Micro Credit Defaulter

Submitted by:

Aniruddha Sawant

ACKNOWLEDGMENT

Aniruddha Sawant ("I") acknowledge that I have used the data provided by Flip Robo ("Company") namely Steps to follow.txt, Data_Description and Data file.csv to build the predictive model and have used Sample and Micro Credit Loan Use Case for guidance and to write report.

INTRODUCTION

- ## Background of the Domain Problem: -

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. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on.

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. Though, the MFI industry is primarily focusing on low-income families and are very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes.

Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients.

- ## Business Problem: -

We are working with one such client that is in Telecom Industry. They are a fixed wireless telecommunications network provider. They have launched various products and have developed its business and organization based on the budget operator model, offering better products at Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

They understand the importance of communication and how it affects a person's life, thus, focusing on providing their services and products to low-income families and poor customers that can help them in the need of hour. 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).

- **Review of Literature: -**

  There are total 36 features or variables present in the data that would be used to predict the label i.e whether a person will default or not. But before using those features to predict the outcome we have cleaned the data, transform the data into structured format and run many EDAs to make findings. This process ends up in selecting those features which are important to predict whether a person will default or not in micro loans.

  Have used total 6 machine learning models to train the data however have chosen GradientBoostingClassifier Model since its accuracy of train data and test data is high and close to each other i.e., 92.27% and 91.71% respectively. Using this model after hyper parameter tuning, we have predicted the test data where we got precision, recall and f1-score as 92%.

- **Motivation for the Problem Undertaken: -**

  In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

Analytical Problem Framing

- **Mathematical/ Analytical Modelling of the Problem**

1. Identifying sources of the data
2. Analysing the data
3. Cleaning and processing the data
4. Selecting most important features
5. Writing down findings and observations
6. Using different models to train the data
7. Selecting best fitted model for predictions
8. Predicting outcome for test data

- **Data Sources and their formats**

  There is only one excel sheet that contains label and 36 features.

- Data file.csv = This excel has train data in it, that will be used to train the machine learning models. This sheet contains total 37 columns including label i.e Label.

  Snapshot: -

label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	medianamnt_loans30	cnt_loans90
0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0	0.0	2.0
1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0	0.0	1.0
1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0	0.0	1.0
1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0	0.0	2.0
1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0	0.0	7.0

In this case, 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.

- **Data Pre-processing**

1. Checked the data type of each column.
2. Changed the Object data type to Integer.
3. Checked whether the data has any Null Values and fill those Null values using Mean and Mode method.
4. Checked whether the data is categorical data or continuous data.
5. There are many columns which has incorrect and uncommon data which needs to process. Below are the names of those columns:
    1. Aon
    2. cnt_da_rech30
    3. fr_da_rech30
    4. maxamnt_loans30
    5. cnt_loans90
    6. last_rech_date_ma
    7. last_rech_date_da
    8. fr_ma_rech30
    9. medianmarechprebal30

6. Encoded remaining Object data type columns to integer using encoder technique.
7. Checked the co-relation of features with label.
8. Checked the Multicollinearity between features.
9. Checked the VIF score of features.
10. Checked the Distribution of data.
11. Identified and removed outliers those are not allowed above and below the specific limit.
12. Used Power transformation to remove the skewness from data.
13. There was a imbalance between label hence have used SMOTE technique to balance the label.

- **Data Inputs**

1. Label = Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan {1: success, 0: failure}.
2. Msisdn = mobile number of users .
3. Aon = age on cellular network in days
4. daily_decr30 = Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)
5. daily_decr90 = Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)
6. rental30 = Average main account balance over last 30 days (Unsure of given definition).
7. rental90 = Average main account balance over last 90 days (Unsure of given definition).
8. last_rech_date_ma = Number of days till last recharge of main account.
9. last_rech_date_da = Number of days till last recharge of data account.
10. last_rech_amt_ma = Amount of last recharge of main account (in Indonesian Rupiah).
11. cnt_ma_rech30 = Number of times main account got recharged in last 30 days.
12. fr_ma_rech30 = Frequency of main account recharged in last 30 days (Unsure of given definition).
13. sumamnt_ma_rech30 = Total amount of recharge in main account over last 30 days (in Indonesian Rupiah).
14. medianamnt_ma_rech30 = Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah).
15. medianmarechprebal30 = Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah).
16. cnt_ma_rech90 = Number of times main account got recharged in last 90 days.
17. fr_ma_rech90 = Frequency of main account recharged in last 90 days (Unsure of given definition).
18. sumamnt_ma_rech90 = Total amount of recharge in main account over last 90 days (in Indonesian Rupiah).
19. medianamnt_ma_rech90 = Median of amount of recharges done in main account over last 90 days at user level (in Indonesian Rupiah).

20. medianmarechprebal90 = Median of main account balance just before recharge in last 90 days at user level (in Indonesian Rupiah).
21. cnt_da_rech30 = Number of times data account got recharged in last 30 days.
22. fr_da_rech30 = Frequency of data account recharged in last 30 days.
23. cnt_da_rech90 = Number of times data account got recharged in last 90 days.
24. fr_da_rech90 = Frequency of data account recharged in last 90 days.
25. cnt_loans30 = Number of loans taken by user in last 30 days.
26. amnt_loans30 = Total amount of loans taken by user in last 30 days.
27. maxamnt_loans30 = maximum amount of loan taken by the user in last 30 days. There are only two options: 5 & 10 Rs., for which the user needs to pay back 6 & 12 Rs. respectively.
28. medianamnt_loans30 = Median of amounts of loan taken by the user in last 30 days.
29. cnt_loans90 = Number of loans taken by user in last 90 days.
30. amnt_loans90 = Total amount of loans taken by user in last 90 days.
31. maxamnt_loans90 = Maximum amount of loan taken by the user in last 90 days.
32. medianamnt_loans90 = Median of amounts of loan taken by the user in last 90 days.
33. payback30 = Average payback time in days over last 30 days.
34. payback90 = Average payback time in days over last 90 days.
35. Pcircle = telecom circle.
36. pdate = date.

- **Hardware and Software Requirements and Tools Used**

1. Libraries and packages used
    - import numpy as np – For Numpy work
    - import pandas as pd – To work on DataFrame
    - import seaborn as sns – Plotting Graphs
    - import matplotlib.pyplot as plt - Plotting Graphs
    - import pickle – To save the Model
    - from sklearn.preprocessing import StandardScaler (To scale the train data), OrdinalEncoder(To encode object data to Integer), PowerTransformer (To remove skewness from dataset)
    - enc = OrdinalEncoder() = Assigned OrdinalEncoder to variable
    - from statsmodels.stats.outliers_influence import variance_inflation_factor – To calculate VIF score
    - from sklearn.model_selection import train_test_split – To split the data into train and test, GridSearchCV – To find the best parameters for model tuning, cross_val_score – To calculate the accuracy
    - from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve, roc_auc_score – To calculate and analyse model metrics.
    - import warnings, warnings.filterwarnings('ignore') – To ignore unwanted Warnings

2. Machine Learning models used
    - from sklearn.ensemble import RandomForestClassifier
    - from sklearn.linear_model import LogisticRegression
    - from sklearn.svm import SVC
    - from sklearn.ensemble import GradientBoostingClassifier
    - from sklearn.tree import DecisionTreeClassifier
    - from sklearn.neighbors import KNeighborsClassifier

3. Hardware used – 11th Gen Intel(R) Core (TM) i3-1115G4 @ 3.00GHz 3.00 GHz with 8.00 GB RAM and Windows 11.

4. Software used – Anaconda and Jupyter Notebook to build the model.

Model/s Development and Evaluation

- **Identification of possible problem**
  1. The data was not structured and organized hence cleaned the data using various data cleaning and pre-processing techniques.
  2. There are many outliers present in the data hence removed outliers.
  3. There was a skewness in the data hence have removed the skewness from the data.
  4. Scaled the data using Standard Scalar to make the data standardized to build a model.
  5. Removed imbalance from the label.

- **Testing of Identified Approaches**

  These are the algorithms which have been used to train and test data.

  1. RandomForestClassifier
  2. LogisticRegression
  3. GradientBoostingClassifier
  4. DecisionTreeClassifier
  5. KNeighborsClassifier
  6. SVC

- **Run and evaluate selected models**

1. **LogisticRegression**: -

```
log = LogisticRegression()
log.fit(x_train,y_train)

print_score(log,x_train,x_test,y_train,y_test, train=True)
print_score(log,x_train,x_test,y_train,y_test, train=False)
```

```
==============Train Result==============
Accuracy Score: 75.63%

==============Test Result==============
Accuracy Score: 76.53%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.72      0.78      0.75     10530
           1       0.81      0.75      0.78     13035

    accuracy                           0.77     23565
   macro avg       0.76      0.77      0.76     23565
weighted avg       0.77      0.77      0.77     23565

Cross Validation Score-  0.7581637411762807
```

2. **RandomForestClassifier**: -

```
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)

print_score(rfc,x_train,x_test,y_train,y_test, train=True)
print_score(rfc,x_train,x_test,y_train,y_test, train=False)
```

```
==============Train Result==============
Accuracy Score: 99.99%

==============Test Result==============
Accuracy Score: 92.82%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.92      0.91      0.92     10530
           1       0.93      0.94      0.94     13035

    accuracy                           0.93     23565
   macro avg       0.93      0.93      0.93     23565
weighted avg       0.93      0.93      0.93     23565

Cross Validation Score-  0.9263515819159764
```

3. **DecisionTreeClassifier**: -

```
dtc = DecisionTreeClassifier()
dtc.fit(x_train,y_train)

print_score(dtc,x_train,x_test,y_train,y_test, train=True)
print_score(dtc,x_train,x_test,y_train,y_test, train=False)
```

```
===============Train Result===============
Accuracy Score: 100.00%

===============Test Result===============
Accuracy Score: 87.96%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.86      0.87      0.87     10530
           1       0.90      0.89      0.89     13035

    accuracy                           0.88     23565
   macro avg       0.88      0.88      0.88     23565
weighted avg       0.88      0.88      0.88     23565

Cross Validation Score-  0.8765626123624191
```

4. **GradientBoostingClassifier**: -

```
gbdt = GradientBoostingClassifier()
gbdt.fit(x_train,y_train)

print_score(gbdt,x_train,x_test,y_train,y_test, train=True)
print_score(gbdt,x_train,x_test,y_train,y_test, train=False)
```

```
===============Train Result===============
Accuracy Score: 89.31%

===============Test Result===============
Accuracy Score: 89.16%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.87      0.88      0.88     10530
           1       0.91      0.90      0.90     13035

    accuracy                           0.89     23565
   macro avg       0.89      0.89      0.89     23565
weighted avg       0.89      0.89      0.89     23565

Cross Validation Score-  0.8872671099364631
```

5. **Support Vector Classifier**: -

```
svc = SVC()
svc.fit(x_train,y_train)

print_score(svc,x_train,x_test,y_train,y_test, train=True)
print_score(svc,x_train,x_test,y_train,y_test, train=False)
```

```
===============Train Result===============
Accuracy Score: 84.90%

===============Test Result===============
Accuracy Score: 85.05%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.83      0.84      0.83     10530
           1       0.87      0.86      0.86     13035

    accuracy                           0.85     23565
   macro avg       0.85      0.85      0.85     23565
weighted avg       0.85      0.85      0.85     23565

Cross Validation Score-  0.8468883500764136
```

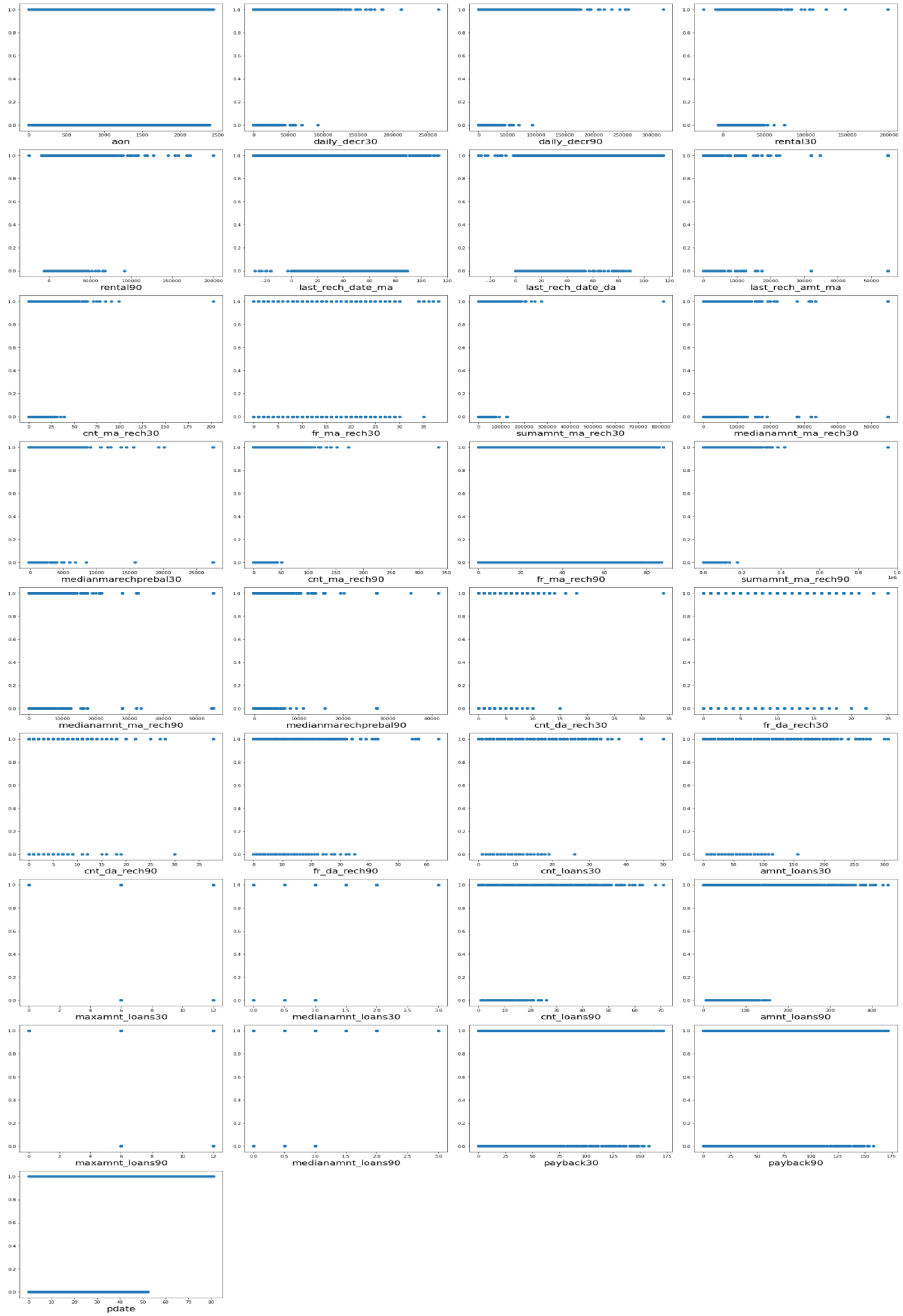
6. **KNeighborsClassifier**: -

```
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)

print_score(knn,x_train,x_test,y_train,y_test, train=True)
print_score(knn,x_train,x_test,y_train,y_test, train=False)
```

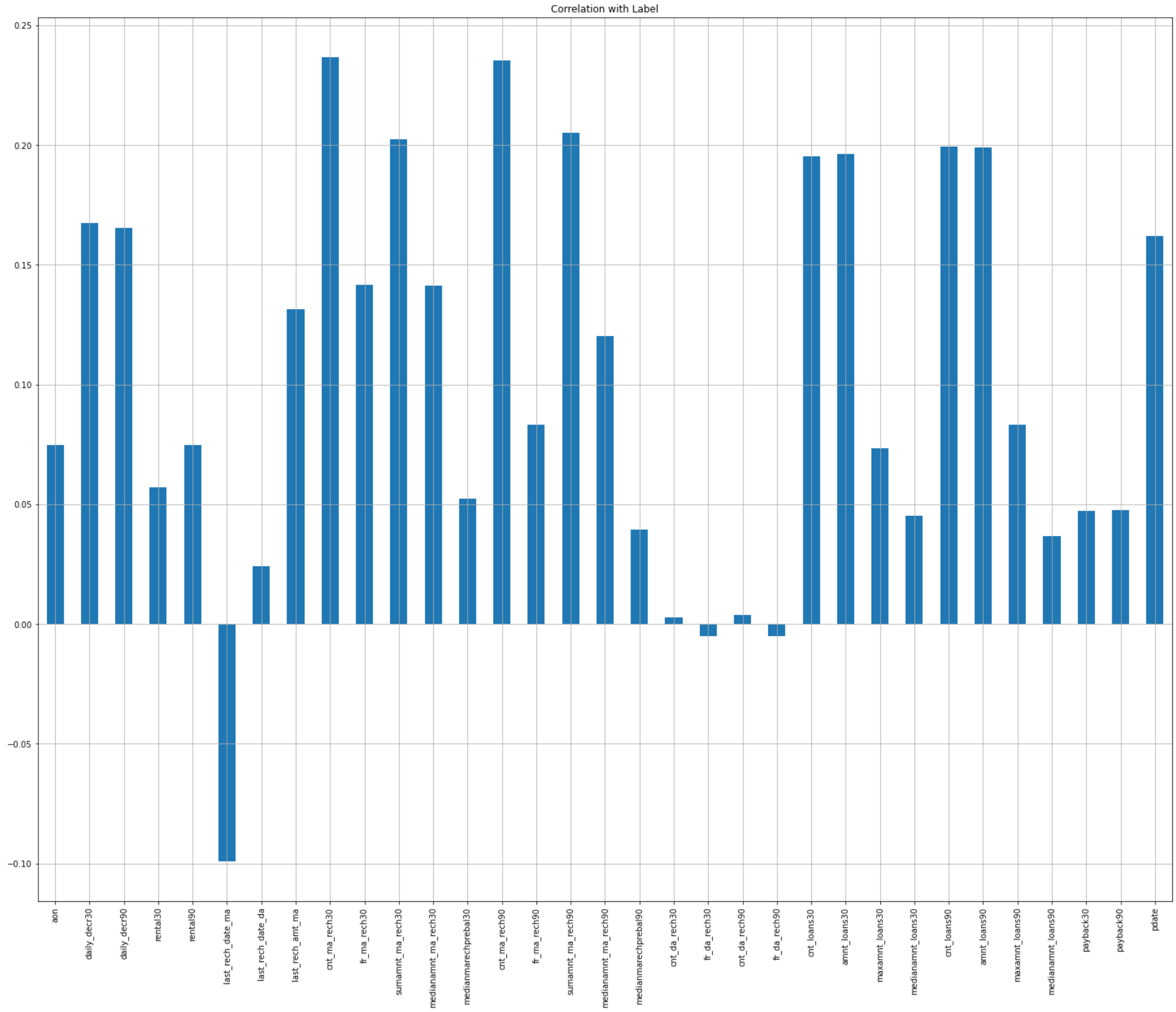
```
===============Train Result===============
Accuracy Score: 91.41%

===============Test Result===============
Accuracy Score: 86.96%


 Test Classification Report
              precision    recall  f1-score   support

           0       0.81      0.93      0.86     10530
           1       0.93      0.82      0.87     13035

    accuracy                           0.87     23565
   macro avg       0.87      0.88      0.87     23565
weighted avg       0.88      0.87      0.87     23565

Cross Validation Score-  0.876615400504811
```

- # Visualizations

1. Scatter Plot: -



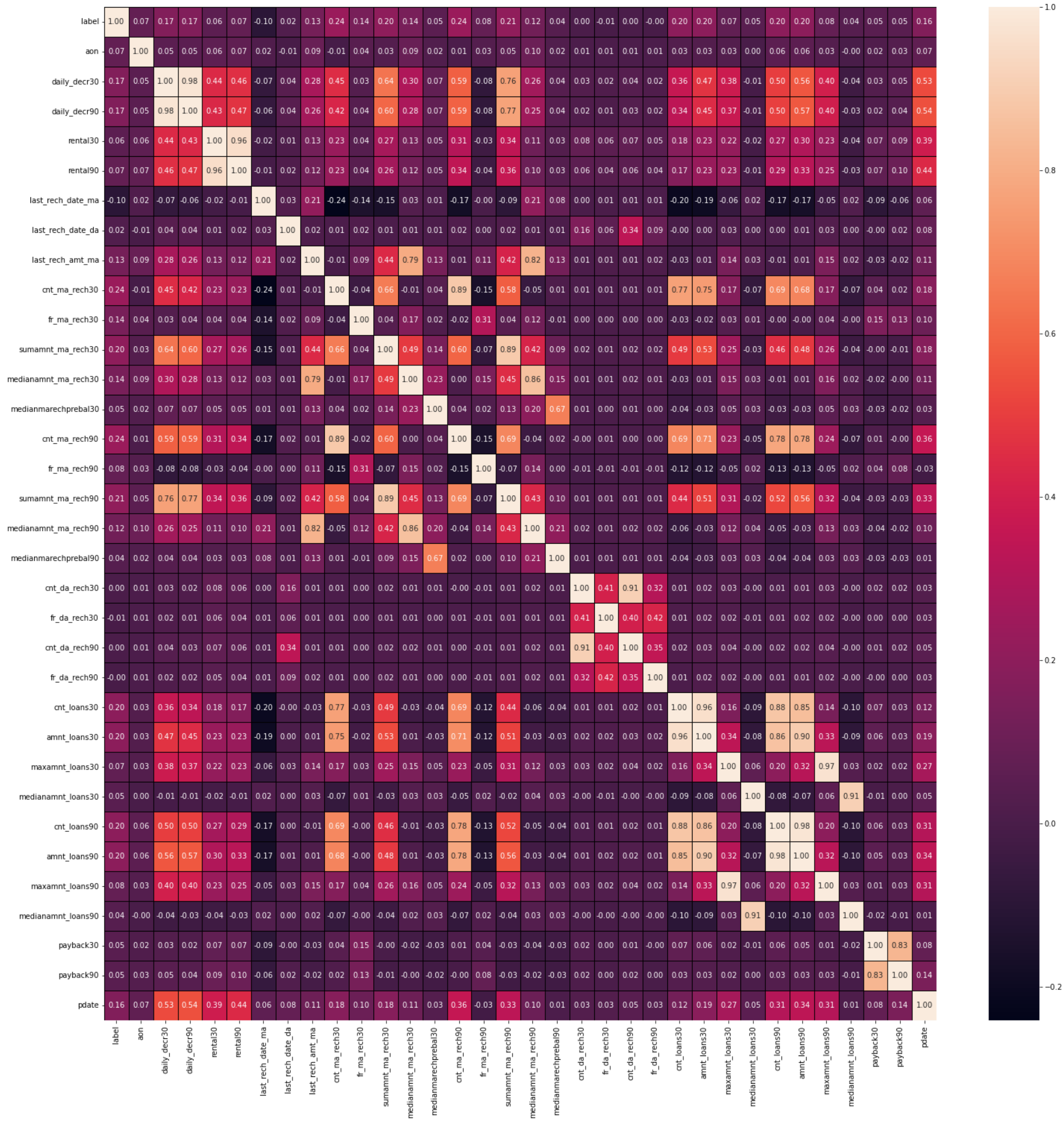
2. BarPlot: -



Observations:

- cnt_da_rech30, fr_da_rech30, cnt_da_rech90 and fr_da_rech90 has no co-relation with Label.
- cnt_ma_rech30, sumamnt_ma_rech30, cnt_ma_rech90 and sumamnt_ma_rech90 has high co-relation with Label.

3. HeatMap: -

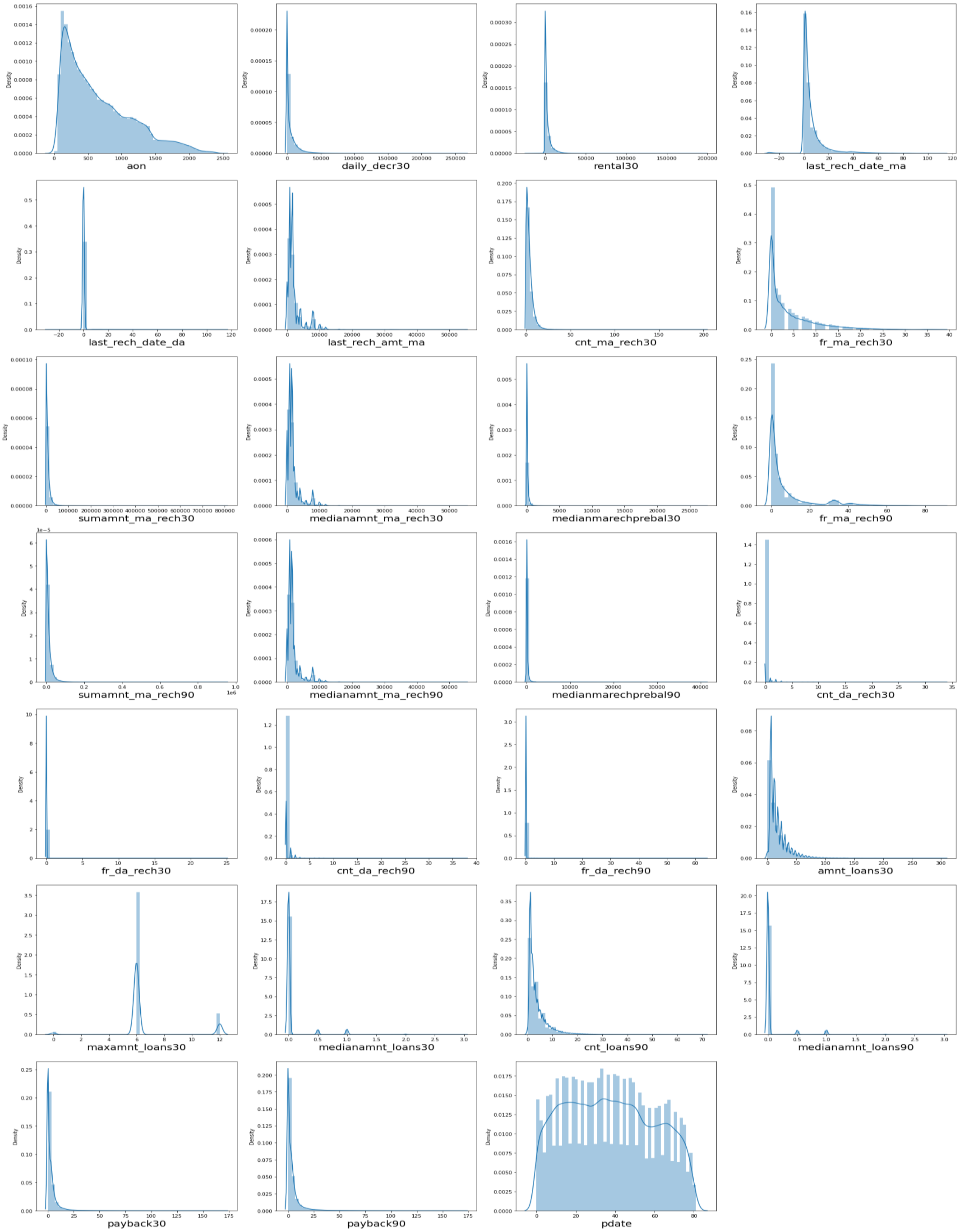


Observations:

Multicollinearity problem exist in this database

1. 'daily_decr30' and 'daily_decr90' columns has Multicollinearity problem.
2. 'rental30' and 'rental90' columns has Multicollinearity problem.
3. 'cnt_loans30' and 'amnt_loans30' columns has Multicollinearity problem.
4. 'cnt_da_rech30' and 'cnt_da_rech90' columns has Multicollinearity problem.
5. 'maxamnt_loans30' and 'maxamnt_loans90' columns has Multicollinearity problem.
6. 'cnt_loans90' and 'amnt_loans90' columns has Multicollinearity problem.
7. 'medianamnt_loans30' and 'medianamnt_loans90' columns has Multicollinearity problem.

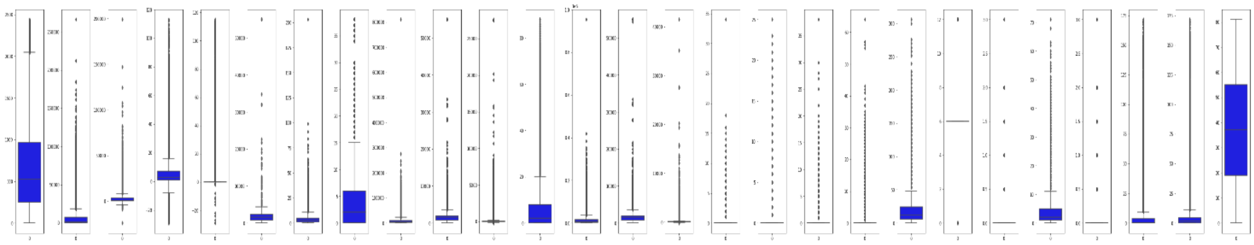
4. Distribution Plot: -



Observations:

- Not considering skewness of categorical data columns.
- aon, daily_decr30, rental30, 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, fr_ma_rech90, sumamnt_ma_rech90, medianamnt_ma_rech90, medianmarechprebal90, cnt_da_rech30, fr_da_rech30, cnt_da_rech90, fr_da_rech90, amnt_loans30, maxamnt_loans30, medianamnt_loans30, cnt_loans90, medianamnt_loans90, payback30, payback90 columns have skewness.

5. Box Plot: -



Observations:

- 'payback30', 'amnt_loans30','payback90','aon', 'daily_decr30', 'rental30', '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', 'fr_ma_rech90', 'sumamnt_ma_rech90','medianamnt_ma_rech90', 'medianmarechprebal90', 'cnt_da_rech30','fr_da_rech30', 'cnt_da_rech90', 'fr_da_rech90' and 'amnt_loans30' Column has outliers.

- **Interpretation of the Results**

Below is the list of highly influencing features or variables to predict the sales price of the house.

1. cnt_ma_rech30 = This feature shows the number of times the account got recharged in last 30 days which is important since it shows whether the person is using his balance or not.
2. sumamnt_ma_rech30 = This feature shows the total amount of recharge in main account over last 30 days as it shows the person's exact expenditure over a balance.
3. cnt_ma_rech90 = This feature shows the number of times the account got recharged in last 90 days which is important since it shows whether the person is using his balance or not.
4. sumamnt_ma_rech90 = This feature shows the total amount of recharge in main account over last 90 days as it shows the person's exact expenditure over a balance.

CONCLUSION

- ## Key Findings and Conclusions of the Study
- **Model Findings**
  1. LogisticsRegression - Cross Validation Score is 75.81%, Accuracy Score of Train Result is 75.63% and Test Result is 76.53%
  2. RandomForestClassifier - Cross Validation Score is 92.63%, Accuracy Score of Train Result is 99.99% and Test Result is 92.82%
  3. DecisionTreeClassifier - Cross Validation Score is 87.65%, Accuracy Score of Train Result is 100.00% and Test Result is 87.96%
  4. GradientBoostingClassifier - Cross Validation Score is 88.72%, Accuracy Score of Train Result is 89.31% and Test Result is 89.16%
  5. Support Vector Classifier - Cross Validation Score is 84.68%, Accuracy Score of Train Result is 84.90% and Test Result is 85.05%
  6. KNeighborsClassifier - Cross Validation Score is 87.66%, Accuracy Score of Train Result is 91.41% and Test Result is 86.96%

- **Model Selection**
  1. Selecting GradientBoostingClassifier model for hyper parameter tunning since the Accuracy score i.e., 89.31% and test scores i.e., 89.16% are greater and close to each other.
  2. Precision, Recall and F1-score score of the model is 89%.
  3. Cross Validation Score of the model is 88.72%

- **Hyper Parameter tunning**
  1. Using below parameters for hyper parameter tunning.
     - 'loss': ['deviance', 'exponential']
     - 'learning_rate': np.arange(0.1,0.9,0.1)
     - 'min_samples_split': range(1,5)
     - 'min_samples_leaf': range(1,5)
  2. Using GridSearchCV to find best parameters to train the model.
  3. Best parameters from hyper parameter tunning are min_samples_leaf = 3, min_samples_split = 2, learning_rate = 0.8, loss = ‘deviance’
  4. Accuracy for train data increased from 89.31% to 92.27%.
  5. Accuracy for test data increased from 89.16% to 91.71%.
  6. Precision, Recall and F1-score increased from 89% to 92%.
  7. New Cross Validation Score of the model is 91.25%.

- **AUC ROC Curve**

  AUC Score of GradientBoostingClassifier is 97%.

- **Learning Outcomes of the Study in respect of Data Science**
  1. Data Cleaning helps to convert unorganized and unstructured data into structured data which will be used to make findings.
  2. Data visualization helps understand and analyse the data.
  3. Model building helps to predict outcomes, in this case GradientBoostingClassifier model fits perfect for this dataset.
  4. While doing pre-processing the high VIF problem arises as many highly co-related features have high VIF score.

- **Limitations of this work and Scope for Future Work**
  1. There are 36 features present in the dataset however due to pre-processing and visualization we have cutdown some features, hence this could become a disadvantage in the future as we update the dataset and there is a possibility that we may lose some important information.
  2. It is necessary to keep an eye on new and updated data to further train the model and make decision as per new data.