MICRO CREDIT DEFAULTER PROJECT

Submitted by:

Abhijit Sarkar

ACKNOWLEDGMENT

I would like to express my gratitude for the opportunity and would like to thank Flip Robo Technologies for giving me an opportunity to work on this Given project. While going through this project I could find the economy facts and a conceptual thinking's of the consumer behind and consumptions of the customers. I am very grateful to DATA TRAINED team for providing me the adequate Trainings which actually helped me a lot to complete this project in the given time. I took the help from Mr. Mohd Kashif where I faced the problem. Moreover, I took the help of Google, Kaggle, Sklearn library and panda's library for solving this data set.

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. 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.

➢ CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM.

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

➢ REVIEW OF LITERATURE.

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).

➢ MOTIVATION FOR THE PROBLEM UNDERTAKEN

Here We need to build a model which can be used to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan. 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.

ANALYTICAL PROBLEM FRAMING.

- ## Mathematical/ Analytical Modelling of the Problem

Firstly, I found Columns with No Missing Values, But While checking Unique contents I saw zeros in many columns, I thought of checking the rows if I have only few rows contains zeros but while checking I saw almost maximum rows with zeros, I replaced those zeros with np.nan and found that almost maximum contains are missing. Even I could see 4 to 5 rows having 90 percentage data missing. The Imputation of those rows can make the prediction model a complete bias. Finally, the columns were dropped. To get better insight on the features I have used plotting like distribution plot, bar plot, reg plot and cat plot, strip plot, count plot as well. With these plotting I was able to understand the relation between the features in better manner. Also, I found outliers and skewness in the dataset so I removed outliers using Zscore and I removed skewness using yeo-Johnson method. I even used Standard Scaler to bring the data under one scale. PCA didn't work well so I removed that, I have used all the Logistic regression models and other classification models while building model then turned the best model and saved the best model.

- ## Data Sources and their formats

The data is being collected from my internship company" Flip Robo" and the dataset is in csv format. The Data Set contains 36 columns and 209593 rows.
The size of the data is 7545348.

Columns contains

- Float Values - 21 Columns
- Integer Values - 12 Columns

- Object Values - 3 Columns
- Memory Used: 58 MB

- **Data Pre-processing Don.**

These Steps:

1. Loading Data Set, Locating Null Values
2. Finding the unique values in categorical and numerical columns.
3. Finding Data Missing percentage
4. Finding nan values and replacing nan values.
5. Finding duplicated values in columns rows.
6. Use encoding was not required.
7. Feature Extraction was done here.

- Data Inputs- Logic- Output Relationships

X variables plays a very import role in machine learning for the Prediction of Target variable. Here 'LABEL 'is the target variable on which the predictions are being made.

I used the following to determine the relationship between variable:

- I have used Catplot for each pair of categorical features that shows the relation with the Target Variable. Used (Box plot and Distplot to determine the relations) And also, for continuous numerical variables I have used Implot, scatterplot to show the relationship between a continuous numerical variable and target variable.
- Used univariate, Bivariate Graph to check for relations.

By the Use of these Graph, I Uncovered the is a relationship between continuous numerical variable and The Target Variable.

- Hardware and Software Requirements and Tools Used

Hardware:


Device specifications

Copy

Device name	DataScientist
Processor	11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz 3.30 GHz
Installed RAM	16.0 GB (15.7 GB usable)
Device ID	4F7BEEF5-7469-44D4-B20A-DB8ACB2591EC
Product ID	00327-30000-00000-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Software Used:

- I Used Jupiter Note Book.
- Microsoft Office 2020
- Windows 11 OS

Library used:

1. NumPy
2. Pandas
3. Seaborn
4. Matplotlib
5. SciPy
6. Sklearn
7. Pickle
8. Imbalance Learn

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  - I have used the simple imputation method to replace Null values.
  - To check outliers, I used boxplot.
  - To remove outliers, I have used Zscore.
  - To check skewness, I used distplot.
  - I remove skewness I have used the yeo-johnson method.
  - Use of Pearson's correlation coefficient to check the correlation between dependent and independent features.
  - Also, I have used standardization.

- ➢ Then followed by model building with all Classifiers and Logistic regression algorithms.

- Testing of Identified Approaches (Algorithms)

  - ➢ Logistic Regression
  - ➢ Decision Tree Classifier
  - ➢ Extra Trees Classifier
  - ➢ Random Forest Classifier
  - ➢ Gradient Boosting Classifier

- Run and evaluate selected models.

Model-Logistic Regression

```
In [111]: x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=73,test_size=0.20)

          #sent for training
          lg.fit(x_train,y_train)

          #predict(x_training data)
          pred_train=lg.predict(x_train)
          pred_test=lg.predict(x_test)

          #result
          print("Accuracy Training Score =",accuracy_score(y_train,pred_train)," Accuracy Test Score =",accuracy_score(y_test,pred_test),"\
          print("Training Confusion_Matrix \n",confusion_matrix(y_test,pred_test),'\n',"Testing Confusion_Matrix \n",confusion_matrix(y_tes
          print("Classification Report \n",classification_report(y_test,pred_test),'\n\n')
```

```
Accuracy Training Score = 0.7450151283868506  Accuracy Test Score = 0.7452297879300005

Training Confusion_Matrix
 [[29477  7186]
 [11507 25202]]
 Testing Confusion_Matrix
 [[29477  7186]
 [11507 25202]]

Classification Report
               precision    recall  f1-score   support

           0       0.72      0.80      0.76     36663
           1       0.78      0.69      0.73     36709

    accuracy                           0.75     73372
   macro avg       0.75      0.75      0.74     73372
weighted avg       0.75      0.75      0.74     73372
```

Model-1

Models Decision Tree

In [130]:

```python
#train and score
dtc.fit(x_train,y_train)
dtc_score=dtc.score(x_train,y_train)

#predict
pred_train=dtc.predict(x_train)
pred_test=dtc.predict(x_test)

#result
print("Training Score",dtc_score)
print("Accuracy Training Score =",accuracy_score(y_train,pred_train)," Accuracy Test Score =",accuracy_score(y_test,pred_test),"\
print("Training Confusion_Matrix \n",confusion_matrix(y_train,pred_train),"Testing Confusion_Matrix \n",confusion_matrix(y_test,p
print("Classification Report \n",classification_report(y_test,pred_test))
```

```
Training Score 0.9946709916589435
Accuracy Training Score = 0.9946709916589435  Accuracy Test Score = 0.8877092078722129

Training Confusion_Matrix
 [[146599    168]
 [  1396 145325]] Testing Confusion_Matrix
 [[32886  3777]
 [ 4462 32247]]
Classification Report
               precision    recall  f1-score   support

           0       0.88      0.90      0.89     36663
           1       0.90      0.88      0.89     36709

    accuracy                           0.89     73372
   macro avg       0.89      0.89      0.89     73372
weighted avg       0.89      0.89      0.89     73372
```

Model-2

Ensamble Model 1 Extra Trees Classifier ¶

```
# train and score
etc.fit(x_train,y_train)
etc_score=etc.score(x_train,y_train)

#predict
pred_train=etc.predict(x_train)
pred_test=etc.predict(x_test)

#result
print("Accuracy Training Score =",accuracy_score(y_train,pred_train)," Accuracy Test Score =",accuracy_score(y_test,pred_test),"\
print("Training Confusion_Matrix \n",confusion_matrix(y_train,pred_train),"Testing Confusion_Matrix \n",confusion_matrix(y_test,p
print("Classification Report \n",classification_report(y_test,pred_test))
```

```
Accuracy Training Score = 0.9946709916589435  Accuracy Test Score = 0.9300141743444366

Training Confusion_Matrix
 [[146599    168]
 [  1396 145325]] Testing Confusion_Matrix
 [[34363  2300]
 [ 2835 33874]]
Classification Report
               precision    recall  f1-score   support

           0       0.92      0.94      0.93     36663
           1       0.94      0.92      0.93     36709

    accuracy                           0.93     73372
   macro avg       0.93      0.93      0.93     73372
weighted avg       0.93      0.93      0.93     73372
```

Model-3

Model2: Random Forest Classifier

```
#random Forest Training and score
rfc.fit(x_train,y_train)
rfc_score=rfc.score(x_train,y_train)

#predict random Forest
pred_train=rfc.predict(x_train)
pred_test=rfc.predict(x_test)

#result random Forest
print("Accuracy Training Score =",accuracy_score(y_train,pred_train)," Accuracy Test Score =",accuracy_score(y_test,pred_test),"\
print("Training Confusion_Matrix \n",confusion_matrix(y_train,pred_train),"Testing Confusion_Matrix \n",confusion_matrix(y_test,p
print("Classification Report \n",classification_report(y_test,pred_test))
```

```
Accuracy Training Score = 0.9946301041269149  Accuracy Test Score = 0.9233222482690945

Training Confusion_Matrix
 [[146506    261]
 [  1315 145406]] Testing Confusion_Matrix
 [[33953  2710]
 [ 2916 33793]]
Classification Report
               precision    recall  f1-score   support

           0       0.92      0.93      0.92     36663
           1       0.93      0.92      0.92     36709

    accuracy                           0.92     73372
   macro avg       0.92      0.92      0.92     73372
weighted avg       0.92      0.92      0.92     73372
```

Model-4

Gradient Boosting Classfier ¶

```
5]: #train and score gradient Boosting Classifier
gbc.fit(x_train,y_train)
gbc_score=gbc.score(x_train,y_train)

#predict Gradient Boosting Classifier
pred_train=gbc.predict(x_train)
pred_test=gbc.predict(x_test)

#result gradient Boosting classifier
print("Accuracy Training Score =",accuracy_score(y_train,pred_train)," Accuracy Test Score =",accuracy_score(y_test,pred_test),"\
print("Training Confusion_Matrix \n",confusion_matrix(y_train,pred_train),"Testing Confusion_Matrix \n",confusion_matrix(y_test,p
print("Classification Report \n",classification_report(y_test,pred_test))
```

```
Accuracy Training Score = 0.858672245543259  Accuracy Test Score = 0.8586381725999018

Training Confusion_Matrix
 [[130948  15819]
 [ 25659 121062]] Testing Confusion_Matrix
 [[32717  3946]
 [ 6426 30283]]
Classification Report
               precision    recall  f1-score   support

           0       0.84      0.89      0.86     36663
           1       0.88      0.82      0.85     36709

    accuracy                           0.86     73372
   macro avg       0.86      0.86      0.86     73372
weighted avg       0.86      0.86      0.86     73372
```

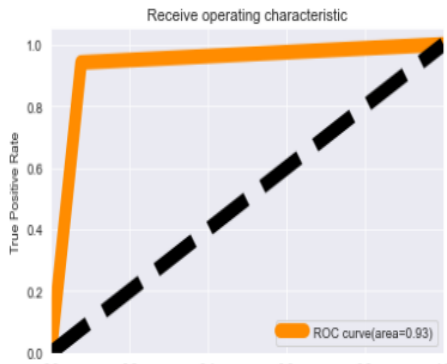
Hyper-Parameter on Extra Trees Classifier

```
[186]: etc=ExtraTreesClassifier(criterion='log_loss', max_features= 'log2', min_samples_split= 3 ,n_estimators= 200, n_jobs= 5,verbose=
                         max_depth=1036,min_samples_leaf=1,min_impurity_decrease=0.00000000001)

#train and score
etc.fit(x_train,y_train)
etc_score=etc.score(x_train,y_train)

#predict
pred_train2=etc.predict(x_train)
pred_test2=etc.predict(x_test)

#result
print("Training_Score",etc_score)
print("Training Accuracy_score ",accuracy_score(y_train,pred_train2)," Testing Accuracy_score ",accuracy_score(y_test,pred_test2
print("Training Confusion_Matrics\n",confusion_matrix(y_train,pred_train2)," Testing Confusion Matrics\n ",confusion_matrix(y_te
print("Classification Report \n",classification_report(y_test,pred_test2))
```

```
Training_Score 0.9935670282941722
Training Accuracy_score  0.9935670282941722  Testing Accuracy_score  0.9320176634138363
Training Confusion_Matrics
 [[146552    215]
 [  1673 145048]]  Testing Confusion Matrics
  [[34546  2117]
 [ 2871 33838]]
Classification Report
               precision    recall  f1-score   support

           0       0.92      0.94      0.93     36663
           1       0.94      0.92      0.93     36709

    accuracy                           0.93     73372
   macro avg       0.93      0.93      0.93     73372
weighted avg       0.93      0.93      0.93     73372
```

Graph:

```python
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds= roc_curve(pred_test2,y_test)
roc_auc = auc(fpr,tpr)

plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=10,label='ROC curve(area=%0.2f)' % roc_auc)

plt.plot([0,1],[0,1],color='black',lw=10,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receive operating characteristic")
plt.legend(loc="lower right")
plt.show()
```



Saving Model and Loading Model:

Saving Model

In [190]:
```python
import pickle
filename='micro_fin.pkl'
pickle.dump(etc,open(filename,'wb'))
```

Loading Model

In [191]:
```python
# loading pack file
pickled_model= pickle.load(open(filename,'rb'))
result=pickled_model.score(x_test,y_test)
result
```

Out[191]: 0.9320176634138363

In [192]:
```python
result*100
```

Out[192]: 93.20176634138363

In [193]:
```python
conclude=pd.DataFrame([pickled_model.predict(x_test)[:],pred_test2[:]],index=['Predicted','Original'])
conclude
```

Out[193]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
Predicted	0	1	0	0	1	1	1	0	1	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0	0	1	1
Original	0	1	0	0	1	1	1	0	1	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0	0	1	1

- ## Key Metrics for success in solving problem under consideration.

```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

The Following Metrics were Used:

1. Accuracy score
2. Classification report
3. Confusion Matrix
4. F1 score
5.

```
#result
print("Training Score",dtc_score)
print("Accuracy Training Score =",accuracy_score(y_train,pred_train)," Accuracy Test Score =",accuracy_score(y_test,pred_test),"\
print("Training Confusion_Matrix \n",confusion_matrix(y_train,pred_train),"Testing Confusion_Matrix \n",confusion_matrix(y_test,p
print("Classification Report \n",classification_report(y_test,pred_test))
```

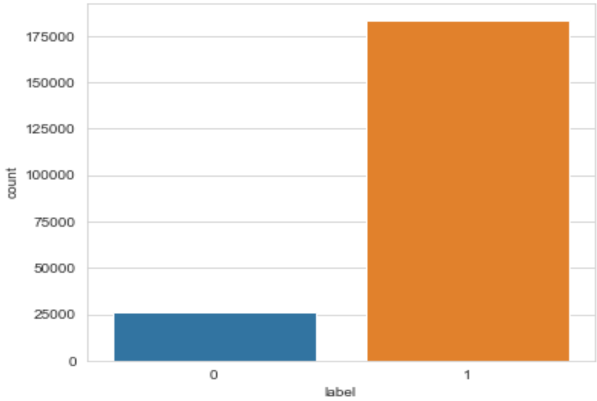
- ## Visualizations

Shows counts of observation for Target Variable.

Univariate Analysis

```
plt.figure(figsize=(6,5))
sns.set_style("whitegrid")
sns.countplot(x='label',data=df)
df['label'].value_counts()
```

```
1    183430
0     26162
Name: label, dtype: int64
```

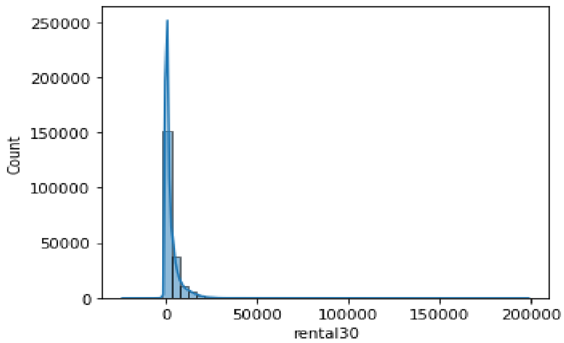


- 1 Shows 183430 Maximum Payments are Clear
- 0 Shows 26162 Defaulter

Distribution:

```
plt.figure(figsize=(5,4))
sns.histplot(x='rental30',data=df,kde=True,bins=50)
df['rental30'].min(),"and",df['rental30'].max()
```

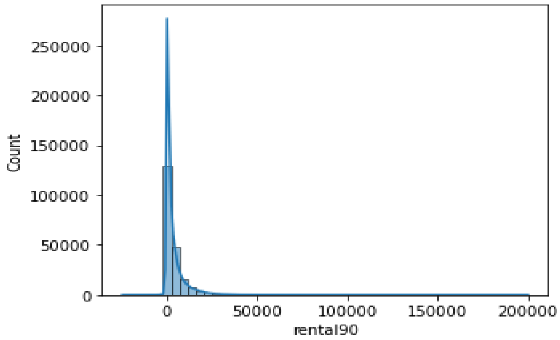
```
(-23737.14, 'and', 198926.11)
```



- Average main account balance over last 30 days Lies (-23737.14, 'and', 198926.11)

```
plt.figure(figsize=(5,4))
sns.histplot(x='rental90',data=df,kde=True,bins=50)
df['rental90'].min(),"and",df['rental90'].max()
```

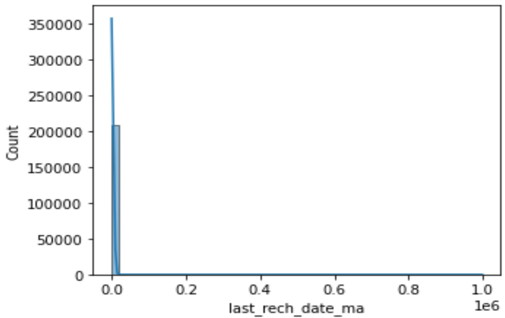
```
(-24720.58, 'and', 200148.11)
```



- Average main account balance over last 90 days shows between (-24720.58, 'and', 200148.11)

```
plt.figure(figsize=(5,4))
sns.histplot(x='last_rech_date_ma',data=df,kde=True,bins=50)
df['last_rech_date_ma'].min(),"and",df['last_rech_date_ma'].max()
```

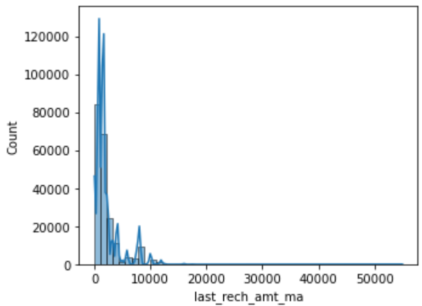
```
(-29.0, 'and', 998650.377732702)
```



- Number of days till last recharge of main account lies between -29.0, 'and', 998650.377732702

```
plt.figure(figsize=(5,4))
sns.histplot(x='last_rech_amt_ma',data=df,kde=True,bins=50)
df['last_rech_amt_ma'].min(),"and",df['last_rech_amt_ma'].max()
```

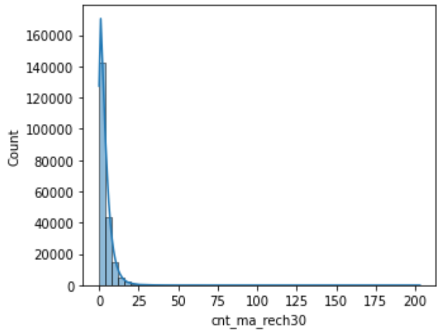
```
(0, 'and', 55000)
```



- Number of days till last recharge of data account lies between 0 to 55000

```
plt.figure(figsize=(5,4))
sns.histplot(x='cnt_ma_rech30',data=df,kde=True,bins=50)
df['cnt_ma_rech30'].min(),"and",df['cnt_ma_rech30'].max()
```

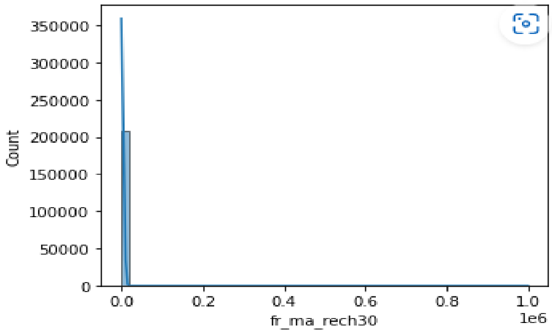
```
(0, 'and', 203)
```



- Number of times main account got recharged in last 30 days lies between (0, 'and', 203)

```
plt.figure(figsize=(5,4))
sns.histplot(x='fr_ma_rech30',data=df,kde=True,bins=50)
df['fr_ma_rech30'].min(),"and",df['fr_ma_rech30'].max()
```

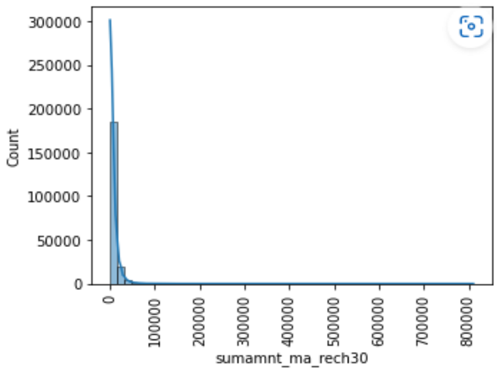
```
(0.0, 'and', 999606.368131936)
```



Frequency of main account recharged in last 30 days lies between (0.0, 'and', 999606.368131936)

```
plt.figure(figsize=(5,4))
sns.histplot(x='sumamnt_ma_rech30',data=df,kde=True,bins=50)
print(df['sumamnt_ma_rech30'].min(),"and",df['sumamnt_ma_rech30'].max())
plt.xticks(rotation=90,fontsize=10)
plt.tight_layout()
```

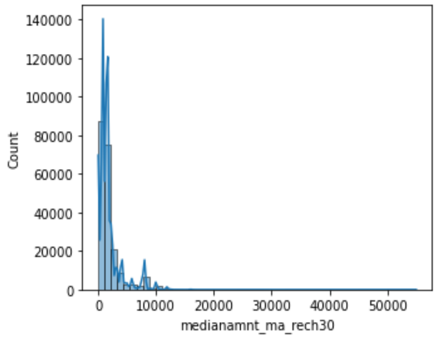
```
0.0 and 810096.0
```



- Total amount of recharge in main account over last 30 days (in Indonesian Rupiah) lies between 0.0 and 810096.0

```
plt.figure(figsize=(5,4))
sns.histplot(x='medianamnt_ma_rech30',data=df,kde=True,bins=50)
df['medianamnt_ma_rech30'].min(),"and",df['medianamnt_ma_rech30'].max()
```

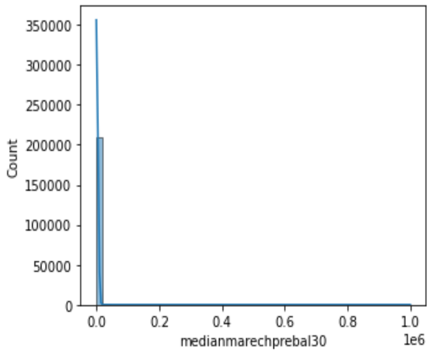
```
(0.0, 'and', 55000.0)
```



- Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)lies between (0.0, 'and', 55000.0)

```
plt.figure(figsize=(5,4))
sns.histplot(x='medianmarechprebal30',data=df,kde=True,bins=50)
df['medianmarechprebal30'].min(),"and",df['medianmarechprebal30'].max()
```

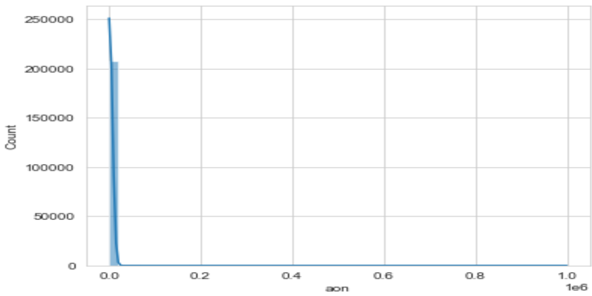
```
(-200.0, 'and', 999479.419318959)
```



- Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah) lies between -200 'and', 999479.419318959

```
37]: plt.figure(figsize=(6,5))
     sns.histplot(x='aon',data=df,kde=True,bins=50)
     df['aon'].min(),"and",df['aon'].max()
```

```
37]: (-48.0, 'and', 999860.755167902)
```

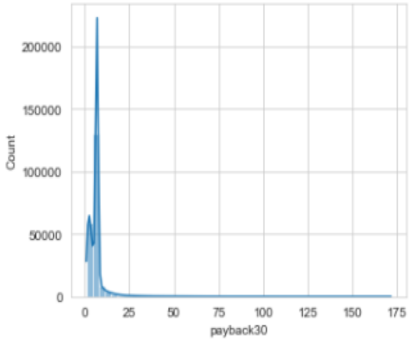


Distribution lies Maximum at 0

- age on cellular network in days distribution lies between (-48.0, 'and', 999860.755167902)

```
plt.figure(figsize=(5,4))
sns.histplot(x='payback30',data=df,kde=True,bins=50)
```

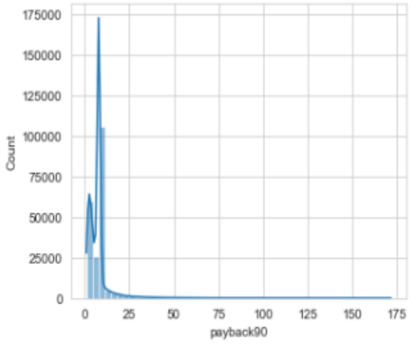
<AxesSubplot:xlabel='payback30', ylabel='Count'>



- Average payback time in days over last 30 days

```
plt.figure(figsize=(5,4))
sns.histplot(x='payback90',data=df,kde=True,bins=50)
```

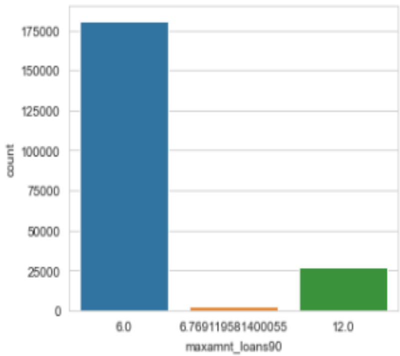
<AxesSubplot:xlabel='payback90', ylabel='Count'>



- Average payback time in days over last 90 days

```
plt.figure(figsize=(5,4))
sns.countplot(x='maxamnt_loans90',data=df)
df['maxamnt_loans90'].value_counts()
```

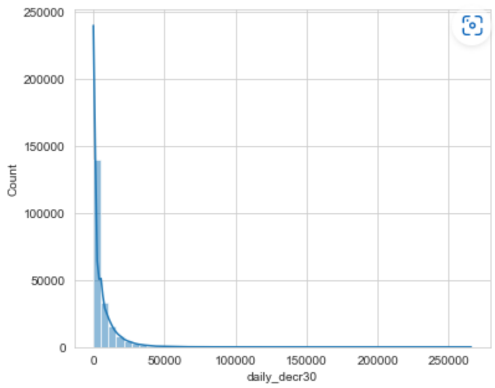
```
6.00000     180944
12.00000     26605
6.76912       2043
Name: maxamnt_loans90, dtype: int64
```



- Maximum amount of loan taken by the user in last 90 days 6.00000->180944, 12.00000 -> 26605, 6.76912-> 2043

```
plt.figure(figsize=(6,5))
sns.histplot(x='daily_decr30',data=df,kde=True,bins=50)
df['daily_decr30'].min(),"and",df['daily_decr30'].max()
```

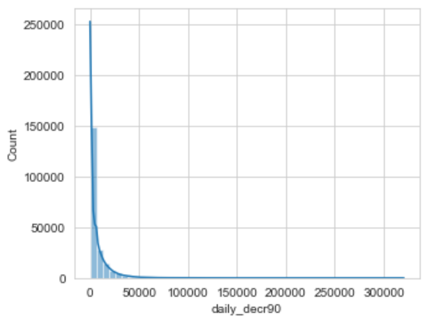
```
(-93.0126666666667, 'and', 265926.0)
```



- Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah) Lies between (-93.0126666666667, 'and', 265926.0)

```
plt.figure(figsize=(5,4))
sns.histplot(x='daily_decr90',data=df,kde=True,bins=50)
df['daily_decr90'].min(),"and",df['daily_decr90'].max()
```

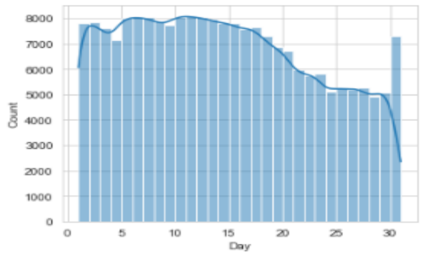
```
(-93.0126666666667, 'and', 320630.0)
```



- Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah),Lies Between (-93.0126666666667, 'and', 320630.0)

```
plt.figure(figsize=(5,4))
sns.histplot(x='Day',data=df,kde=True,bins=30)
df['Day'].value_counts()[:5]
```

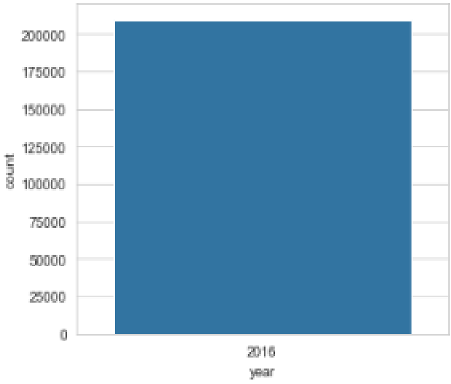
```
11    8092
10    8050
6     8030
12    8028
7     8026
Name: Day, dtype: int64
```



- High Observation in the 11th and 12th of the Month

```python
plt.figure(figsize=(5,4))
sns.countplot(x='year',data=df)
df['year'].value_counts()
```

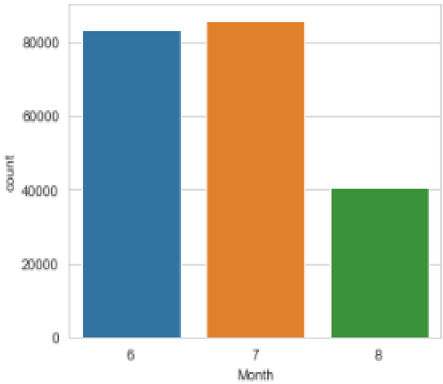
```
2016    209592
Name: year, dtype: int64
```



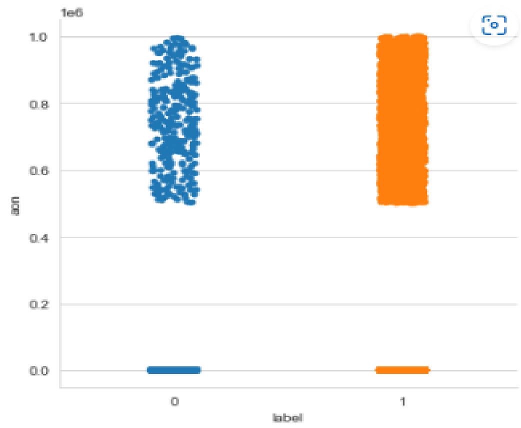
All observation where taken in 2016

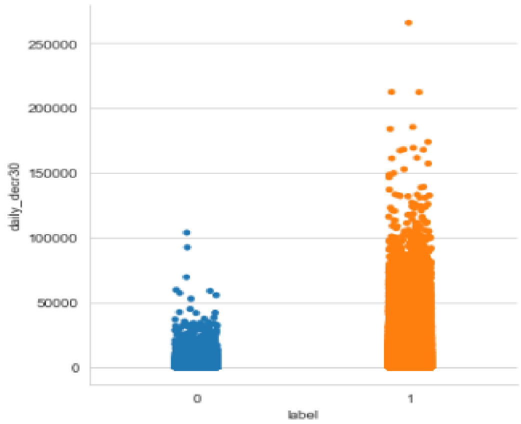
```python
plt.figure(figsize=(5,4))
sns.countplot(x='Month',data=df)
df['Month'].value_counts()
```

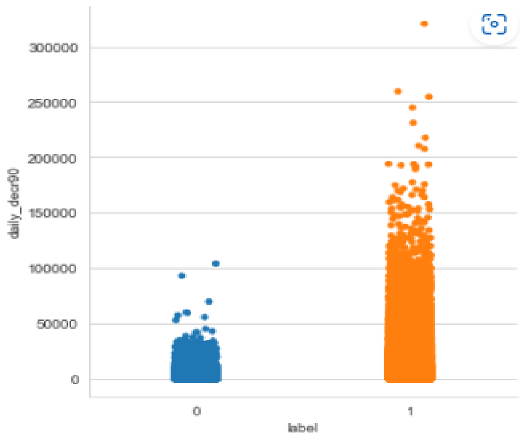
```
7    85764
6    83154
8    40674
Name: Month, dtype: int64
```

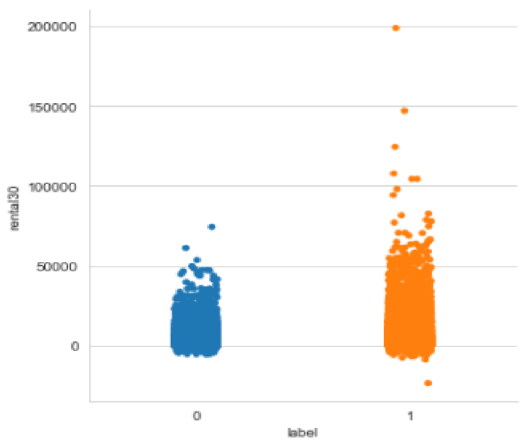


- High observation 7th Month

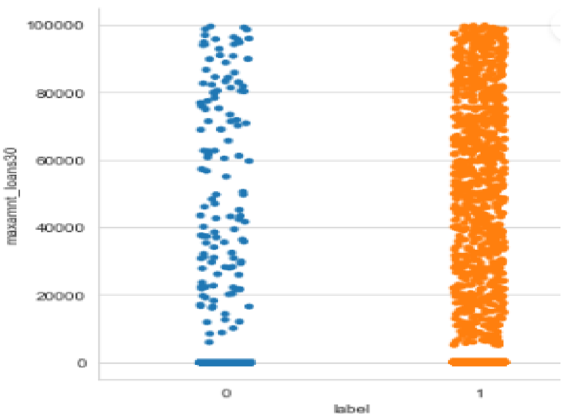

1e6
1.0
0.8
0.6
0.4
0.2
0.0
aon
0
1
label

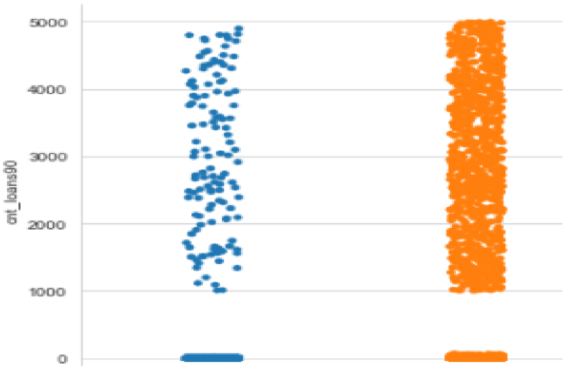

250000
200000
150000
100000
50000
0
daily_decr30
0
1
label

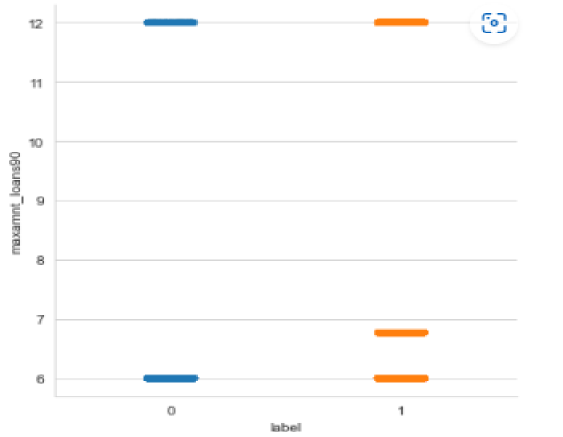

300000
250000
200000
150000
100000
50000
0
daily_decr90
0
1
label

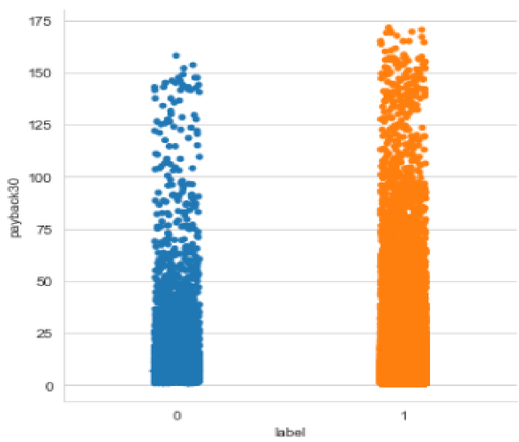

200000
150000
100000
50000
0
rental30
0
1
label

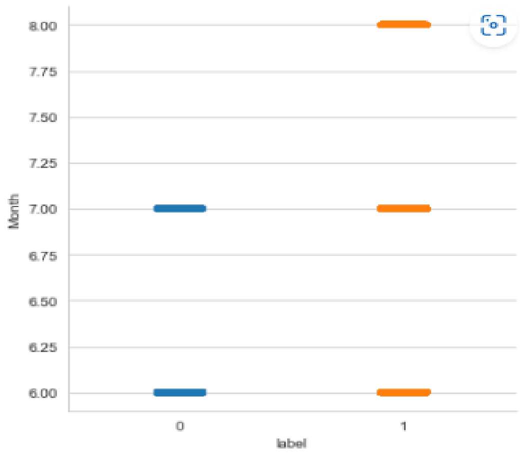
label

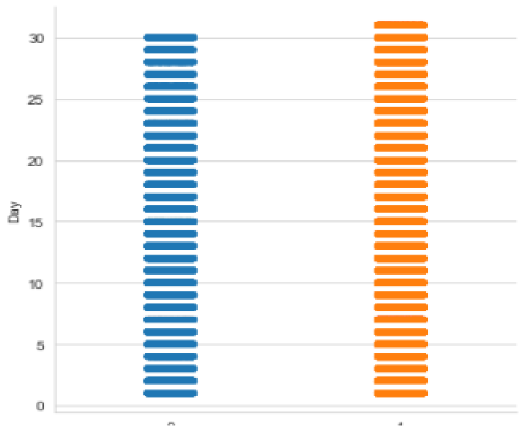

maxamnt_loans30
100000
80000
60000
40000
20000
0
0
1
label

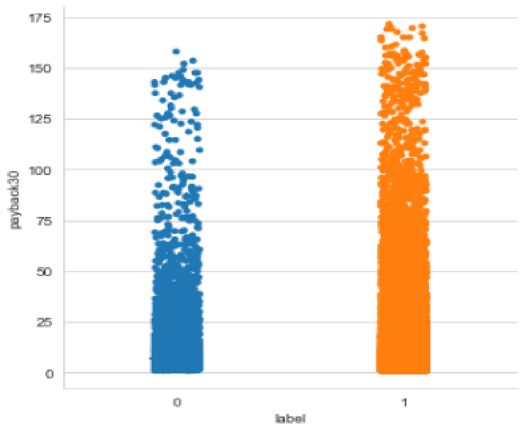

cnt_loans90
5000
4000
3000
2000
1000
0

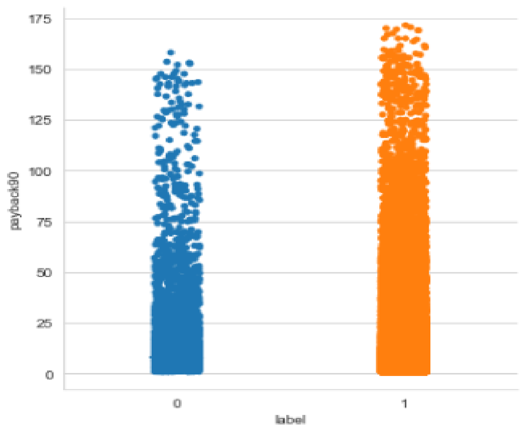

maxamnt_loans90
12
11
10
9
8
7
6
0
1
label


payback30
175
150
125
100
75
50
25
0
0
1
label


Month
8.00
7.75
7.50
7.25
7.00
6.75
6.50
6.25
6.00
0
1
label

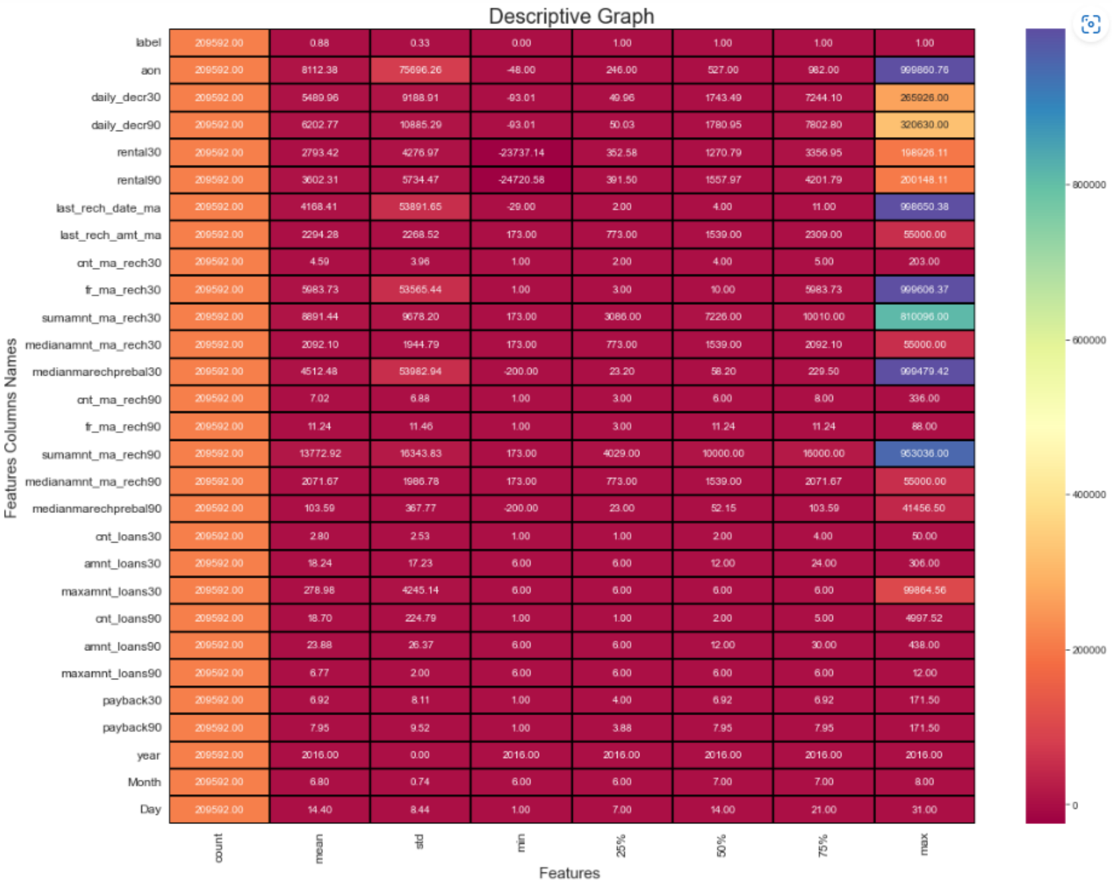

Day
30
25
20
15
10
5
0


payback30
175
150
125
100
75
50
25
0
0
1
label


payback90
175
150
125
100
75
50
25
0
0
1
label

2125

Descriptive Graph:



Few Observations:

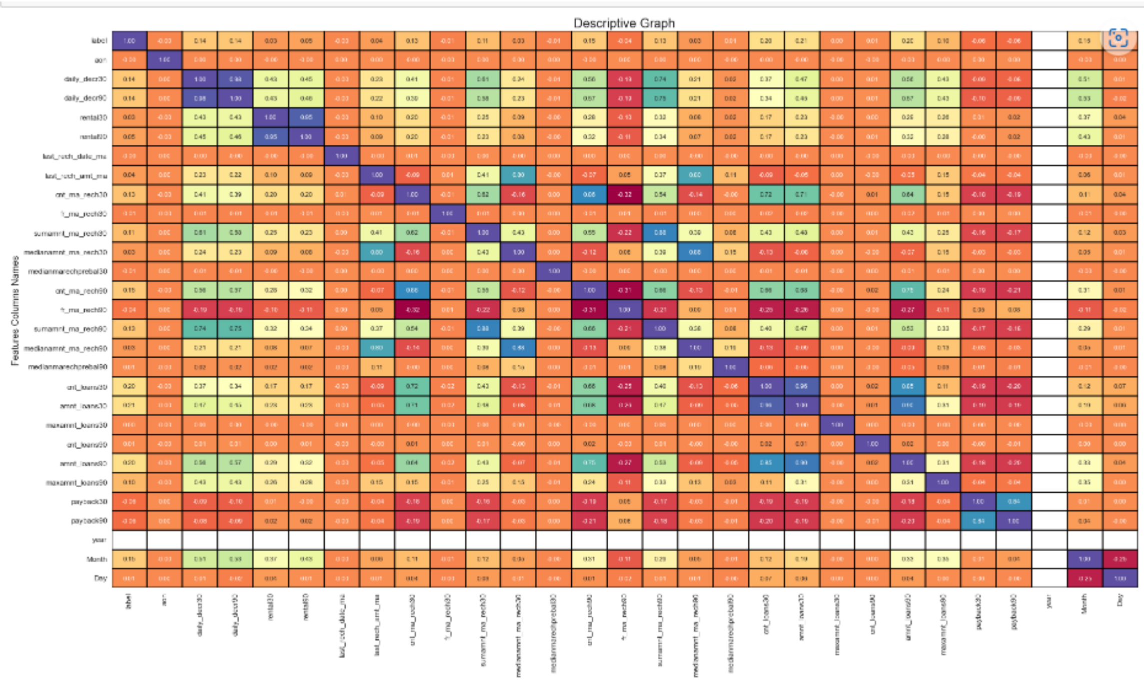
1. Null Values- All values are intact (No Null Values)

2. Right Skew- aon, daily_decr30, daily_decr90, rental30, rental90, last_rech_date_ma, 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_loans30 amnt_loans30 maxamnt_loans30 cnt_loans90 amnt_loans90 maxamnt_loans90 payback30 payback90 year Month Day

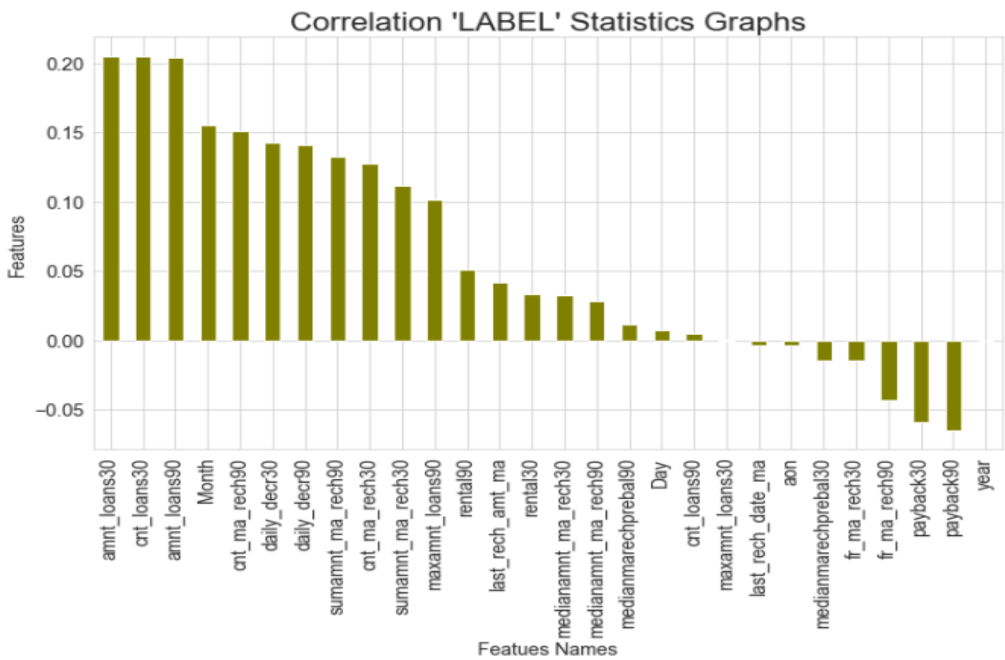
3. Left Skew- Null
4. Standard Deviation-aon daily_decr30 daily_decr90 rental30 rental90 last_rech_date_ma last_rech_amt_ma cnt_ma_rech30 fr_ma_rech30 sumamnt_ma_rech30 medianamnt_ma_rech30 medianmarechprebal30

5. Outliers- aon, 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_loans30 amnt_loans30 maxamnt_loans30 cnt_loans90 amnt_loans90 maxamnt_loans90 payback30 payback90 year Month Day

Correlation




```python
plt.figure(figsize=(10,8))
df.corr()['label'].sort_values(ascending=False).drop(['label']).plot(kind='bar',color='olive')
plt.title("Correlation  'LABEL'  Statistics Graphs",fontsize=22)
plt.xlabel("Featues Names",fontsize=14)
plt.ylabel('Features',fontsize=14)
plt.xticks(rotation=90,fontsize=14)
plt.yticks(rotation=0,fontsize=14)
plt.tight_layout()
```



Observation Shows:

- payback90 is 6 percent negatively correlated with the target variable which is extreamly weak.
- payback30 is 6 percent negatively correlated with the target variable which is extreamly weak.
- fr_ma_rech90 is 4 percentage negatively correlated with the target variable.
- fr_ma_rech30 is 1 percentage negatively correlated with the target variable.
- medianmarechprebal30 is 1 percentage negatively correlated with the target variable.
- aon is 0 percentage negatively correlated with the target variable.
- last_rech_date_ma is 0 percentage negatively correlated with the target variable.
- maxamnt_loans30 is 0 percentage negatively correlated with the target variable.
- cnt_loans90 is 1 percentage positively correlated with the target variable.
- Day is 1 percentage positively correlated with the target variable.
- medianmarechprebal90 is 1 percentage positively correlated with the target variable.
- medianamnt_ma_rech90 is 3 percentage positively correlated with the target variable.
- medianamnt_ma_rech30 is 3 percentage positively correlated with the target variable.
- rental30 is 3 percentage positively correlated with the target variable.
- last_rech_amt_ma is 4 percentage positively correlated with the target variable.
- rental90 is 5 percentage positively correlated with the target variable.
- maxamnt_loans90 is 10 percentage positively correlated with the target variable.
- sumamnt_ma_rech30 is 11 percentage positively correlated with the target variable.
- cnt_ma_rech30 is 13 percentage positively correlated with the target variable
- sumamnt_ma_rech90 is 13 percentage positively correlated with the target variable.
- daily_decr90 is 14 percentage positively correlated with the target variable.
- daily_decr30 is 14 percentage positively correlated with the target variable.
- cnt_ma_rech90 is 15 percentage positively correlated with the target variable.
- Month is 15 percentage positively correlated with the target variable.
- amnt_loans90 is 20 percentage positively correlated with the target variable.
- cnt_loans30 is 20 percentage positively correlated with the target variable.
- amnt_loans30 is 21 percentage positively correlated with the target variable which is very strongly correlated..
- label is 100 percentage positively correlated with the target variable.
- year is No correlation with the target variable.

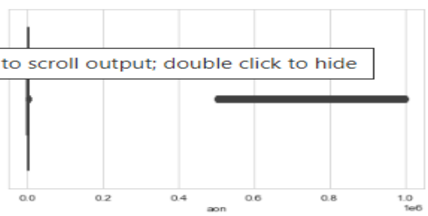
NOTE

- Payback is very negatively correlated with target variable.
- Amnt Loan is 21 percent positively correlated with target variable.
- Label is 100 percentage correlated with target variable.
- Year is No correlation with the target Variable.

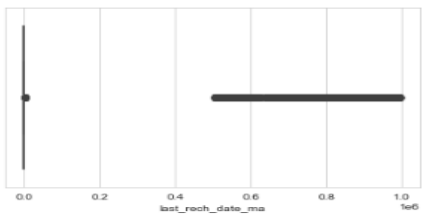
Outliers:
ALL float columns contains outliers:

```
*****************************************************
```

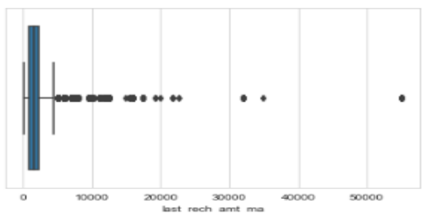
click to scroll output; double click to hide

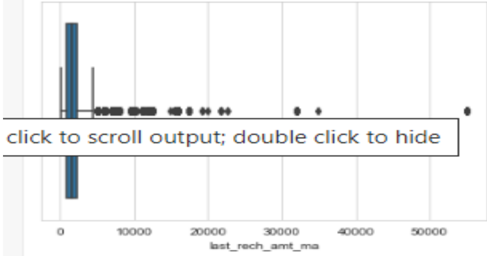



```
Column  last_rech_date_ma
*****************************************************
```

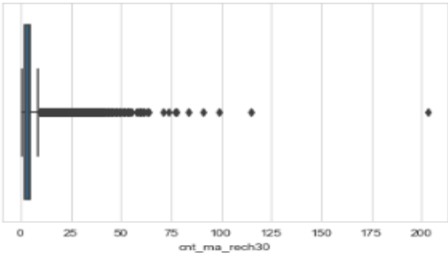



```
Column  last_rech_amt_ma
*****************************************************
```

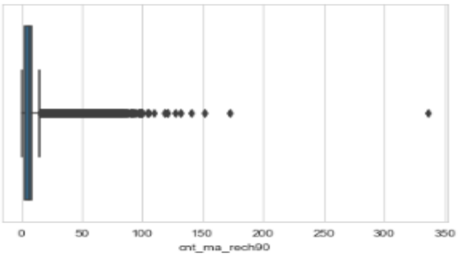


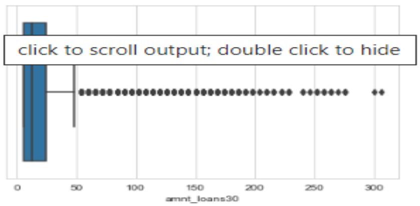



```
Column   cnt_ma_rech30
**************************************************
```

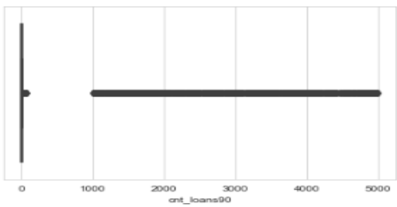



```
Column   cnt_ma_rech90
**************************************************
```

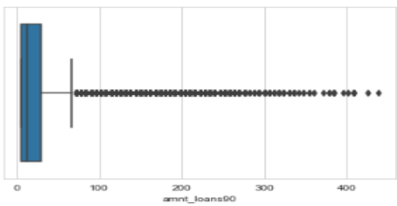





```
Column   cnt_loans90
**************************************************
```



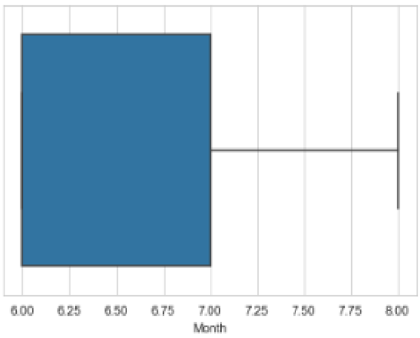

```
Column   amnt_loans90
**************************************************
```



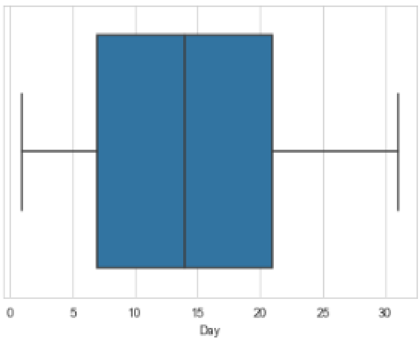

```
Column   Avg_Payback
**************************************************
```

year

```
Column  Month
****************************************
```

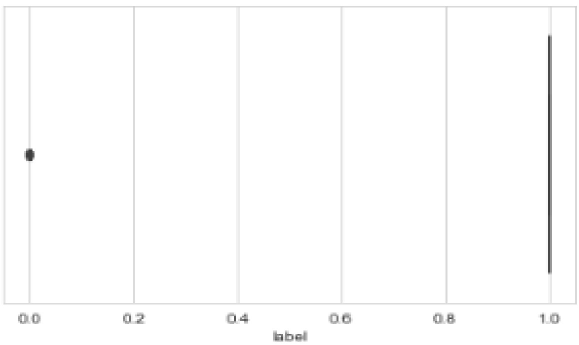



```
Column  Day
****************************************
```

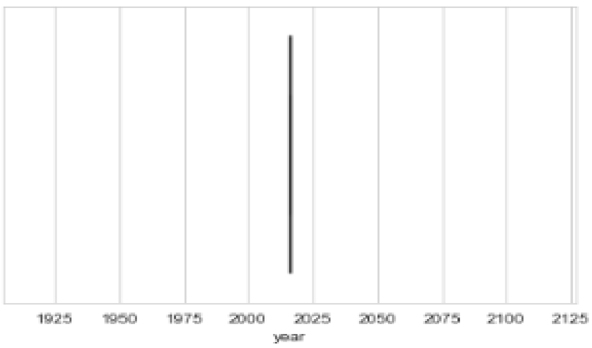


- Label Shows Outliers, Its a Target Variable

```
Column  label
****************************************
```




```
Column  year
****************************************
```

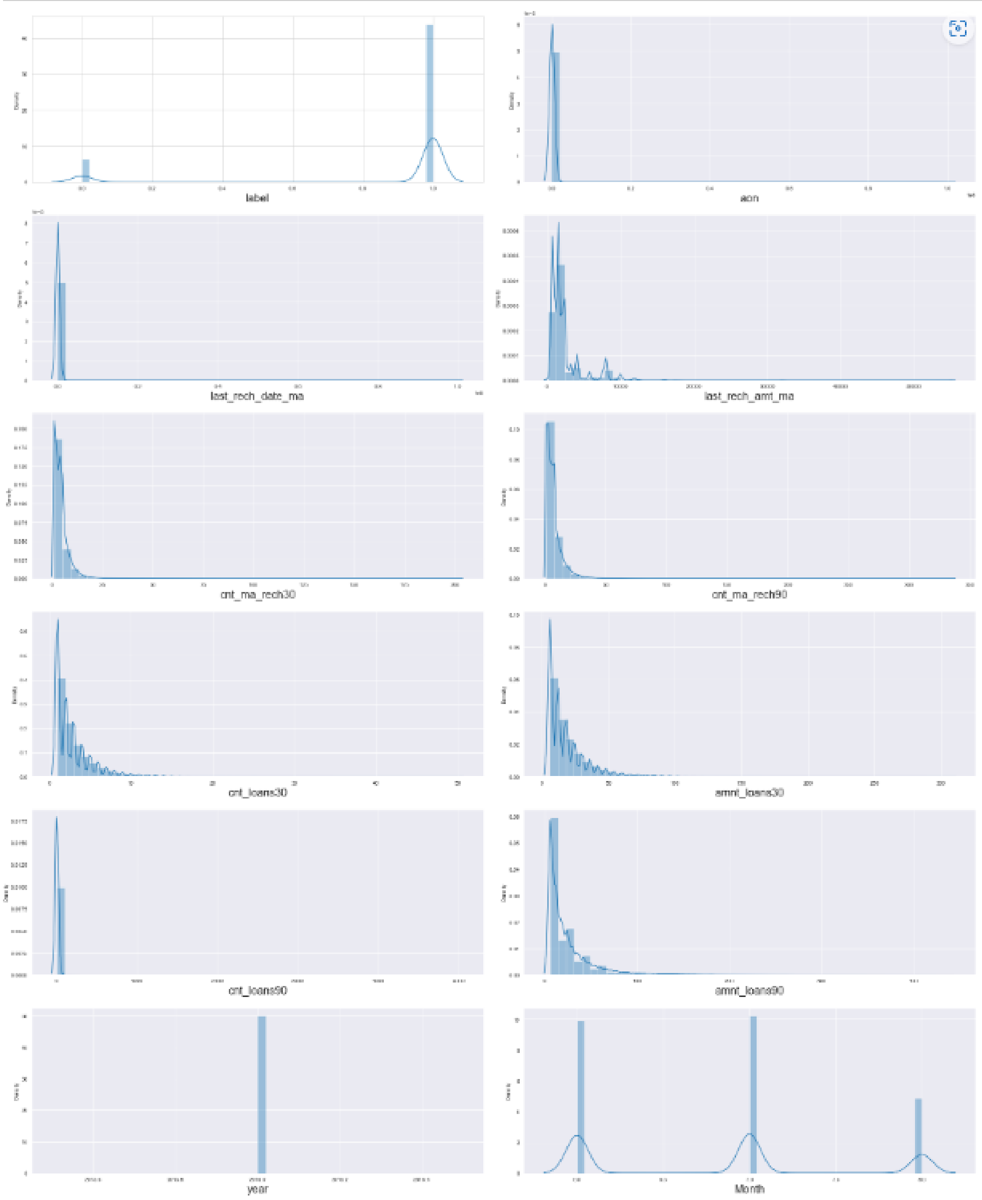


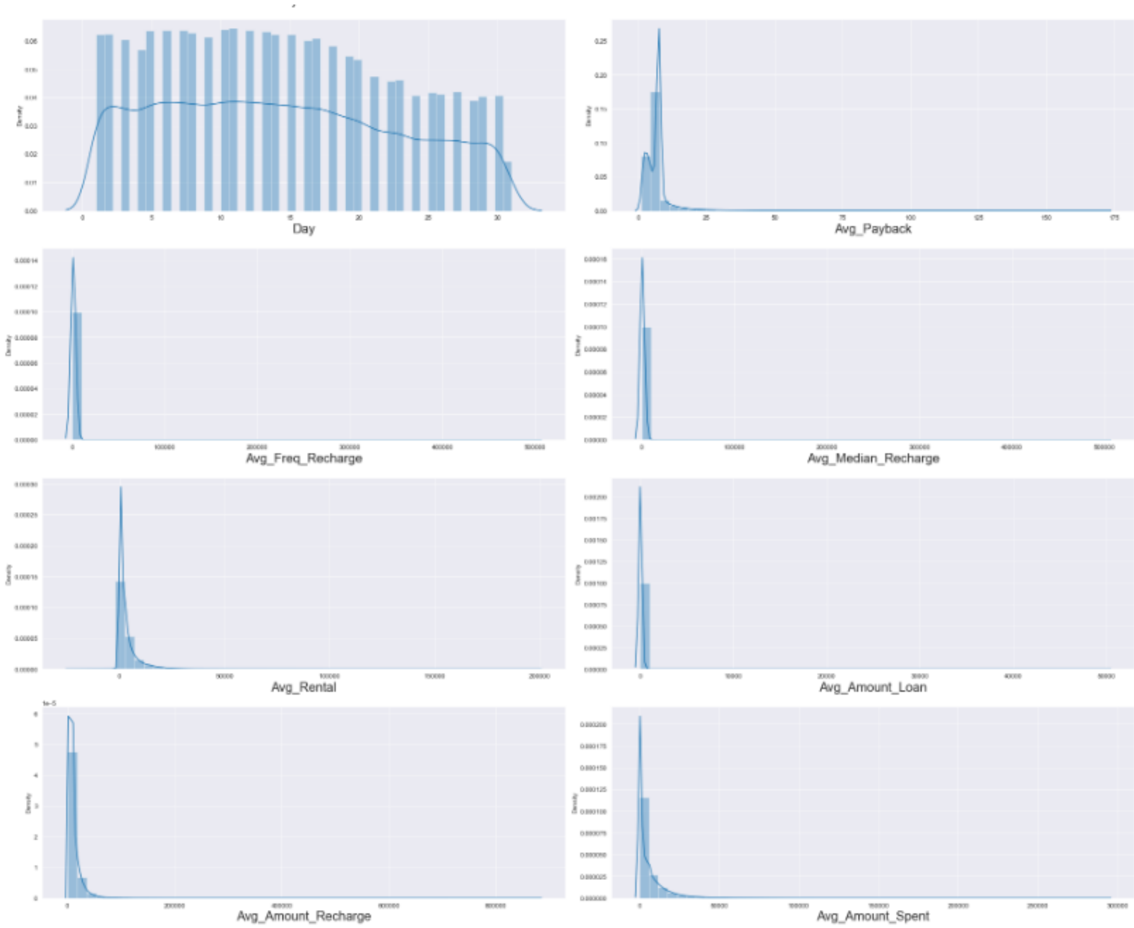

```
Column  Month
****************************************
```

Skewness in these columns

Normal Skewness Lies between +/-65

- cnt_loans30 2.758665
- amnt_loans30 3.020769
- amnt_loans90 3.168028
- cnt_ma_rech90 3.687818
- cnt_ma_rech30 3.746757
- Avg_Amount_Spent 4.066399
- last_rech_amt_ma 4.086556
- Avg_Rental 4.447820
- Avg_Amount_Recharge 5.749919
- Avg_Payback 8.462428
- aon 10.392923
- Avg_Freq_Recharge 14.711856
- Avg_Median_Recharge 14.760024
- last_rech_date_ma 14.779844
- cnt_loans90 16.593606
- Avg_Amount_Loan 17.656686





Observation Shows

- Maximum Graph lies beyond the normal Graph,
- Graph contains models that are Bi-Model,Tri-Modal and Multi-Modal Graph.
- Graph Shows Right and Left Skewness

➢ Interpretation of the Results

- ➢ This dataset was very special as it had a separate train and test datasets.
- ➢ Firstly, the datasets were having no null values but full of null zeros values which is firstly converted to null values. Then Imputation is done of that dataset. entries maximum columns so we have to be careful while going through the statistical analysis of the datasets.
- ➢ I found maximum numerical continuous columns were in relationship with target column.
- ➢ I notice a huge number of outliers and skewness in the data so we have chosen proper methods to deal with the outliers and skewness. If we ignore this outlier and skewness, we may end up with a bad model which has less accuracy.
- ➢ Then scaling both train and test dataset has a good impact like it will help the model not to get biased.
- ➢ We have to use multiple models while building model using train dataset as to get the best model out of it.
- ➢ Extra Trees and Random Forest were the best among all the models. Result received with both model is approx. above 92 percentage.
- ➢ Finally selected Extra Trees Classifier as CV_Score has better result and result was quite noticeable.

CONCLUSION

- Key Findings and Conclusions of the Study

In this project report, we have used machine learning algorithms to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan. We have mentioned the step-by-step procedure to analyse the dataset and finding the correlation Between the features. Thus, we can select the features which are not correlated to each other and are independent in nature. These feature set were then given as an input to five algorithms to predict output. Hence, we calculated the performance of each model using different performance metrics and compared them based on these metrics.

And saved the model as filename="micro_fin.pkl"

- Learning Outcomes of the Study in respect of Data Science

I found that the dataset was quite interesting to handle as it contains all types of data in it. Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analysed. New analytical techniques of machine learning can be used in property research. The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps to remove missing value and to replace null value and zero values with their respective mean, median or mode. This study is an exploratory attempt to use machine learning algorithms in finding the probability for each loan transaction against each customer. To conclude the application of Machine Learning in prediction is still at an early stage. I hope this study has moved a small step ahead in providing solution to the companies.

The changes I faced was when I saw a lot of zero and I was thinking to impute those zero but as the missing rows percentage were too high, so finally I had to drop those columns. Another Issue was while using binning process taking the median values helped me to achieve a good accuracy. I was actually thinking to get best out of those. However, I finally achieved a good model out of this.

➢ Limitations of this work and Scope for Future Work

This model doesn't predict future probability. The future will be unpredictable at all times due to this, the risk in investment in an import factor. This can predict for a time period and needs to be updated on a basis. Machine can predict when loan can be given but it can’t predict the intensity of the person who is grabbing the loan.