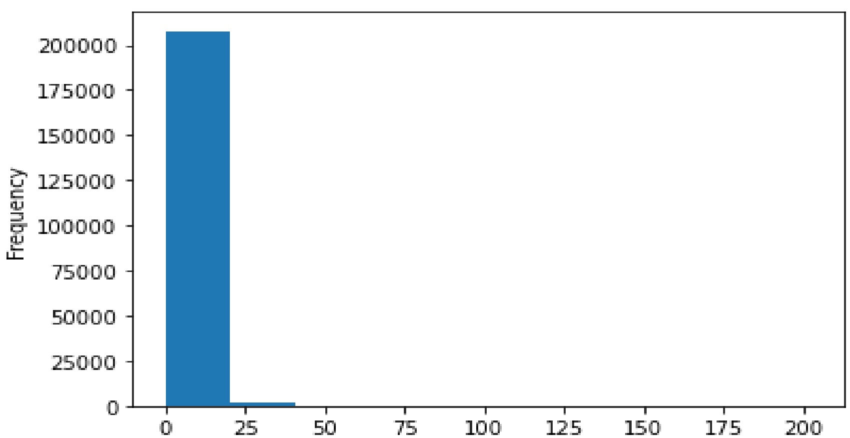
Amount of last recharge of main account (in Indonesian Rupiah)



- Majority customers have last recharge amount of main account as 0 or 2300

cnt_ma_rech30

Number of times main account got recharged in last 30 days

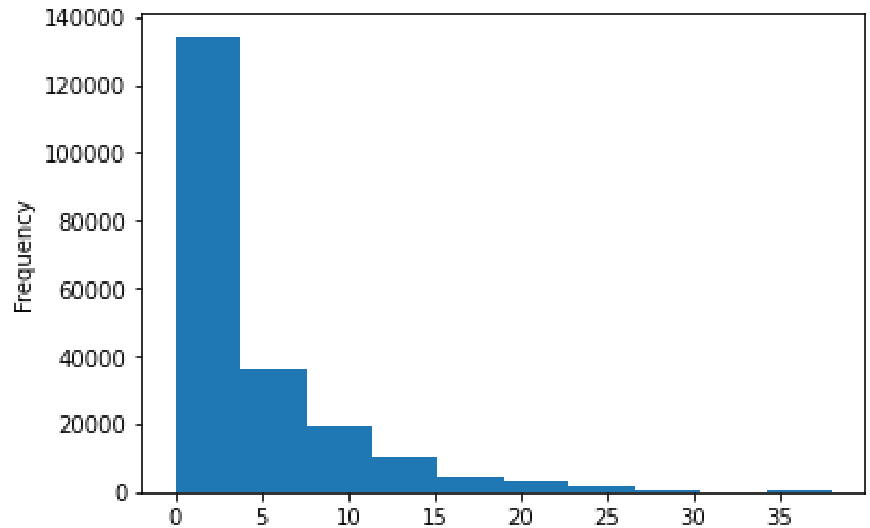


- Majority customers have recharged main account 0-4 days in last 30 days

fr_ma_rech30

Frequency of main account recharged in last 30 days

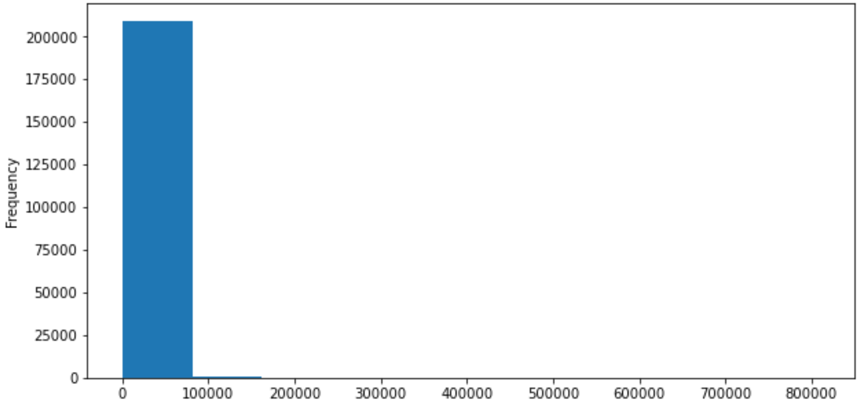
Frequency of times data account got recharged in last 30 days should not be more than 3000. Hence replacing frequency >3000 with the mean of the column



- Majority customers have main account recharge frequency between 1-15 times

sumamnt_ma_rech30

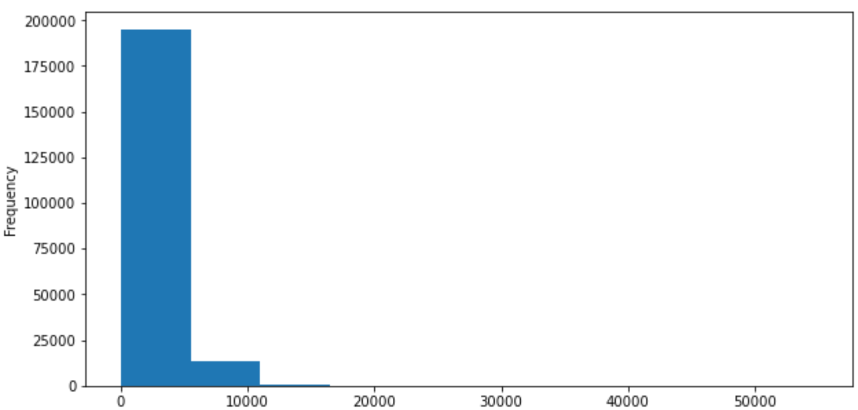
Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)



- Majority customers have total recharge amount of main account of 0 Rupiah

medianamnt_ma_rech30

Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)

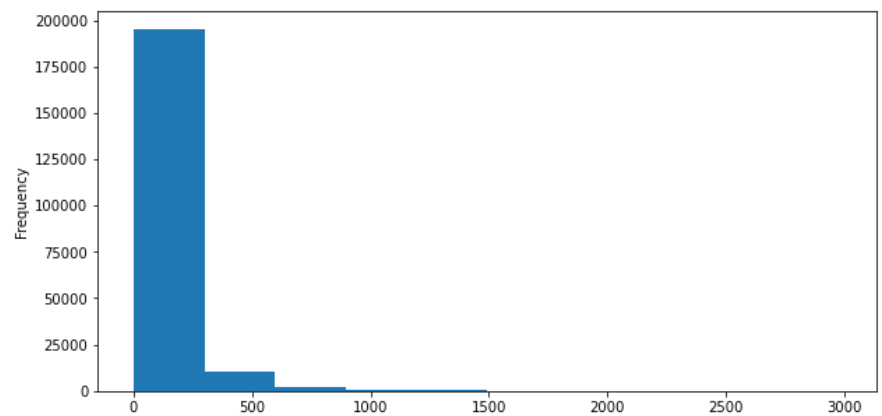


- Majority customers median of amount of recharges done in main account in last 30 days between 0-6000

medianmarechprebal30

Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)

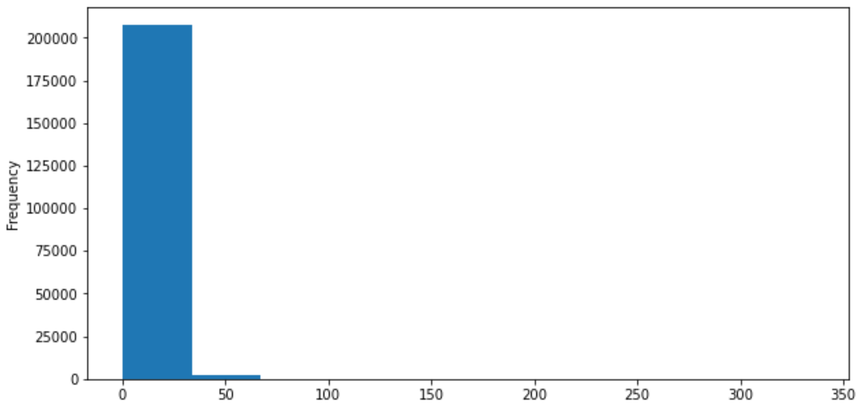
Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah) should not be more than 3000. Hence replacing amount >3000 with the mean of the column



- Majority customers have Median of main account balance just before recharge in last 30 days at user level is between 0-250

cnt_ma_rech90

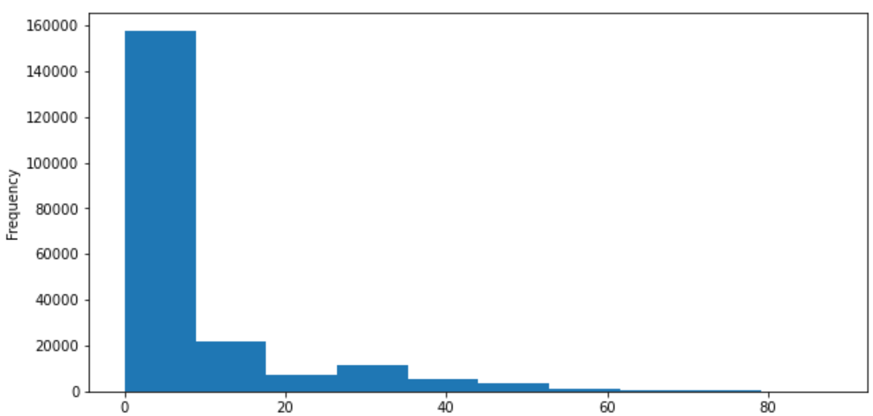
Number of times main account got recharged in last 90 days



- Majority customers have recharged main account 0-4 times in last 90 days

fr_ma_rech90

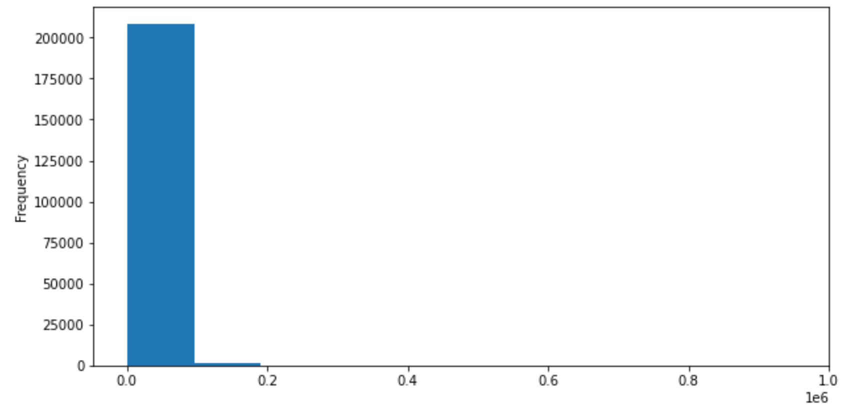
Frequency of main account recharged in last 90 days



- Majority customers have main account recharge frequency between 1-15 times

sumamnt_ma_rech90

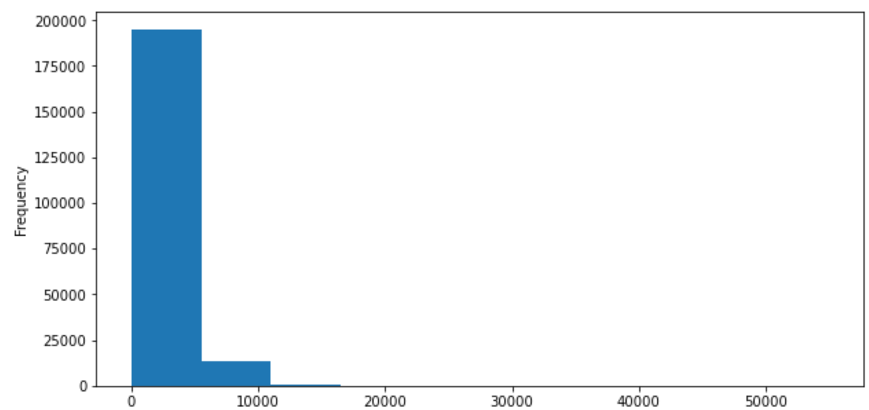
Total amount of recharge in main account over last 90 days (in Indonesian Rupiah)



- Majority customers have total recharge amount of main account of 0 Rupiah

medianamnt_ma_rech90

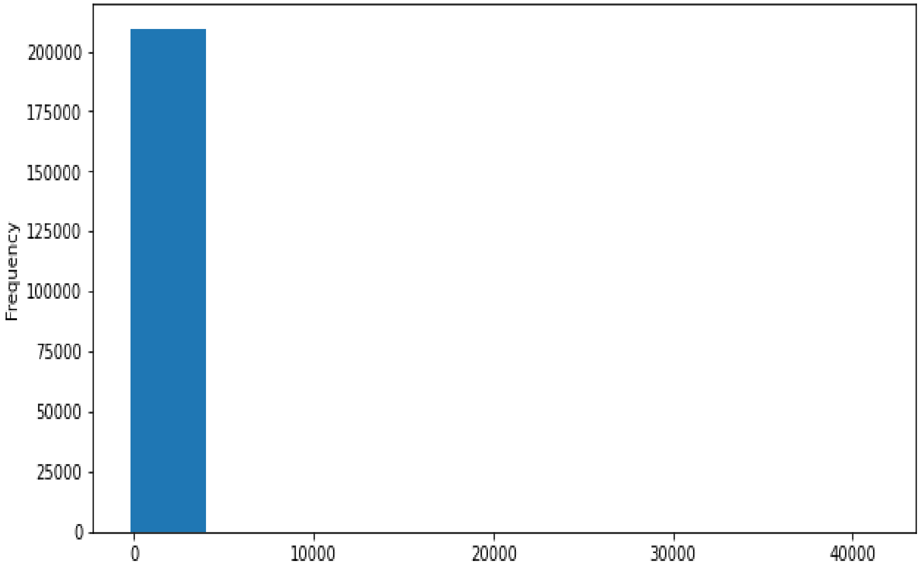
Median of amount of recharges done in main account over last 90 days at user level (in Indonesian Rupiah)



- Majority customers median of amount of recharges done in main account in last 90 days between 0-6000

medianmarechprebal90

Median of main account balance just before recharge in last 90 days at user level (in Indonesian Rupiah)



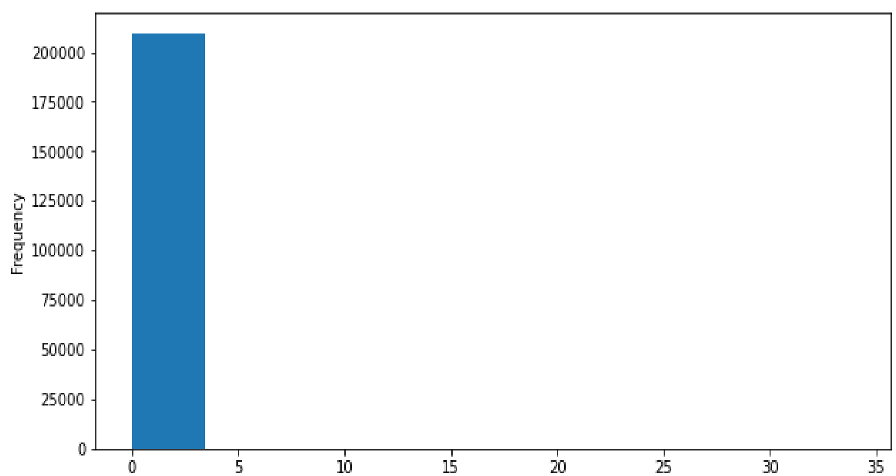
- Majority customers have Median of main account balance just before recharge in last 90 days at user level is between 0-250

cnt_da_rech30

Number of times data account got recharged in last 30 days

Number of times data account got recharged in last 30 days should not be more than 3000. Hence replacing number of times >3000 with the mean of the column

As number of times should be preferably in whole numbers, hence converted to int datatype

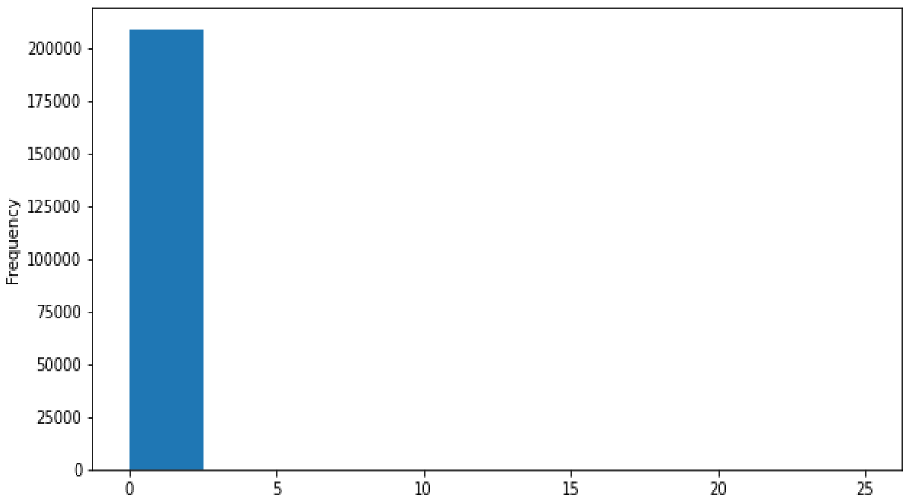


- Majority of customers have recharged their data account 0-1 time

fr_da_rech30

Frequency of data account recharged in last 30 days

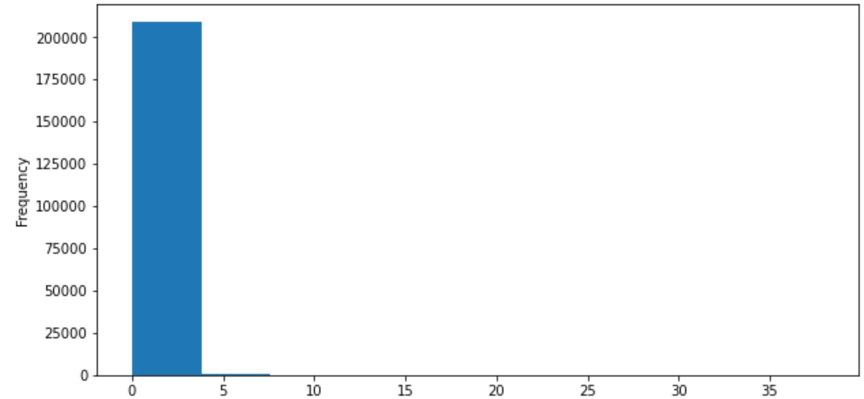
Frequency of times data account got recharged in last 30 days should not be more than 3000. Hence replacing frequency >3000 with the mean of the column



- Majority customers have data account recharge frequency of 0 times

cnt_da_rech90

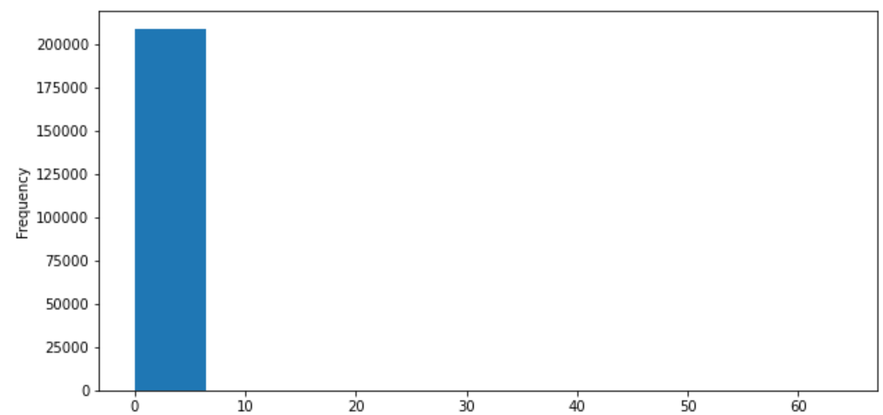
Number of times data account got recharged in last 90 days



- Majority customers have recharged data account 0-1 times in last 90 days

fr_ma_rech90

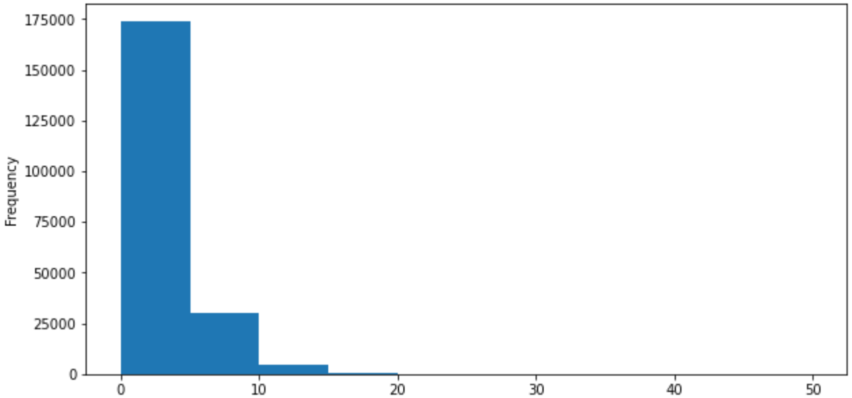
Frequency of data account recharged in last 90 days



- Majority customers have data account recharge frequency of 0 times

cnt_loans30

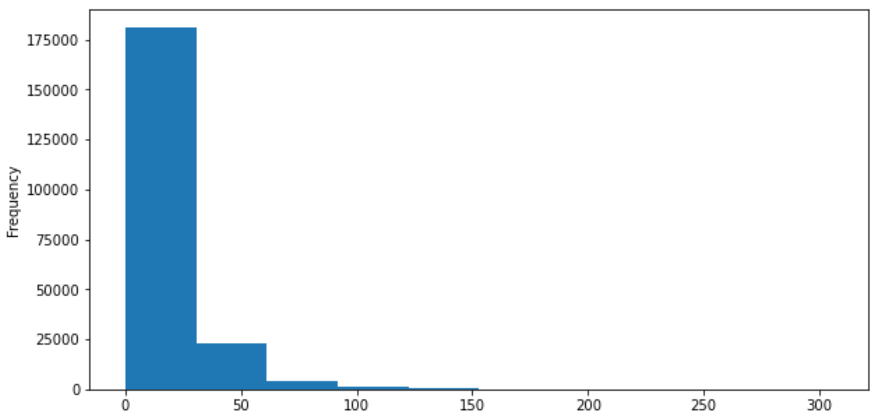
Number of loans taken by user in last 30 days



- Majority customers have taken 0-10 loans in last 30 days

amnt_loans30

Total amount of loans taken by user in last 30 days



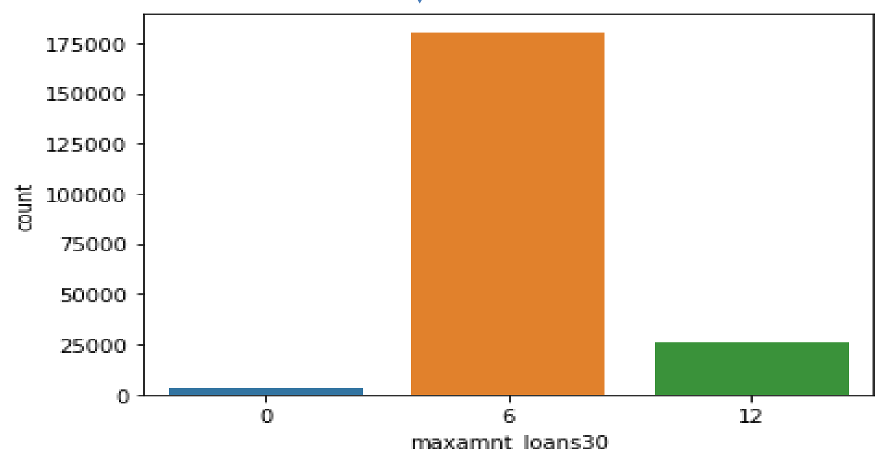
- Majority customers have taken 0-30 loans in last 30 days

maxamnt_loans30

maximum amount of loan taken by the user in last 30 days

Maximum amount of loan taken by the user in last 30 days should be 0,6 or12. Hence replacing absurd amount >3000 with the mode of the column

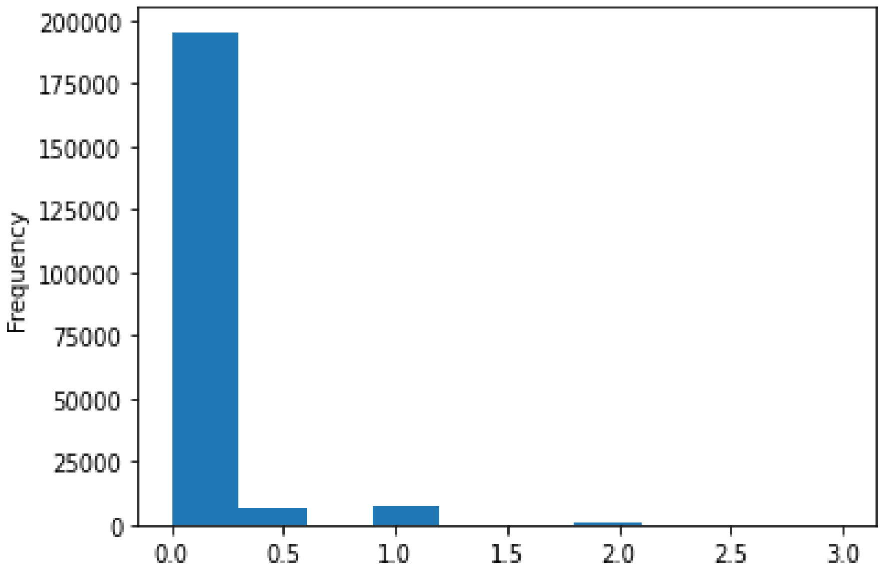
As max amount should be preferably in whole numbers, hence converted to int datatype



- Majority customers have maximum amount of loan taken by the user in last 30 days as 6

medianamnt_loans30

Median of amounts of loan taken by the user in last 30 days



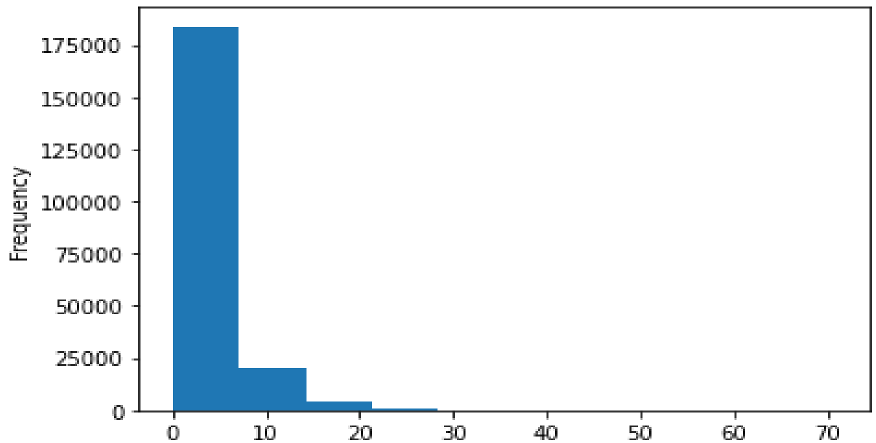
- Majority customers have median of amounts of loan taken by the user in last 30 days as 0

cnt_loans90

Number of loans taken by user in last 90 days

Number of loans taken by user in last 90 days should be less than 300. Hence replacing number>300 with the mean of the column

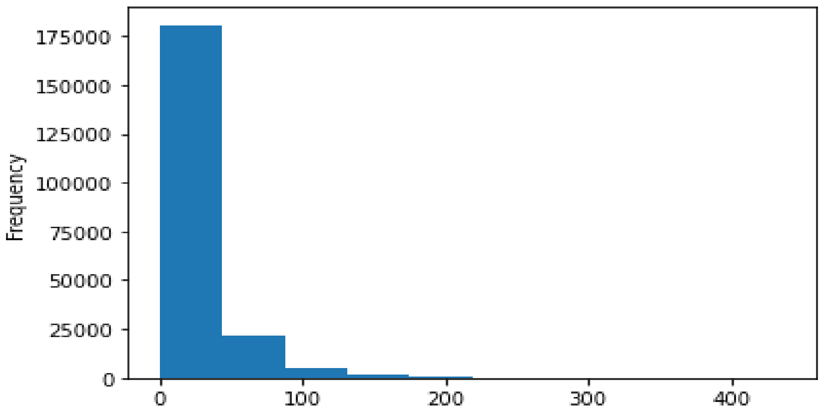
As max number should be in whole numbers, hence converted to int datatype



- Majority customers have taken 0-5 loans in last 90 days

amnt_loans90

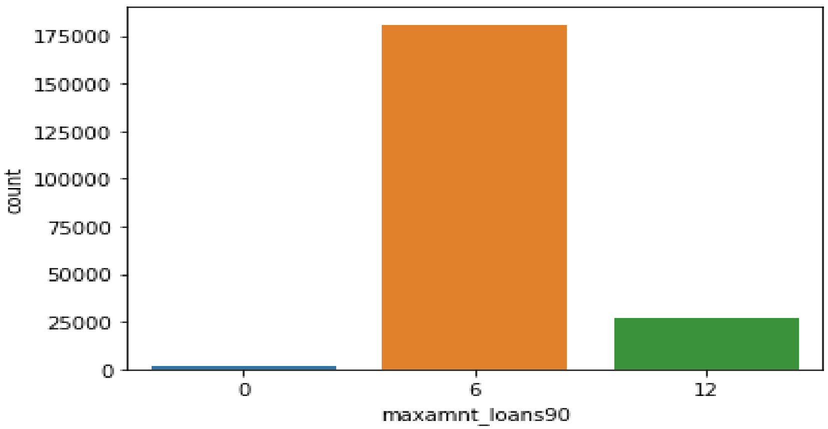
Total amount of loans taken by user in last 90 days



- Majority customers have taken 0-30 loans in last 30 days

maxamnt_loans90

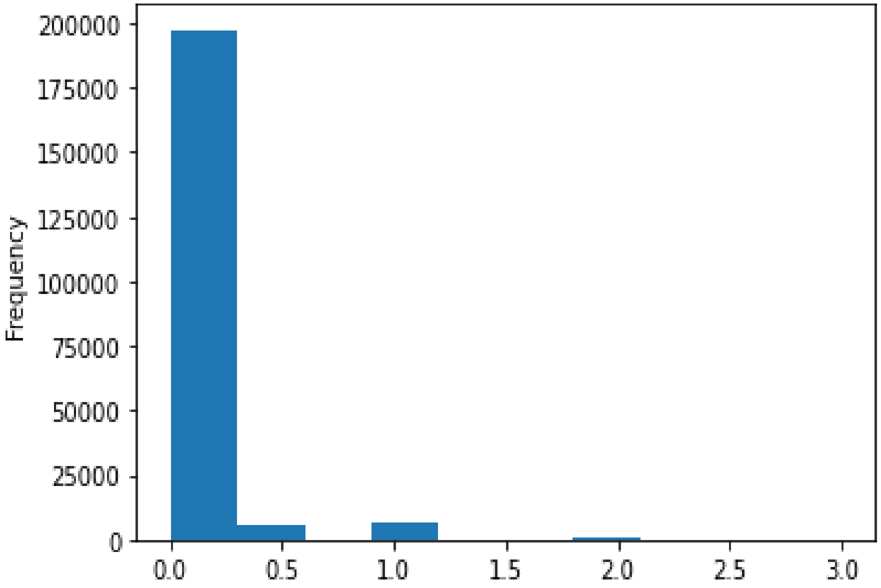
maximum amount of loan taken by the user in last 90 days



- Majority customers have maximum amount of loan taken by the user in last 90 days as 6

medianamnt_loans90

Median of amounts of loan taken by the user in last 90 days

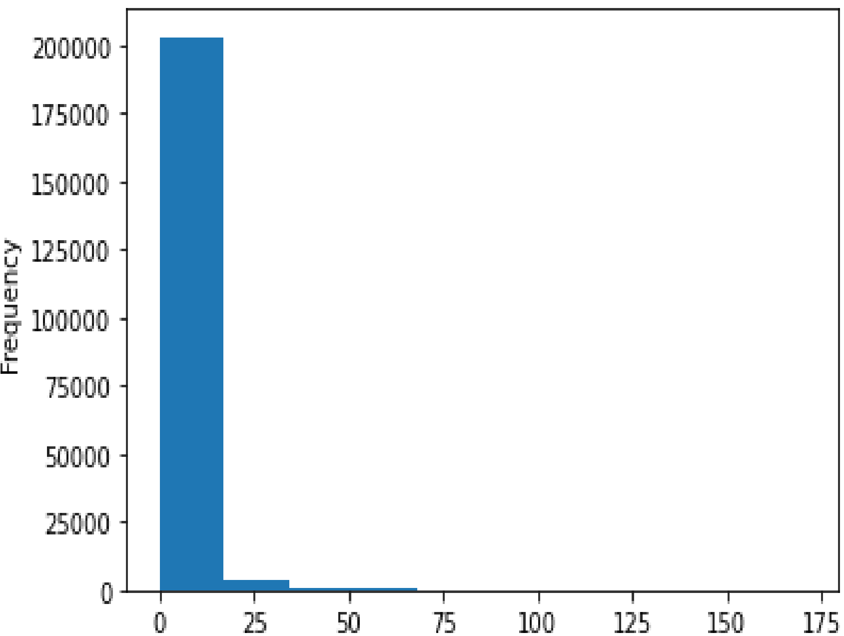


- Majority customers have median of amounts of loan taken by the user in last 90 days as 0

Payback30

Average payback time in days over last 30 days

As days should be in whole numbers, hence converted to int datatype

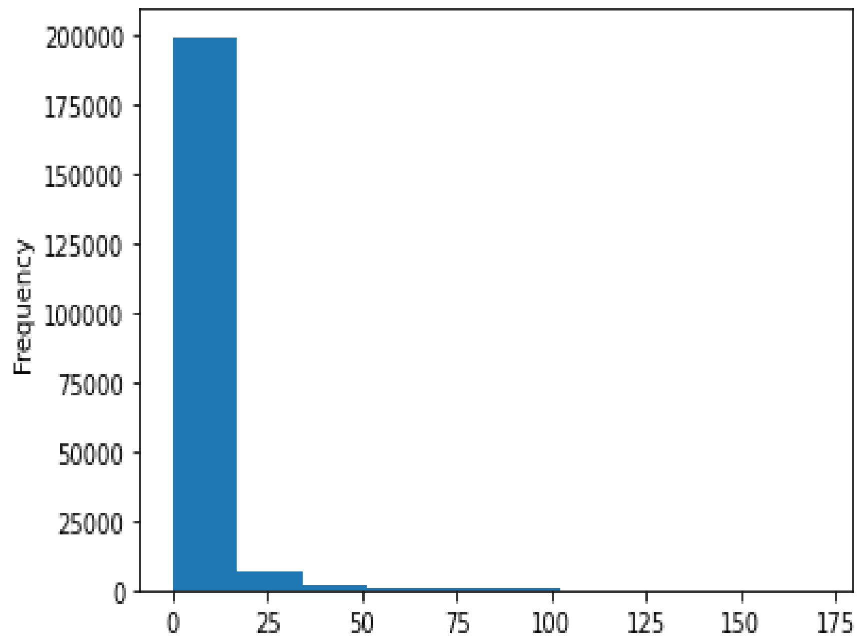


- Majority customers have 0-15 days average payback time over last 30 days

Payback90

Average payback time in days over last 90 days

As days should be in whole numbers, hence converted to int datatype



- Majority customers have 0-15 days average payback time over last 30 days

Statistical Analysis using Describe method-

label
- count 209593.0000
- mean 0.875177
- std 0.330519
- min 0.000000
- 25% 1.000000
- 50% 1.000000
- 75% 1.000000
- max 1.000000

aon
- count 209593.0000
- mean 660.200880
- std 493.406361
- min 1.000000
- 25% 252.000000
- 50% 537.000000
- 75% 957.000000
- max 2440.000000

daily_decr30
- count 209593.0000
- mean 520.427440
- std 619.024143
- min 0.000000
- 25% 45.500000
- 50% 520.427440
- 75% 520.427440
- max 3000.000000

daily_decr90
- count 209593.00000
- mean 511.04720
- std 610.80538
- min 0.00000
- 25% 45.79200
- 50% 511.04720
- 75% 511.04720
- max 3000.00000

rental30
- count 209593.0000
- mean 1272.025581
- std 1169.950934
- min 0.000000
- 25% 329.980000
- 50% 1218.580000
- 75% 1584.360000
- max 5000.000000

rental90
- count 209593.0000
- mean 1332.011362
- std 1166.600553
- min 0.000000
- 25% 380.130000
- 50% 1332.011362
- 75% 1596.550000
- max 4999.990000

last_rech_date_ma
- count 209593.0000
- mean 6.14886
- std 9.33035
- min 0.00000
- 25% 1.00000
- 50% 3.00000
- 75% 7.00000
- max 113.00000

last_rech_date_da
- count 209593.000000
- mean 0.930518
- std 7.028073
- min 0.000000
- 25% 0.000000
- 50% 0.000000
- 75% 0.000000
- max 115.000000

last_rech_amt_ma

- count 209593.0000
- mean 2064.452797
- std 2370.786034
- min 0.000000
- 25% 770.000000
- 50% 1539.000000
- 75% 2309.000000
- max 55000.000000

cnt_ma_rech30

- count 209593.0000
- mean 3.978057
- std 4.256090
- min 0.000000
- 25% 1.000000
- 50% 3.000000
- 75% 5.000000
- max 203.000000

fr_ma_rech30

- count 209593.0000
- mean 3.895563
- std 5.426773
- min 0.000000
- 25% 0.000000
- 50% 2.000000
- 75% 6.000000
- max 38.000000

sumamnt_ma_rech30

- count 209593.000000
- mean 7704.501157
- std 10139.621714
- min 0.000000
- 25% 1540.000000
- 50% 4628.000000
- 75% 10010.000000
- max 810096.000000

medianamnt_ma_rech30

- count 209593.000000
- mean 1812.817952
- std 2070.864620
- min 0.000000
- 25% 770.000000
- 50% 1539.000000
- 75% 1924.000000
- max 55000.000000

medianmarechprebal30

- count 209593.00000
- mean 87.984085
- std 176.261707
- min 0.000000
- 25% 11.670000
- 50% 35.000000
- 75% 85.670000
- max 2988.000000

cnt_ma_rech90

- count 209593.0000
- mean 6.31543
- std 7.19347
- min 0.00000
- 25% 2.00000
- 50% 4.00000
- 75% 8.00000
- max 336.00000

fr_ma_rech90

- count 209593.000000
- mean 7.716780
- std 12.590251
- min 0.000000
- 25% 0.000000
- 50% 2.000000
- 75% 8.000000
- max 88.000000

sumamnt_ma_rech90

- count 209593.000000
- mean 12396.218352
- std 16857.793882
- min 0.000000
- 25% 2317.000000
- 50% 7226.000000
- 75% 16000.000000
- max 953036.000000

medianamnt_ma_rech90

- count 209593.00000
- mean 1864.595821
- std 2081.680664
- min 0.000000
- 25% 773.000000
- 50% 1539.000000
- 75% 1924.000000
- max 55000.000000

medianmarechprebal90

- count 209593.00000
- mean 92.025541
- std 369.215658
- min -200.000000
- 25% 14.600000
- 50% 36.000000
- 75% 79.310000
- max 41456.500000

cnt_da_rech30

- count 209593.000000
- mean 0.022944
- std 0.267795
- min 0.000000
- 25% 0.000000
- 50% 0.000000
- 75% 0.000000
- max 34.000000

fr_da_rech30

- count 209593.0000
- mean 0.018039
- std 0.442169
- min 0.000000
- 25% 0.000000
- 50% 0.000000
- 75% 0.000000
- max 25.000000

cnt_da_rech90

- count 209593.0000
- mean 0.041495
- std 0.397556
- min 0.000000
- 25% 0.000000
- 50% 0.000000
- 75% 0.000000
- max 38.000000

fr_da_rech90

- count 209593.0000
- mean 0.045712
- std 0.951386
- min 0.000000
- 25% 0.000000
- 50% 0.000000
- 75% 0.000000
- max 64.000000

cnt_loans30

- count 209593.000000
- mean 2.758981
- std 2.554502
- min 0.000000
- 25% 1.000000
- 50% 2.000000
- 75% 4.000000
- max 50.000000

amnt_loans30

- count 209593.0000
- mean 17.952021
- std 17.379741
- min 0.000000
- 25% 6.000000
- 50% 12.000000
- 75% 24.000000
- max 306.000000

maxamnt_loans30

- count 209593.0000
- mean 6.654554
- std 2.147858
- min 0.000000
- 25% 6.000000
- 50% 6.000000
- 75% 6.000000
- max 12.000000

medianamnt_loans30

- count 209593.0000
- mean 0.054029
- std 0.218039
- min 0.000000
- 25% 0.000000
- 50% 0.000000
- 75% 0.000000
- max 3.000000

cnt_loans90

- count 209593.0000
- mean 3.689408
- std 4.016255
- min 0.000000
- 25% 1.000000
- 50% 2.000000
- 75% 5.000000
- max 71.000000

amnt_loans90

- count 209593.0000
- mean 23.645398
- std 26.469861
- min 0.000000
- 25% 6.000000
- 50% 12.000000
- 75% 30.000000
- max 438.000000

maxamnt_loans90

- count 209593.0000
- mean 6.703134
- std 2.103864
- min 0.000000
- 25% 6.000000
- 50% 6.000000
- 75% 6.000000
- max 12.000000

medianamnt_loans90

- count 209593.0000
- mean 0.046077
- std 0.200692
- min 0.000000
- 25% 0.000000
- 50% 0.000000
- 75% 0.000000
- max 3.000000

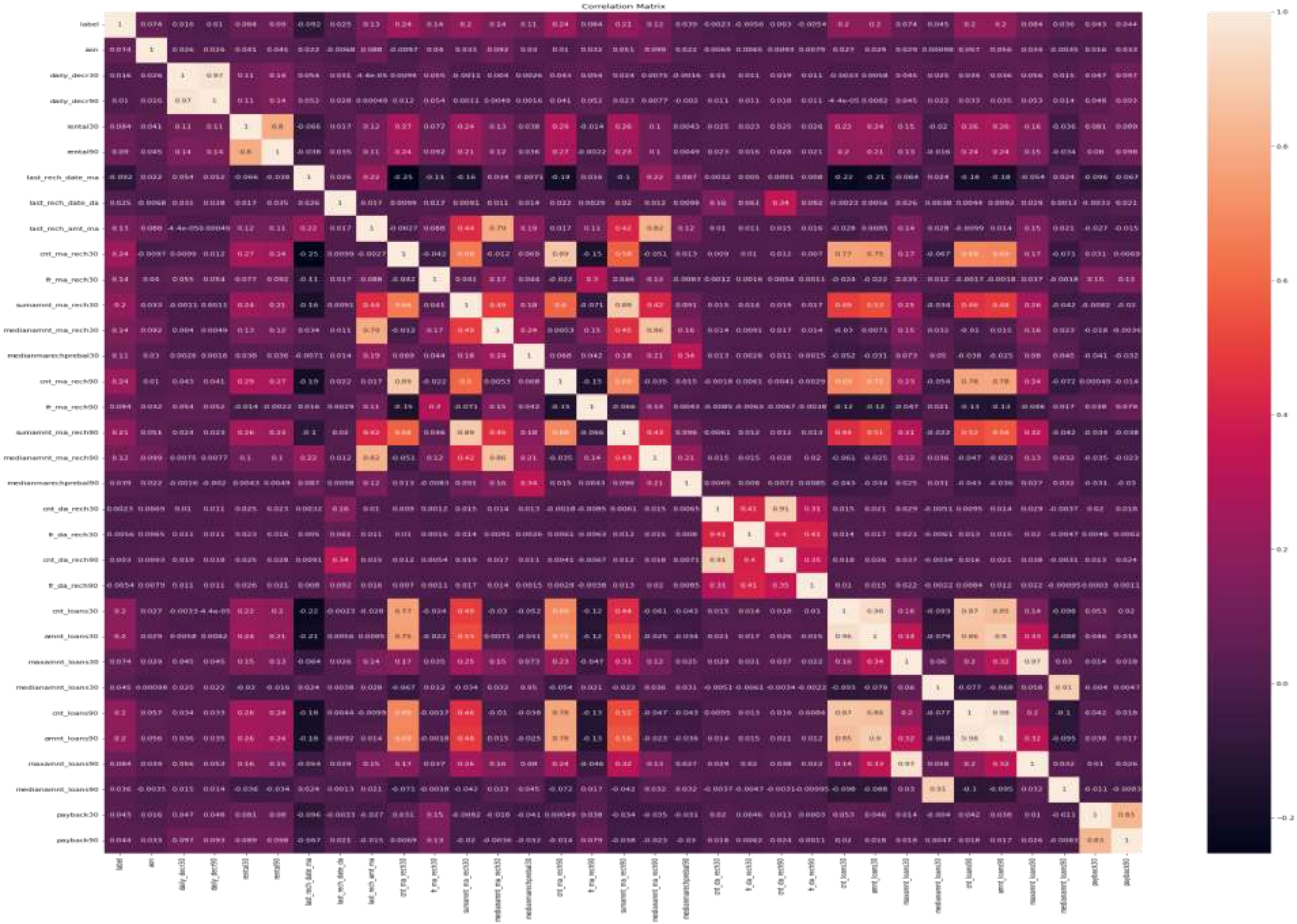
payback30

- count 209593.0000
- mean 3.232226
- std 8.762775
- min 0.000000
- 25% 0.000000
- 50% 0.000000
- 75% 3.000000
- max 171.000000

payback90

- count 209593.0000
- mean 4.126517
- std 10.256986
- min 0.000000
- 25% 0.000000
- 50% 1.000000
- 75% 4.000000
- max 171.000000

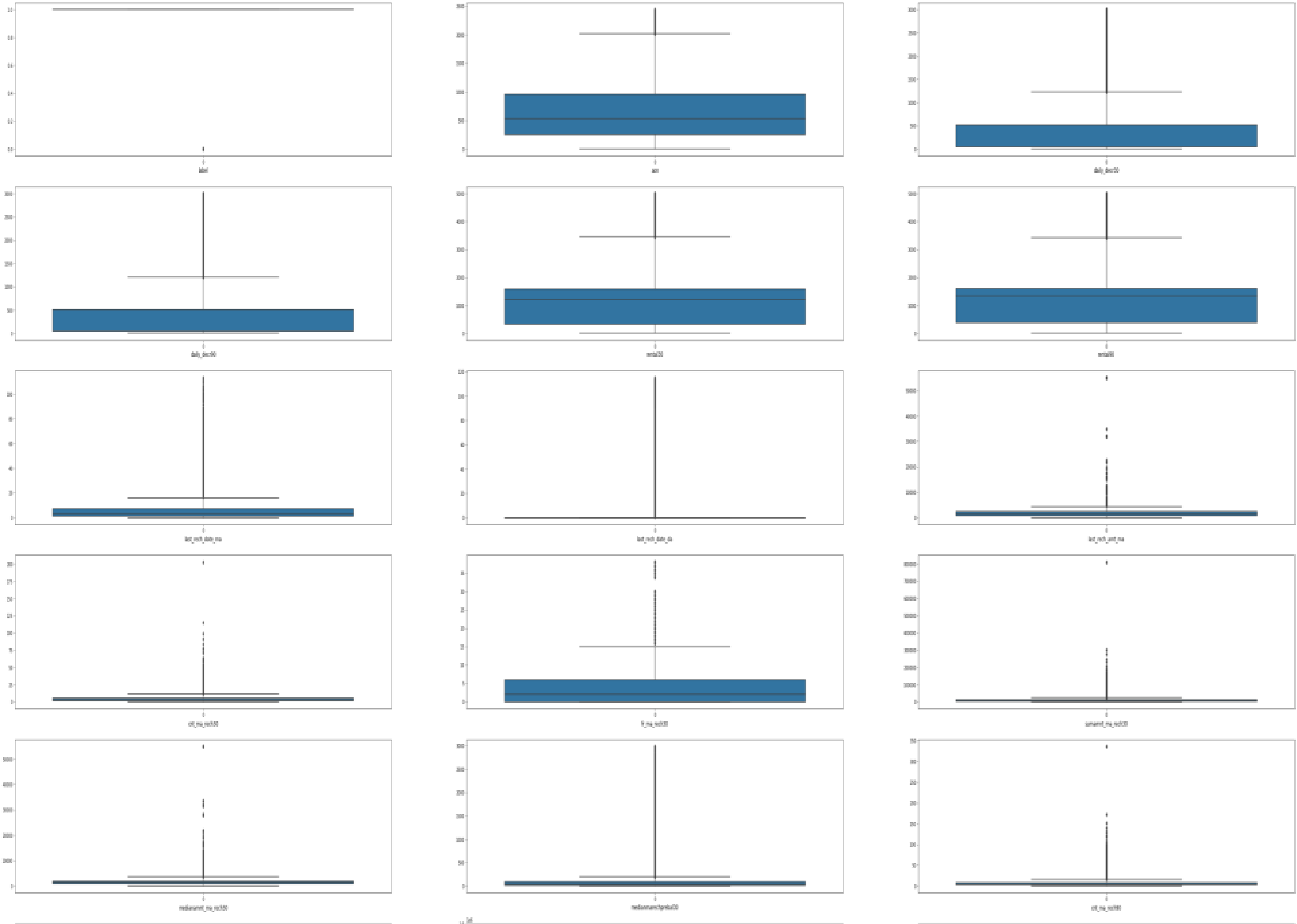
The Correlation Matrix using heatmap

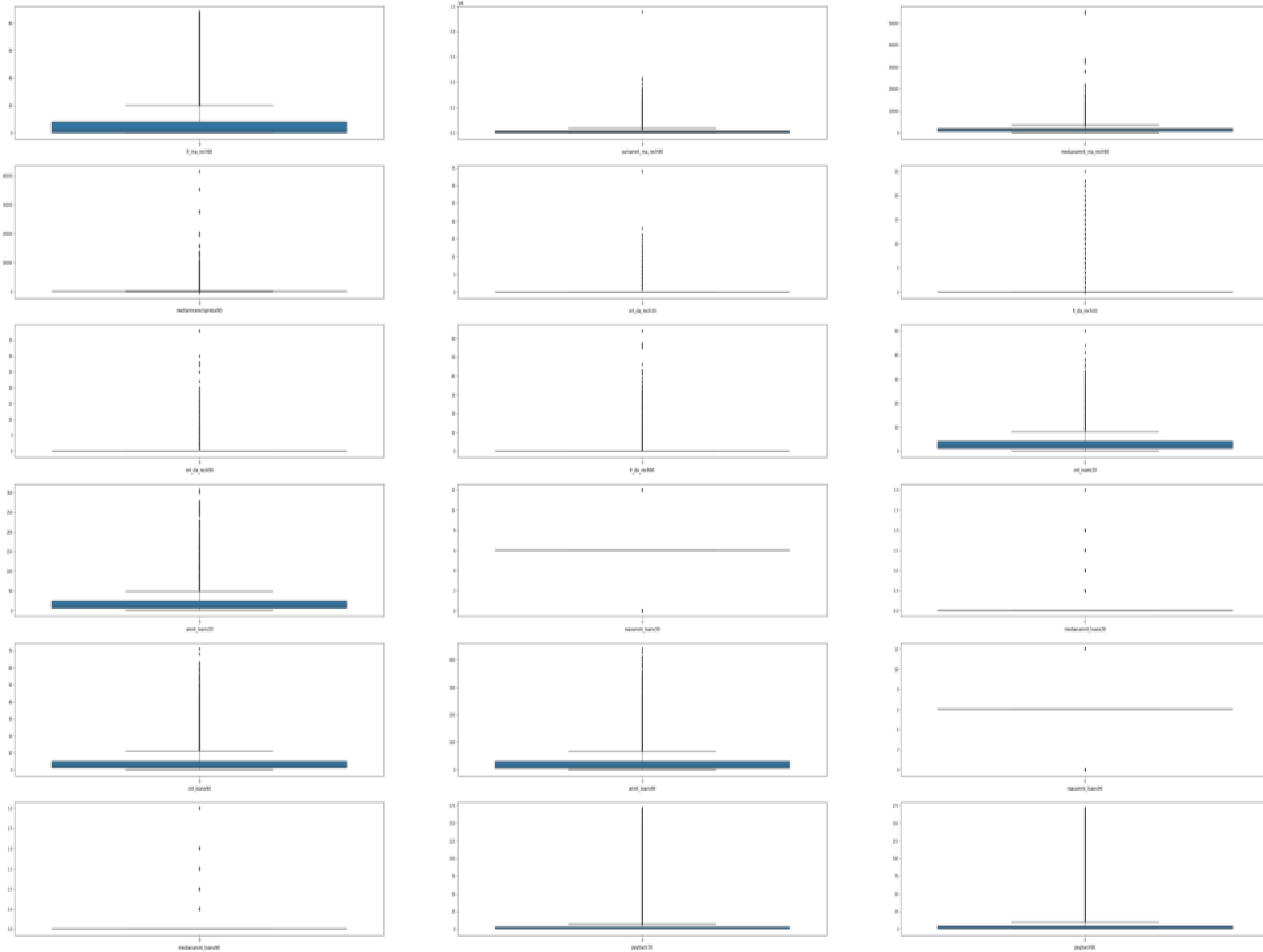


- Correlation between the columns and the label ‘label’ using corr method-
  - label - 1.000000
  - cnt_ma_rech30 - 0.237331
  - cnt_ma_rech90 - 0.236392
  - sumamnt_ma_rech90 - 0.205793
  - sumamnt_ma_rech30 - 0.202828
  - amnt_loans90 - 0.199788
  - cnt_loans90 - 0.199593
  - amnt_loans30 - 0.197272
  - cnt_loans30 - 0.196283
  - fr_ma_rech30 - 0.142612
  - medianamnt_ma_rech30 - 0.141490
  - last_rech_amt_ma - 0.131804
  - medianamnt_ma_rech90 - 0.120855
  - medianmarechprebal30 - 0.106691
  - rental90 - 0.090129
  - fr_ma_rech90 - 0.084385
  - maxamnt_loans90 - 0.084144
  - rental30 - 0.083731
  - maxamnt_loans30 - 0.073959
  - aon - 0.073587
  - medianamnt_loans30 - 0.044589
  - payback90 - 0.044201
  - payback30 - 0.043311
  - medianmarechprebal90 - 0.039300
  - medianamnt_loans90 - 0.035747
  - last_rech_date_da - 0.024841
  - daily_decr30 - 0.015940
  - daily_decr90 - 0.010440
  - cnt_da_rech90 - 0.002999
  - cnt_da_rech30 - 0.002333
  - fr_da_rech90 : -0.005418
  - fr_da_rech30 : -0.005563
  - last_rech_date_ma : -0.091982

- fr_da_rech90, fr_da_rech30 and last_rech_date_ma are negatively correlated to ‘label, the others are positively correlated
- cnt_ma_rech30 is 23.7% positively correlated
- last_rech_date_ma is 9.1% negatively correlated

- Visualizing outliers using boxplot method-





- Removing outliers using zscore method-

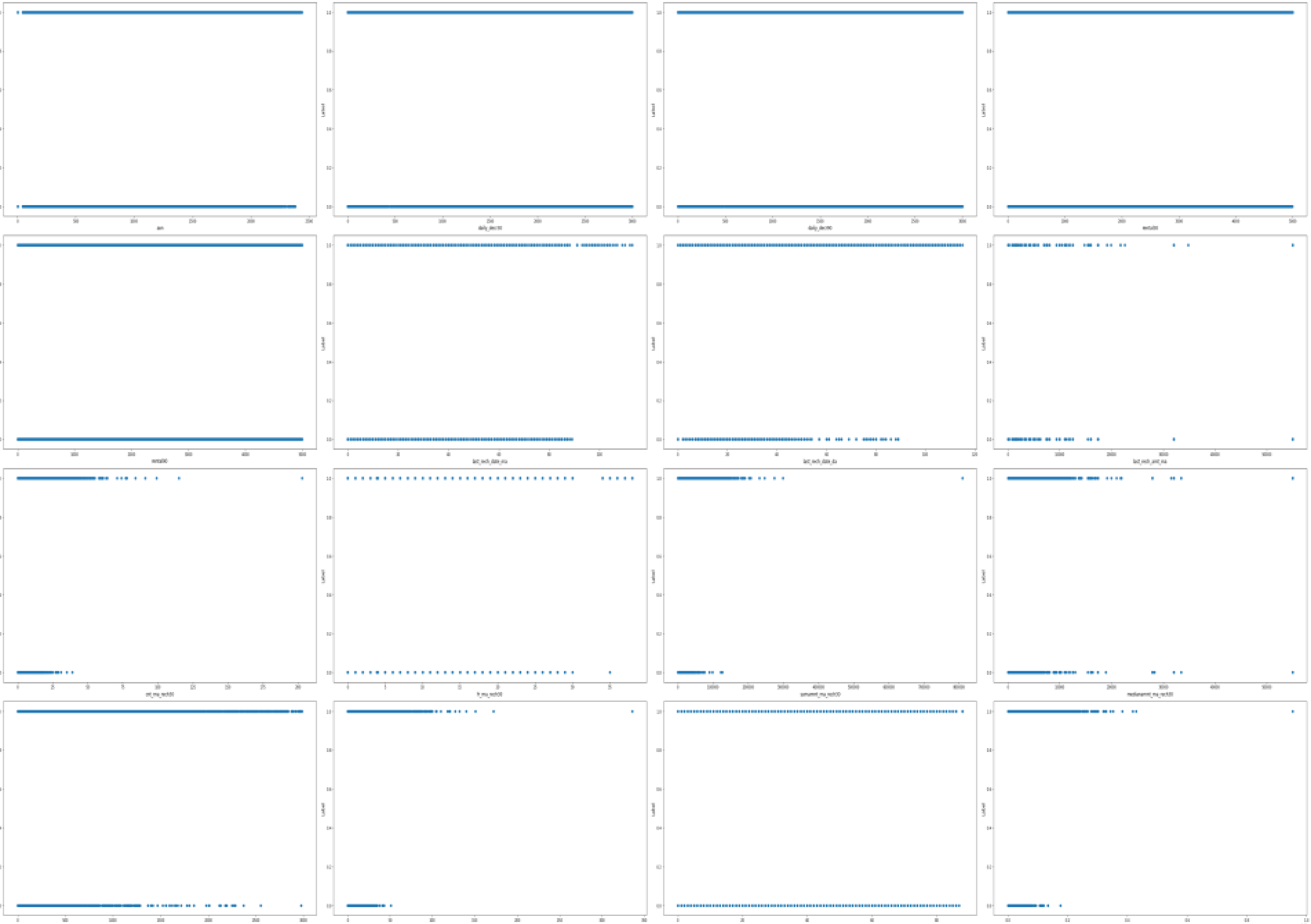
On removing the outliers the data loss is 27.9%, which is not acceptable, hence outliers are tolerated

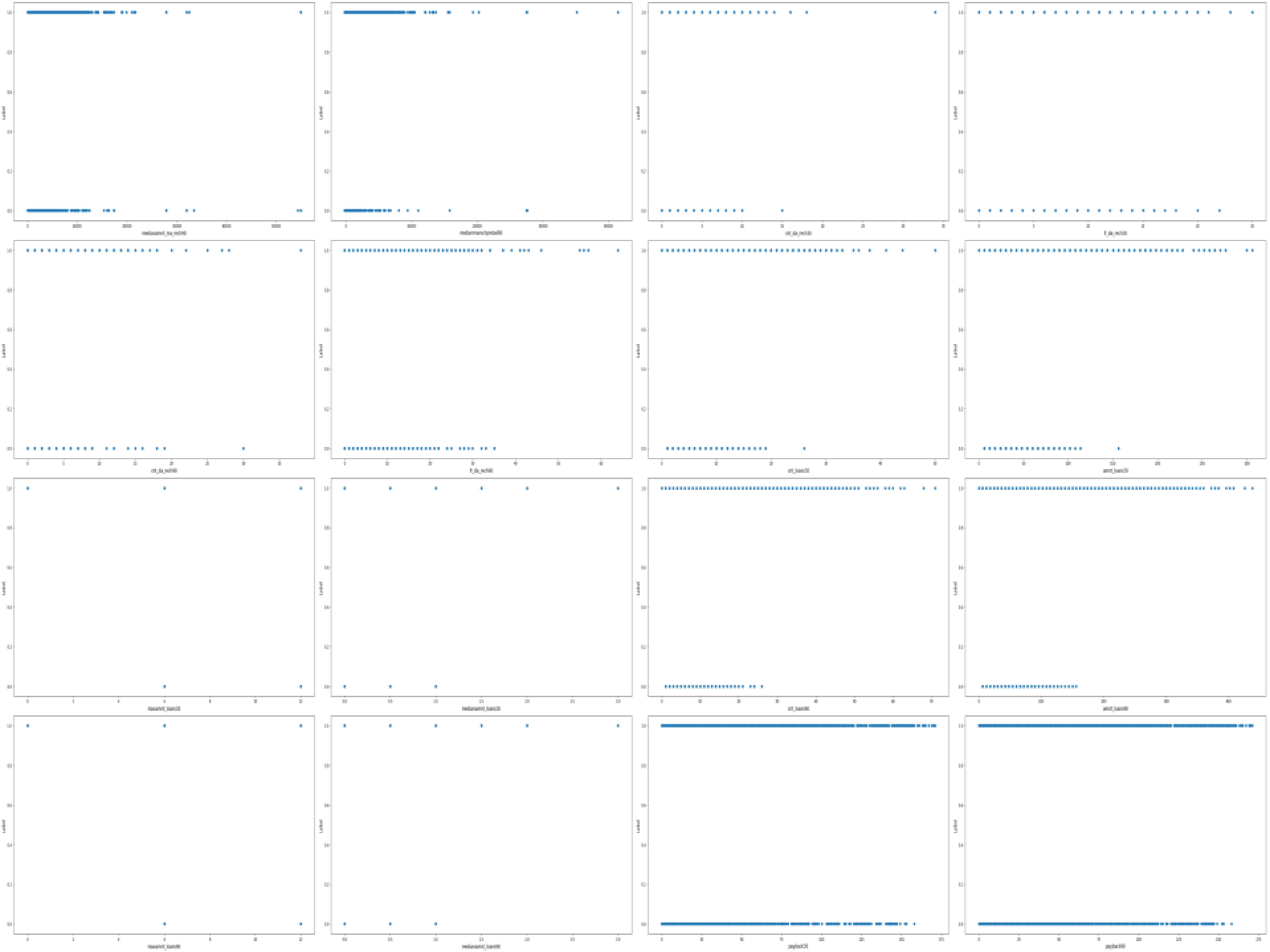
- The dataset is divided into x(features) and y (label)-

The x contains all the features other than the label ‘label’

The y contains only the label ‘label’

- Visualizing relationship between features and label-





- The skewness observed in graphical analysis was confirmed by using the skew method-
    - Driven km : 1.082297
    - Location : 0.801142
    - Model : 0.025468
    - Brand : 0.023501
    - Variant : -0.213326
    - Fuel : -0.277616
    - Manufacturing Year : -0.760945
    - Kind : -0.814154
- This skewness was removed using the power transformer
- The x(features) were scaled using the Standard Scaler
- The dataset was divided into train and test set using train test split and the best random state was found to be 62

- The skewness observed in graphical analysis was confirmed by using the skew method-
    - Medianmarechprebal90 - 44.880503
    - fr_da_rech30 - 31.164842
    - cnt_da_rech30 - 30.832071
    - fr_da_rech90 - 28.988083
    - cnt_da_rech90 - 27.267278
    - last_rech_date_da - 9.708988
    - Payback30 - 8.429388
    - Payback90 - 6.985942
    - sumamnt_ma_rech30 - 6.386787
    - Medianmarechprebal30- 6.166626
    - sumamnt_ma_rech90- 4.897950
    - medianamnt_loans90- 4.895720
    - medianamnt_loans30- 4.551043
    - last_rech_amt_ma- 3.781149
    - medianamnt_ma_rech90 - 3.752706
    - last_rech_date_ma - 3.583927
    - medianamnt_ma_rech30 - 3.512324
    - cnt_ma_rech90 - 3.425254
    - cnt_ma_rech30 - 3.283842
    - amnt_loans90 - 3.150006
    - cnt_loans90 - 3.004244
    - amnt_loans30 - 2.975719
    - cnt_loans30 0- 2.713421
    - fr_ma_rech90 - 2.285423
    - daily_decr90 - 2.096214
    - daily_decr30 - 2.050816
    - fr_ma_rech30 - 2.024554
    - maxamnt_loans90 - 1.678304
    - maxamnt_loans30 - 1.435587
    - rental30 - 1.256978
    - rental90 - 1.190303
    - aon - 0.959455
- This skewness was removed using the power transformer
- The x(features) were scaled using the Standard Scaler
- The dataset was divided into train and test set using train test split and the best random state was found to be 170
- The imbalanced dataset was downsampled using NearMiss as follows-
  Original y_train-: {1: 146633, 0: 21041}
  Downsampled y_train-: {1: 28054, 0: 21041}

Sofware Requirements-

- Jupyter Notebook – Interface for the program
- Pandas – for datafram working
- Numpy – to deal with null data
- matplotlib.pyplot – for data visualization
- Seaborn - for data visualization
- Warnings- to omit warnings
- sklearn.preprocessing – to import powertransform
- scipy.stats – to import zscore
- Zscore- to remove outliers
- power_transform- to remove the skewness in the data
- sklearn.model_selection- to import train_test_split
- train_test_split- to spit dataset into train and test samples
- sklearn.linear_model- to import Logistic Regression model,
- Logistic Regression – to use Logistic Regression model
- sklearn.metrics- to import accuracy, confusion matric and classification report, plot_roc_curve
- imblearn.under_sampling- to import NearMiss
- MearMiss- to undersample the data
- Collections- to import Counter
- Counter- to check the number of data under each classification in the label
- sklearn.neighbors- to import Kneighbours CLassifierModel
- Kneighbours-Classifier to use Kneighbours model
- sklearn.tree- to import DecisionTreeClassifier
- DecisionTreeClassifier- to use DecisionTreeClassifier Model
- sklearn.ensemble – to import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
- RandomForestClassifier- to use RandomForestClassifier model
- AdaBoostClassifier- to use AdaBoostClassifier model
- GradientBoostingClassifier- to use GradientBoostingClassifier model
- sklearn.svm – to import SVC
- SVC- to use SVC model
- sklearn.model_selection- to import cross_val_score
- cross_val_score- to check for overfitting and underfitting
- sklearn.model_selection- to import GridSearchCV
- GridSearchCV to enhance the working of the model by manipulating the parameters
- plot_roc_curve- to plot ROC_AUC plot

Train Test Split

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report

maxAccu=0       #maximum accuracy
maxRS=0         #best random state

#Finding Best random state
for i in range(0,200):
    x_train, x_test, y_train, y_test= train_test_split(X_scaled, y, test_size=0.2, random_state=i)
    LR=LogisticRegression()
    LR.fit(x_train, y_train)                #fitting the data will train the model
    predrf=LR.predict(x_test)               #this is the predicted target variable
    acc=accuracy_score(y_test, predrf)      #accuracy score
    print('accuracy', acc, 'random_state', i)

    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
        print('accuracy', maxAccu, 'random_state', i)
```

```
accuracy 0.8786707698179823 random_state 168
accuracy 0.878527636632553 random_state 169
accuracy 0.882773921133614 9 random_state 170
accuracy 0.8827739211336149 random_state 170
accuracy 0.877382571149120 9 random_state 171
accuracy 0.8788377585343162 random_state 172
accuracy 0.879147880436079 random_state 173
accuracy 0.8801975237958921 random_state 174
```

Downsampling the Imbalanced Dataset

```
from imblearn.under_sampling import NearMiss
from collections import Counter
ds=NearMiss(0.75)
x_train_ns,y_train_ns=ds.fit_sample(x_train,y_train)
print("Before fit ",Counter(y_train))
print("After fit ",Counter(y_train_ns))
```

```
Before fit  Counter({1: 146633, 0: 21041})
After fit  Counter({1: 28054, 0: 21041})
```

Model/s Development and Evaluation

The train and test data were applied on different models as follows

Logistic Regression Model

```
LR=LogisticRegression()
LR.fit(x_train_ns, y_train_ns)
predlr=LR.predict(x_test)
print("Accuracy ",accuracy_score(y_test, predlr)*100)     #accuracy score
print(confusion_matrix(y_test,predlr))
print(classification_report(y_test,predlr))
```

```
Accuracy  54.17829623798278
[[ 2006  3115]
 [16093 20705]]
              precision    recall  f1-score   support

           0       0.11      0.39      0.17      5121
           1       0.87      0.56      0.68     36798

    accuracy                           0.54     41919
   macro avg       0.49      0.48      0.43     41919
weighted avg       0.78      0.54      0.62     41919
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **54.18%**

- The **Confusion Matrix** for target test and pred_test(data predicted on features_test) is –

	True Positive	False Positive	
	2006	**3115**	
False Negative	**16093**	**20705**	True Negative

- The **Classification Report** for target test and pred_test(data predicted on features_test) is

	precision	recall	f1-score	support
0	0.11	0.39	0.17	5121
1	0.87	0.56	0.68	36798
accuracy			0.54	41919
macro avg	0.49	0.48	0.43	41919
weighted avg	0.78	0.54	0.62	41919

KNeighbors Classifier Model

```
from sklearn.neighbors import KNeighborsClassifier

kn=KNeighborsClassifier()
kn.fit(x_train_ns, y_train_ns)
predkn=kn.predict(x_test)
print("Accuracy ",accuracy_score(y_test, predkn)*100)     #accuracy score
print(confusion_matrix(y_test,predkn))
print(classification_report(y_test,predkn))
```

```
Accuracy  56.09866647582243
[[ 3128  1993]
 [16410 20388]]
              precision    recall  f1-score   support

           0       0.16      0.61      0.25      5121
           1       0.91      0.55      0.69     36798

    accuracy                           0.56     41919
   macro avg       0.54      0.58      0.47     41919
weighted avg       0.82      0.56      0.64     41919
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **56.09%**

- The **Confusion Matrix** for target test and pred_test(data predicted on features_test) is –

	True Positive	False Positive	
	3128	**1993**	
False Negative	**16410**	**20388**	True Negative

- The **Classification Report** for target test and pred_test(data predicted on features_test) is

```
              precision    recall  f1-score   support

           0       0.16      0.61      0.25      5121
           1       0.91      0.55      0.69     36798

    accuracy                           0.56     41919
   macro avg       0.54      0.58      0.47     41919
weighted avg       0.82      0.56      0.64     41919
```

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier()
dt.fit(x_train_ns, y_train_ns)
preddt=dt.predict(x_test)
print("Accuracy ",accuracy_score(y_test, preddt)*100)      #accuracy score
print(confusion_matrix(y_test,preddt))
print(classification_report(y_test,preddt))
```

```
Accuracy  43.293017486104155
[[ 3471  1650]
 [22121 14677]]
              precision    recall  f1-score   support

           0       0.14      0.68      0.23      5121
           1       0.90      0.40      0.55     36798

    accuracy                           0.43     41919
   macro avg       0.52      0.54      0.39     41919
weighted avg       0.81      0.43      0.51     41919
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **43.29%**

- The **Confusion Matrix** for target test and pred_test(data predicted on features_test) is –

	True Positive	False Positive	
	3471	**1650**	
False Negative	**22121**	**14677**	True Negative

- The **Classification Report** for target test and pred_test(data predicted on features_test) is

```
              precision    recall  f1-score   support

           0       0.14      0.68      0.23      5121
           1       0.90      0.40      0.55     36798

    accuracy                           0.43     41919
   macro avg       0.52      0.54      0.39     41919
weighted avg       0.81      0.43      0.51     41919
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier()
rf.fit(x_train_ns, y_train_ns)
predrf=rf.predict(x_test)
print("Accuracy ",accuracy_score(y_test, predrf)*100)      #accuracy score
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
Accuracy  41.17464634175433
[[ 3722  1399]
 [23260 13538]]
              precision    recall  f1-score   support

           0       0.14      0.73      0.23      5121
           1       0.91      0.37      0.52     36798

    accuracy                           0.41     41919
   macro avg       0.52      0.55      0.38     41919
weighted avg       0.81      0.41      0.49     41919
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **41.17%**

- The **Confusion Matrix** for target test and pred_test(data predicted on features_test) is –

	True Positive	False Positive	
	3722	**1399**	
False Negative	**23260**	**13538**	True Negative

- The **Classification Report** for target test and pred_test(data predicted on features_test) is

```
              precision    recall  f1-score   support

           0       0.14      0.73      0.23      5121
           1       0.91      0.37      0.52     36798

    accuracy                           0.41     41919
   macro avg       0.52      0.55      0.38     41919
weighted avg       0.81      0.41      0.49     41919
```

AdaBoost Classifier

```
from sklearn.ensemble import AdaBoostClassifier

ada=AdaBoostClassifier()
ada.fit(x_train_ns, y_train_ns)
predada=ada.predict(x_test)
print("Accuracy ",accuracy_score(y_test, predada)*100)      #accuracy score
print(confusion_matrix(y_test,predada))
print(classification_report(y_test,predada))
```

```
Accuracy  48.398101099739975
[[ 2886  2235]
 [19396 17402]]
              precision    recall  f1-score   support

           0       0.13      0.56      0.21      5121
           1       0.89      0.47      0.62     36798

    accuracy                           0.48     41919
   macro avg       0.51      0.52      0.41     41919
weighted avg       0.79      0.48      0.57     41919
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **48.39%**

- The **Confusion Matrix** for target test and pred_test(data predicted on features_test) is –

	True Positive	False Positive	
	2886	**2235**	
False Negative	**19396**	**17402**	True Negative

- The **Classification Report** for target test and pred_test(data predicted on features_test) is

	precision	recall	f1-score	support
0	0.13	0.56	0.21	5121
1	0.89	0.47	0.62	36798
accuracy			0.48	41919
macro avg	0.51	0.52	0.41	41919
weighted avg	0.79	0.48	0.57	41919

Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier

gbdt= GradientBoostingClassifier()
gbdt.fit(x_train_ns, y_train_ns)
gbdt_pred=gbdt.predict(x_test)
print("Accuracy ",accuracy_score(y_test, gbdt_pred)*100)      #accuracy score
print(confusion_matrix(y_test,gbdt_pred))
print(classification_report(y_test,gbdt_pred))
```

```
Accuracy  50.39958014265608
[[ 3263  1858]
 [18934 17864]]
              precision    recall  f1-score   support

           0       0.15      0.64      0.24      5121
           1       0.91      0.49      0.63     36798

    accuracy                           0.50     41919
   macro avg       0.53      0.56      0.44     41919
weighted avg       0.81      0.50      0.58     41919
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **50.39%**

- The **Confusion Matrix** for target test and pred_test(data predicted on features_test) is –

	True Positive	False Positive	
	3263	**1858**	
False Negative	**18934**	**17864**	True Negative

- The **Classification Report** for target test and pred_test(data predicted on features_test) is

	precision	recall	f1-score	support
0	0.15	0.64	0.24	5121
1	0.91	0.49	0.63	36798
accuracy			0.50	41919
macro avg	0.53	0.56	0.44	41919
weighted avg	0.81	0.50	0.58	41919

SVC

```
from sklearn.svm import SVC

svc=SVC()
svc.fit(x_train_ns, y_train_ns)
ad_pred=svc.predict(x_test)
print("Accuracy ",accuracy_score(y_test, ad_pred)*100)      #accuracy score
print(confusion_matrix(y_test,ad_pred))
print(classification_report(y_test,ad_pred))
```

```
Accuracy  43.397981822085455
[[ 3346  1775]
 [21952 14846]]
              precision    recall  f1-score   support

           0       0.13      0.65      0.22      5121
           1       0.89      0.40      0.56     36798

    accuracy                           0.43     41919
   macro avg       0.51      0.53      0.39     41919
weighted avg       0.80      0.43      0.51     41919
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **43.39%**

- The **Confusion Matrix** for target test and pred_test(data predicted on features_test) is –

	True Positive	False Positive	
	3346	**1775**	
False Negative	**21952**	**14846**	True Negative

- The **Classification Report** for target test and pred_test(data predicted on features_test) is

	precision	recall	f1-score	support
0	0.13	0.65	0.22	5121
1	0.89	0.40	0.56	36798
accuracy			0.43	41919
macro avg	0.51	0.53	0.39	41919
weighted avg	0.80	0.43	0.51	41919

Cross Validation

```
from sklearn.model_selection import cross_val_score

#validation accuracy
scr=cross_val_score(LR,x,y,cv=5)
print("Cross validation score of Logistic Regression: ", scr.mean())
```

```
Cross validation score of Logistic Regression:  0.874943437401111
```

```
scr2=cross_val_score(kn,x,y,cv=5)
print("Cross validation score of KNeighbor Classifier: ", scr2.mean())
```

```
Cross validation score of KNeighbor Classifier:  0.8815943318857924
```

```
scr3=cross_val_score(dt,x,y,cv=5)
print("Cross validation score of Decision Tree Classifier: ", scr3.mean())
```

```
Cross validation score of Decision Tree Classifier:  0.8630488644873712
```

```
scr4=cross_val_score(rf,x,y,cv=5)
print("Cross validation score of Random Forest Classifier: ", scr4.mean())
```

```
Cross validation score of Random Forest Classifier:  0.910488429890521
```

```
scr5=cross_val_score(ada,x,y,cv=5)
print("Cross validation score of Ada Boost Classifier: ", scr5.mean())
```

```
Cross validation score of Ada Boost Classifier:  0.8972007631616414
```

```
scr6=cross_val_score(gbdt,x,y,cv=5)
print("Cross validation score of Gradient Boost Classifier: ", scr6.mean())
```

```
Cross validation score of Gradient Boost Classifier:  0.9079215471069035
```

Model	**Cross Validation Score**
Logistic Regression	0.8749
KNeighbor Classifier	0.8815
Decision Tree Classifier	0.8630
Random Forest Classifier	0.9104
Ada Boost Classifier	0.8972
Gradient Boost Classifier	0.9079

- Random Forest Classifier is performing better, hence it is carried forward

Hyperparameter tuned Random Forest Classifier Model on Downsampled Data

```
RandomForestClassifier()
```

```
RandomForestClassifier()
```

```
from sklearn.model_selection import GridSearchCV

#Creating parameter list to pass in GridSearchCV
parameters={'max_features':['auto','sqrt','log2'], 'max_depth':[4,5,6,7,8], 'criterion':['gini', 'entropy']}
```

```
GCV=GridSearchCV(RandomForestClassifier(), parameters, cv=5, scoring="accuracy")
GCV.fit(x_train_ns,y_train_ns)       #fitting data in the model
GCV.best_params_                   #printing the best parameters found in GridSearchCV
```

```
{'criterion': 'gini', 'max_depth': 8, 'max_features': 'auto'}
```

```
GCV.best_estimator_
```

```
RandomForestClassifier(max_depth=8)
```

```
GCV_pred=GCV.best_estimator_.predict(x_test)        #Predicting with best parameters
accuracy_score(y_test,GCV_pred)
```

```
0.5207185285908538
```

ROC AUC Plot

```
from sklearn.metrics import plot_roc_curve
plot_roc_curve(GCV.best_estimator_,x_test,y_test)
plt.title("ROC AUC Plot")
plt.show()
```

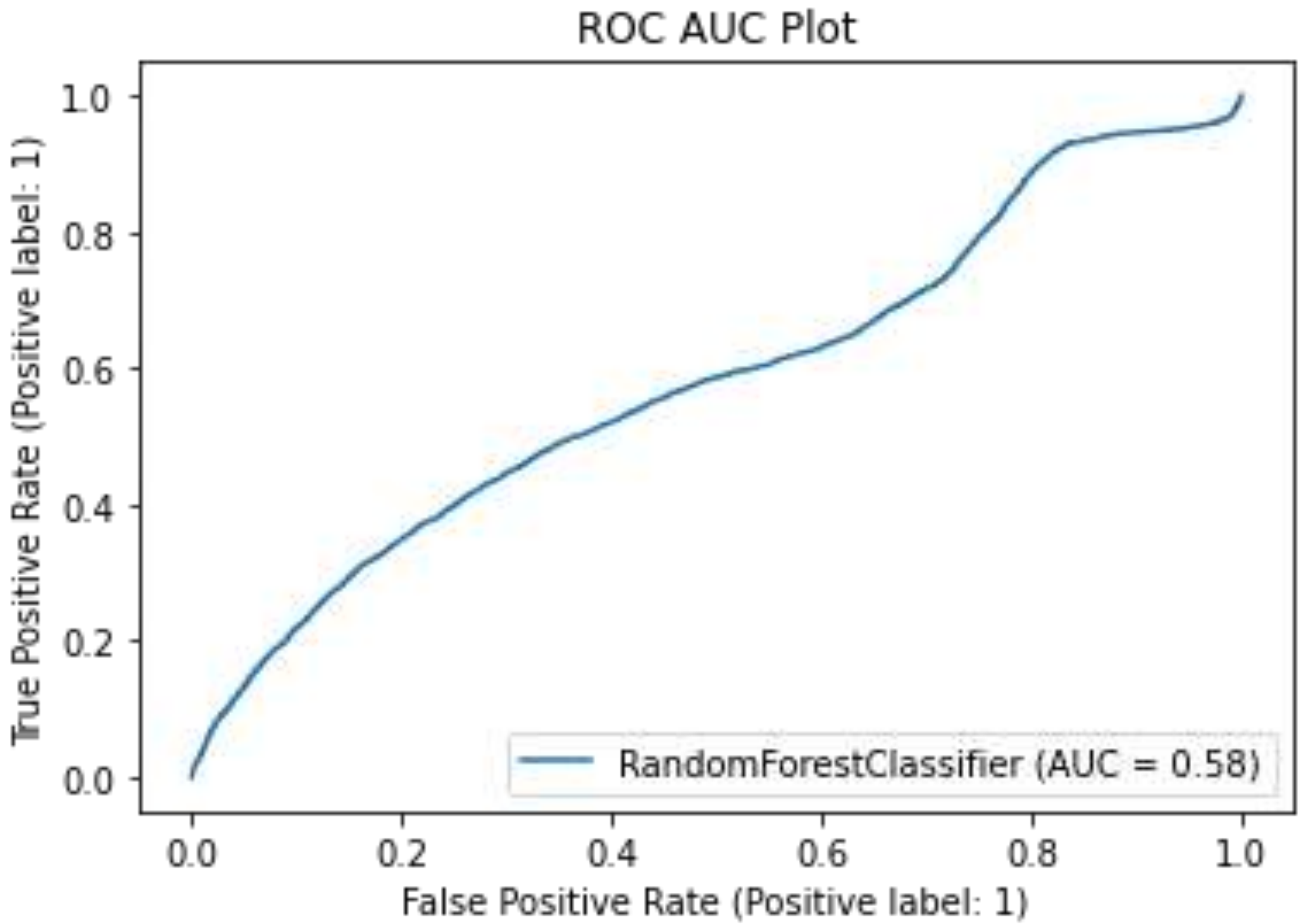
- Random Forest Classifier Model is hyperparameter tuned using GridSearchCV

- The best parameters for criterion, max_depth and max_features are found as follows-
    - **criterion: gini**
    - **max_depth: 8**
    - **Max_features: auto**

- Applying the above found best parameters on Random Forest Classifier Model, the following was obtained-

    - The **Accuracy** for target test and pred_test(data predicted on features_test) is **52.07%**

The ROC AUC Plot on the Hyperparameter tuned Random Forest Classifier Model on Downsampled Data



- **Final Accuracy** is **52%** and **AUC score** is **58%,** which depicts that our model is working okay

Hyperparameter tuned Random Forest Classifier Model on Original Data

```
RandomForestClassifier()
```

```
RandomForestClassifier()
```

```
from sklearn.model_selection import GridSearchCV

#Creating parameter list to pass in GridSearchCV
parameters={'max_features':['auto','sqrt','log2'], 'max_depth':[4,5,6,7,8], 'criterion':['gini', 'entropy']}
```

```
GCV=GridSearchCV(RandomForestClassifier(), parameters, cv=5, scoring="accuracy")
GCV.fit(x_train,y_train)        #fitting data in the model
GCV.best_params_                #printing the best parameters found in GridSearchCV
```

```
{'criterion': 'gini', 'max_depth': 8, 'max_features': 'log2'}
```

```
GCV.best_estimator_
```

```
RandomForestClassifier(max_depth=8, max_features='log2')
```

```
GCV_pred=GCV.best_estimator_.predict(x_test)        #Predicting with best parameters
accuracy_score(y_test,GCV_pred)
```

```
0.90739282902741
```

```
from sklearn.metrics import plot_roc_curve
plot_roc_curve(GCV.best_estimator_,x_test,y_test)
plt.title("ROC AUC Plot")
plt.show()
```

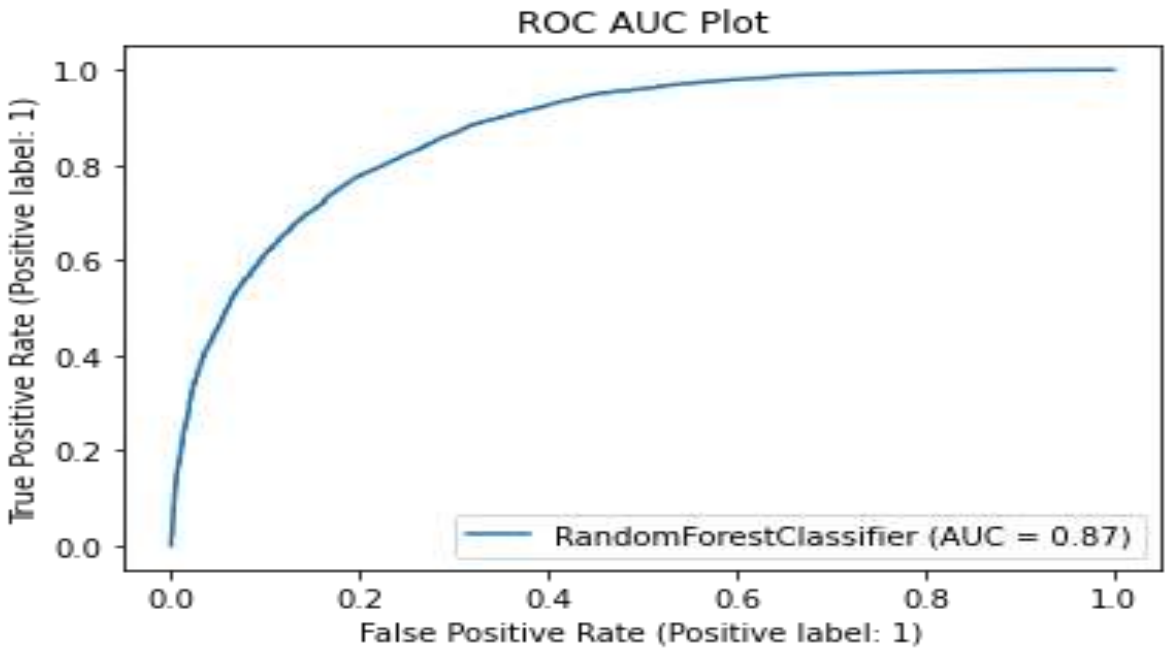
- Random Forest Classifier Model is hyperparameter tuned using GridSearchCV

- The best parameters for criterion, max_depth and max_features are found as follows-
  - **criterion: gini**
  - **max_depth: 8**
  - **Max_features: log2**

- Applying the above found best parameters on Random Forest Classifier Model, the following was obtained-

  - The **Accuracy** for target test and pred_test(data predicted on features_test) is **90.74%**

The ROC AUC Plot on the Hyperparameter tuned Random Forest Classifier Model on Original Data



- **Final Accuracy** is **90.74%** and **AUC score** is **87%,** which depicts that our model is working well

- **However, as the original dataset was imbalanced, working on the downsampled data is safer, even though its accuracy is not satisfactory**

CONCLUSION

- The EDA analysis of the data is essential as it helps to understand the relationship between the target and features as well as omit out unwanted columns, thereby taking care of overfitting scenario

- It is necessary to have the imbalanced dataset downsampled

- The models should be used properly, as their regularization /hyperparamter tuning is highly advisable for the best outcome.

- The project imparted key knowledge about the telecommunication sector and microfinance area and how essential it is to know the defaulters of the credit loan system. This allows the company to differentiate between potential defaulters as well. In addition it helps to identify the appropriate section of people eligible for the loan as well

- The limitation of the solution is that it predicts if the user is a defaulter or not. However, it does given any insight if a future user has a chance to be a defaulter or not. The limitation can be handled by assessing future users data to regression models to estimate in how many days they repay their previous loans. If the time period is between the allowed duration, they will classify as non-defaulter, else otherwise.