FLIP ROBO

Micro-Credit Defaulter Project

Submitted by:

Pankaj Suryawanshi

ACKNOWLEDGMENT

The dataset was provided by FilpRobo technologies. The dataset mainly deals with the information ‘Micro-Credit Defaulter Model’ . The project was done using python libraries using jupyter notebook. There libraries used for the processing of data and its analysis are numpy, pandas, sklearn . Along with the help from professionals at data trained for running various algorithms also from my SME at FlipRobo. Moreover, websites like https://towardsdatascience.com/ , https://scikit-learn.org/stable/ , were used to draw references and complete the project.

INTRODUCTION

Business Problem Framing

A telecom company has tied up with a MFI and have decided to help customers by providing a small amount of loan as credit on the mobile balance and the payback time is 5 days. As MFIs have turned out to be a great force in the field of economics as it serves as an effective tool for the low income group. The telecom industry is of great importance especially as it can reach out to the common masses. So the telecom industry coupled with the MFI can prove to be really effective tool reach out more amount of people and in a quicker and cleaner way and provide support to the really needy. But for that to happen the company needs to be able to make better business decisions when it comes to lending money to the people. It would be helpful for the company could somehow be able to identify the customers to whom they should be giving the loans to, i.e., they want to identify the defaulters from potential customers (people paying back the loan amount). This would prove to viable to sustain the business as it would help the companied to reach out the potential people and in turn keep the wheel of economy moving. So it becomes important to come up with a solution that will help the telecom services to separate the defaulters from the non-defaulters. A way subtle way to predict form the information already available with company seems a viable approach.

Conceptual Background of the Domain Problem

The company wants to explore a way to predict out defaulters from non-defaulters for future endeavours based on the current data they have collected . So as this project deals in such a sector the telecom company already have a way to differentiate among the each and everyone of its customers within their network via the phone numbers. It also has a way to differentiate among the people in the network based on who is actively using the network to make calls and who is using it less or not using it at all. Along with this a lot of other features like how many times the person recharges the account and what is the gap in between each recharge. Also which of its customers are paying back the loan amount and who are not paying back along with frequency of paybacks in a span of time like 30-90 days. So these data can help in predicting the future behaviours of the customers based on the current behaviour and this is a substantial for the telecom company in a way that is similar to the banking industry.

Review of Literature

The project comprises of first exploring the data. Finding the features and its data-type. For standard dataset ready for processing it must not contain any string type data. An ideal dataset is one that is numerical. The data set had to checked for discrepancies like finding out the outliers , checking the imbalanced features. Upon correcting such discrepancies further processing of the data can be done. The dataset had no missing values. Dataset had contained more than 2 lakh rows and

therefore can be considered for proper prediction. A number of algorithms /models had to be trained till the one with lowest error and highest score was found. In this case random forest from the scikit learn library proved to be the best with highest score of 95%.

Motivation for the Problem Undertaken

Finance is one of the biggest sectors of the world today and it has somehow always intrigued me and to be able to part of a project that is going to be directly responsible for that sector was a source of great satisfaction. The underlying factor for this problem lies in the behaviour of the customers for the telecom company. The company along with the MFI have come up with away for helping the low income class with a credit system that will be very helpful but for this enterprise to be successful the company should be able to predict the defaulter to ease the process and keep the business running forward.

Analytical Problem Framing

Analytical Modelling of the Problem

For a grouped dataset with so much information statistical approach was done. Firstly the mean, standard deviation(std.), median was calculated for checking distribution of the data. For all the datapoints that were not in alignment with desired std. ,z-score values were calculated simply by finding the difference between each datapoint and the mean and dividing the difference by the std. and later on removing all the values more than 3 . After which correlations among each of the feature was explored and depicted with help of a heatmap that plots values based on the correlation values. The 'label' column proved to be the dependent feature(Y) and the multiple independent features (X) and it was clear that it was a classification

problem that could be represent the relationship of the X and Y values through a sigmoid curve and for that reason logistic regression model was trained to predict the values. As the dataset contained a lot of outliers the best approach would be to classify the dataset with the help of Decision tree classifier as it breaks up into multiple nodes and leaves to reach a result so it is generally not affected by outliers. Random Tree classifier is another such algorithm that is going to give proper results as it consists of multiple decision trees so it gives better result.

Data Sources and their formats

The dataset contained 209593 rows and 37 columns.

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0

5 rows × 37 columns

..	maxamnt_loans30	medianamnt_loans30	cnt_loans90	amnt_loans90	maxamnt_loans90	medianamnt_loans90	payback30	payback90	pcircle	pdate
..	6.0	0.0	2.0	12	6	0.0	29.000000	29.000000	UPW	20-07-2016
..	12.0	0.0	1.0	12	12	0.0	0.000000	0.000000	UPW	10-08-2016
..	6.0	0.0	1.0	6	6	0.0	0.000000	0.000000	UPW	19-08-2016
..	6.0	0.0	2.0	12	6	0.0	0.000000	0.000000	UPW	06-06-2016
..	6.0	0.0	7.0	42	6	0.0	2.333333	2.333333	UPW	22-06-2016

The columns present are as follows:

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

msisdn------------ > mobile number of user

aon ------------- > age on cellular network in days

daily_decr30 ------------- > Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)

daily_decr90------------- > Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)

rental30------------ > Average main account balance over last 30 days

rental90------------ > Average main account balance over last 90 days

last_rech_date_ma------------ > Number of days till last recharge of main account

last_rech_date_da------------ > Number of days till last recharge of data account

last_rech_amt_ma------------ > Amount of last recharge of main account (in Indonesian Rupiah)

cnt_ma_rech30 > Number of times main account got recharged in last 30 days

fr_ma_rech30------------- > Frequency of main account recharged in last 30 days

sumamnt_ma_rech30------------- > Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)
medianamnt_ma_rech30 ------------ > Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
medianmarechprebal30 Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
cnt_ma_rech90 -------------> Number of times main account got recharged in last 90 days
fr_ma_rech90------------ > Frequency of main account recharged in last 90 days
sumamnt_ma_rech90------------- > Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)
medianamnt_ma_rech90 ------------ > Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)
medianmarechprebal90 Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)
cnt_da_rech30------------ > Number of times data account got recharged in last 30 days
fr_da_rech30------------ > Frequency of data account recharged in last 30 days
cnt_da_rech90------------ > Number of times data account got recharged in last 90 days
fr_da_rech90------------ > Frequency of data account recharged in last 90 days
cnt_loans30------------- > Number of loans taken by user in last 30 days
amnt_loans30------------ > Total amount of loans taken by user in last 30 days
maxamnt_loans30------------- > maximum amount of loan taken by the user in last 30 days
medianamnt_loans30 ------------ > Median of amounts of loan taken by the user in last 30 days
cnt_loans90----------- > Number of loans taken by user in last 90 days
amnt_loans90 ----------- > Total amount of loans taken by user in last 90 days
maxamnt_loans90------------- > maximum amount of loan taken by the user in last 90 days
medianamnt_loans90 ------------ > Median of amounts of loan taken by the user in last 90 days
payback30------------ > Average payback time in days over last 30 days
pcircle ------------- > telecom circle
pdate ------------ > date

Data Preprocessing Done

For cleaning the data firstly the data was checked for inconsistencies by checking the standard deviation. On finding higher std. values it could be guessed that there were outliers present in the data. This suspicion was confirmed by plotting the values in a boxplot. After which the outliers were removed by importing z-score from scipy.stats library and as a result over 18726 rows were removed.

Data Inputs- Logic- Output Relationships

The dataset contained independent values which affected the dependent feature . The data also contained string datatype values that had to be dropped for proper analysis to be done. Features like (independent) medians of loan taken , average payback time , number of loans taken, frequency of data accounts recharged, total amount of recharge in the main account play a key factor. Whereas , mobile number of the user , date ,circle had to be dropped as it was not affecting the process of data modelling.

Hardware and Software Requirements and Tools Used

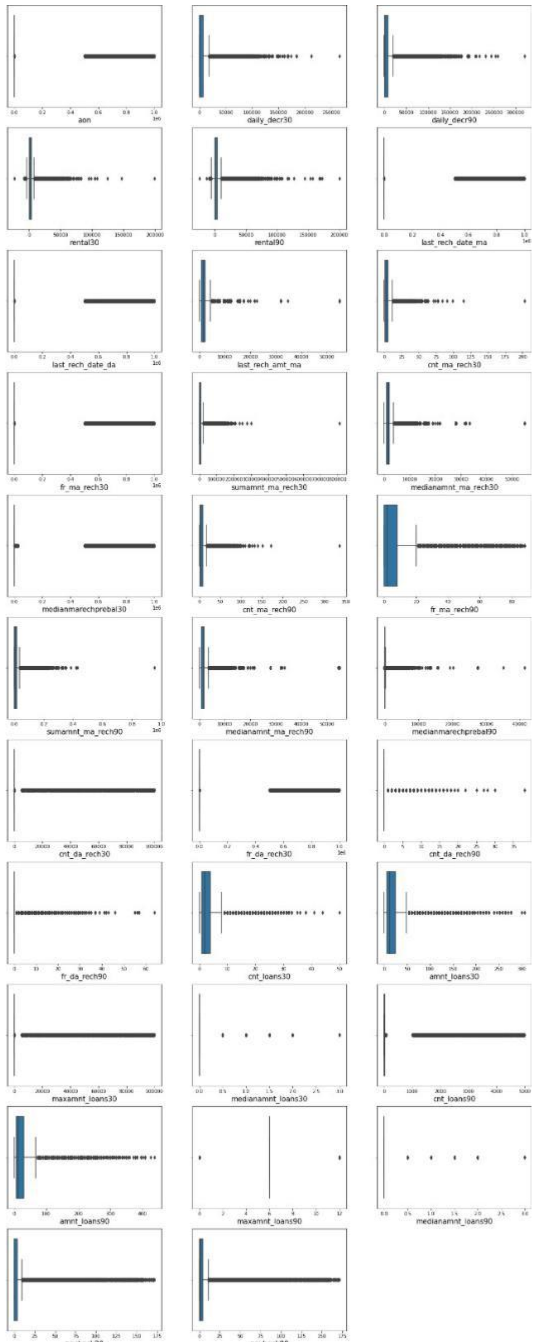
The project was done on a 8 GB ram and 320 GB SSD + 500 GB HDD .Which proved to be a bit cumbersome when it came to tackling the dataset at hand as a result a limitation was reached with respect to the processes that could be done. Libraries used for the project.

1. Pandas DataFrame
2. Numpy array
3. Matplotlib.pyplot(Scatterplot)
4. Seaborn (countplot, heatmap,barplot,boxplot,distplot)
5. Scipy.stats.zscore
6. Imblearn.over_sampling.SMOTE
7. sklearn.model_selection .train_test_split
8. sklearn.preprocessing .MinMaxScaler
9. sklearn.tree .DecisionTreeClassifier
10. sklearn.neighbors .KNeighborsClassifier
11. sklearn.metrics .accuracy_score
12. sklearn.metrics.confusion_matrix
13. sklearn.metrics. classification_report
14. sklearn.svm.LinearSVC
15. sklearn.linear_model.LogisticRegression
16. sklearn.ensemble.RandomForestClassifier
17. from sklearn.metrics.mean_squared_error
18. sklearn.metrics . roc_curve
19. from sklearn.metrics .roc_auc_score
20. from sklearn.metrics .auc

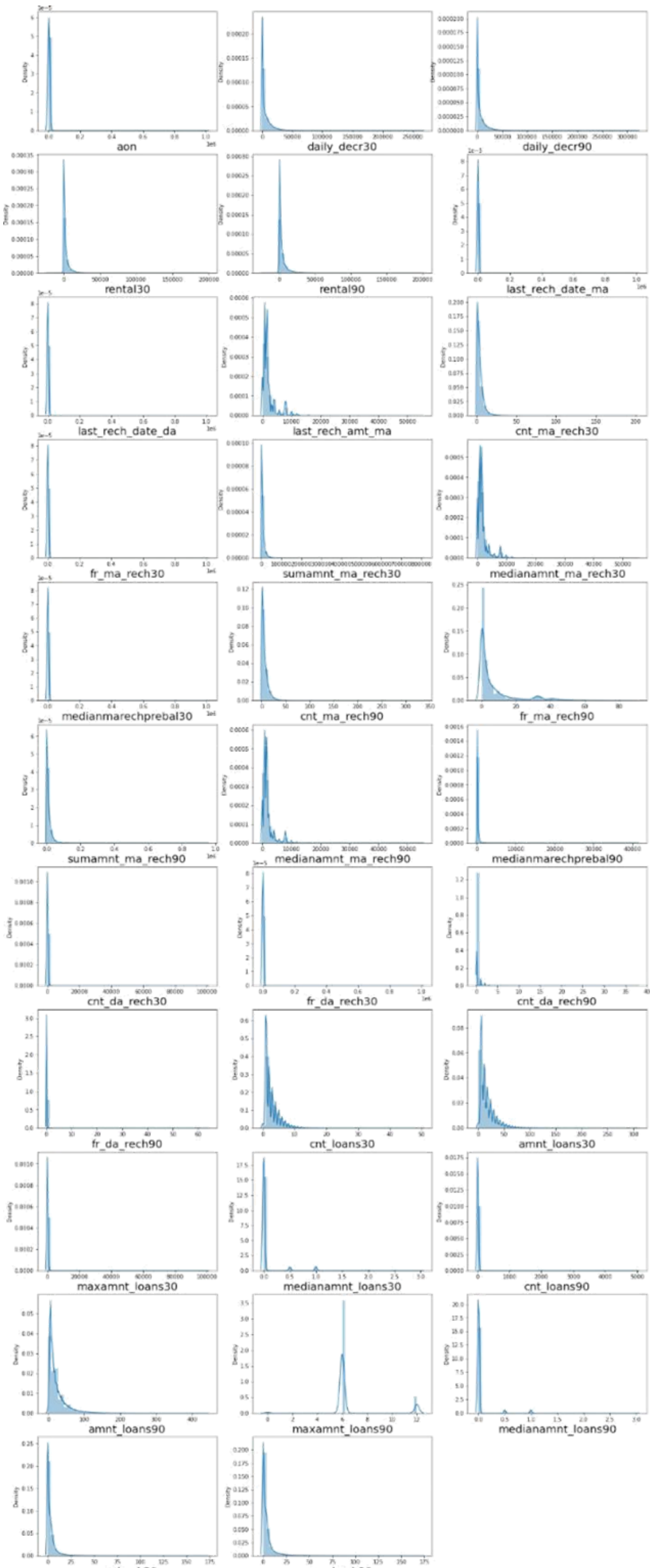
Model/s Development and Evaluation

The dataset had a lot of outliers and missing values which was found by doing exploratory data analysis. It was done by using outlier detection by trying find

plotting each data(continuous) through boxplot.



For the dataset we have checked for skewness by using a KDE plot to understand the nature of the skewness and it was found to be positively skewed.

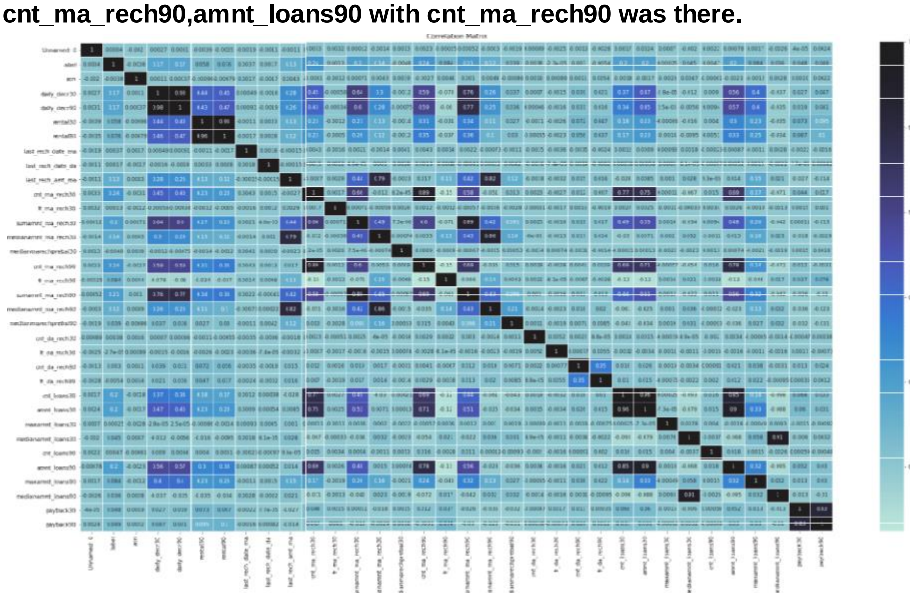

aon
daily_decr30
daily_decr90
rental30
rental90
last_rech_date_ma
last_rech_date_da
last_rech_amt_ma
cnt_ma_rech30
fr_ma_rech30
sumamnt_ma_rech30
medianamnt_ma_rech30
medianmarechprebal30
cnt_ma_rech90
fr_ma_rech90
sumamnt_ma_rech90
medianamnt_ma_rech90
medianmarechprebal90
cnt_da_rech30
fr_da_rech30
cnt_da_rech90
fr_da_rech90
cnt_loans30
amnt_loans30
maxamnt_loans30
medianamnt_loans30
cnt_loans90
amnt_loans90
maxamnt_loans90
medianamnt_loans90
payback30
payback90

- **Algorithms useed**

1. **from sklearn.model_selection import train_test_split,GridSearchCV**
2. **from sklearn.tree impport DecisionTreeRegressor**
3. **from sklearn.ensemble import RandomForestClassifier**
4. **from sklearn.linear_model import LogisticRegression**
5. **from sklearn.tree impport DecisionTreeClassifier**
6. **from sklearn.ensemble import GradientBoostingClassifier**
7. **from sklearn.model_selection import cross_val_score**

Identification of poossible problem-solving approaches (methods)

Some of the features werre correlated with each other so, those are droppped to create more accurate models. correlation between daily_decr90 and daily_decr30, rental90 and rental30,sumamnt_ma_rech90 with daily_decr90 and dailyy_decr30, sumamnt_ma_rech30 correlation between medianamnt_ma_rech90 with last_rech_amt_ma, correelation between cnt_ma_rech30 with cnt_ma_rech90,amnt_loans90 with cnt_ma_rech90 was there.



As , most of the features were having outliers and skewness, we have reemove the skewness by using PowerrTransformer(yeo-johnson) and for outliers Z-Sccore has used.

```
# skewed features are taken into a list
features=['aon','daily_decr30','rental30','last_rech_date_ma','last_rech_amt_ma',
          'fr_ma_rech30','sumamnt_ma_rech30','medianamnt_ma_rech30','medianmarechprebal30','cnt_ma_rech90',
          'fr_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','maxamnt_loans90',
         'payback30','payback90']
```

```
scaler=PowerTransformer(method='yeo-johnson',standardize=True)   # using yeo-johnson method
```

```
df[features]=scaler.fit_transform(df[features].values)   # scaling the features
```

```
z_score = zscore(df[['aon','daily_decr30','rental30','last_rech_date_ma','last_rech_date_da',"last_rech_amt_ma',
                   'sumamnt_ma_rech30','medianamnt_ma_rech30','medianmarechprebal30','cnt_ma_rech90','fr_da_rech90','medianm
                     'amnt_loans30"]])
abs_z_score=np.abs(z_score)#converting data into standard normal distribution

filtering_entry=(abs_z_score<3).all(axis=1)

df=df[filtering_entry]
df.describe()
```

Then some of the featurees needed to be dropped like ‘msisdn’, ‘Unnameed:0’,’ pdate’, 'Year','pcircle'.

Testing of Identified Approaches (Algorithms)

1. **from sklearn.tree import DeccisionTreeRegressor**
2. **from sklearn.ensemble import RandomForestClassifier**
3. **from sklearn.linear_model immport LogisticRegression**
4. **from sklearn.tree import DeccisionTreeClassifier**
5. **from sklearn.ensemble import GradientBoostingClassifier**

Run and Evaluate selected models

<u>Random forest classiffier</u> - A random forest is a meta estimator that fits a number of decision tree classiifiers on various sub-samples of the dataset andd uses averaging to improve the predictive accuracy and control over-fittingg. It gave a score of 95% accuracy which was the highest among all the scores.

```
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
rf_pred = rf.predict(x_train)
rf_clf_report = pd.DataFrame(classification_report(y_train,rf_pred,output_dict=True))
print("\n======================Train Result==========================")
print(f"Accuracy score:{accuracy_score(y_train,rf_pred)*100:.2f}%")
print("_______________________________________________________")
print(f"CLASSIFICATION REPORT:\n{rf_clf_report}")
print("_______________________________________________________")
print(f" Confusion Matrix:\n{confusion_matrix(y_train,rf_pred)}\n")
```

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

_______________________________________________________
CLASSIFICATION REPORT:
                     0              1  accuracy      macro avg  \
precision     0.999984       0.999968  0.999976       0.999976
recall        0.999968       0.999984  0.999976       0.999976
f1-score      0.999976       0.999976  0.999976       0.999976
support  124925.000000  125084.000000  0.999976  250009.000000

           weighted avg
precision      0.999976
recall         0.999976
f1-score       0.999976
support   250009.000000
```

```python
rf_pred=rf.predict(x_test)
rf_clf_report = pd.DataFrame(classification_report(y_test,rf_pred,output_dict=True))
print("\n=======================Test Result of RF_clf============================")
print(f"Accuracy score:{accuracy_score(y_test,rf_pred)*100:.2f}%")
print("_______________________________________________________________")
print(f"CLASSIFICATION REPORT:\n{rf_clf_report}")
print("________________________________________________________________")
print(f" Confusion Matrix:\n{confusion_matrix(y_test,rf_pred)}\n")
```

```
=======================Test Result of RF_clf============================
Accuracy score:95.02%
_______________________________________________________________
CLASSIFICATION REPORT:
                      0             1  accuracy     macro avg  weighted avg
precision      0.951117      0.949263   0.95019      0.950190      0.950192
recall         0.949363      0.951021   0.95019      0.950192      0.950190
f1-score       0.950239      0.950141   0.95019      0.950190      0.950190
support    41748.000000  41589.000000   0.95019  83337.000000  83337.000000
________________________________________________________________
 Confusion Matrix:
[[39634  2114]
 [ 2037 39552]]
```

__Decision Tree Classifier__- non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation. It gave a score of 90% accuracy .

```python
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
dt_pred = dt.predict(x_train)
dt_clf_report = pd.DataFrame(classification_report(y_train,dt_pred,output_dict=True))
print("\n=======================Train Result===========================")
print(f"Accuracy score:{accuracy_score(y_train,dt_pred)*100:.2f}%")
print("_______________________________________________________________")
print(f"CLASSIFICATION REPORT:\n{dt_clf_report}")
print("________________________________________________________________")
print(f" Confusion Matrix:\n{confusion_matrix(y_train,dt_pred)}\n")

```

```
=======================Train Result===========================
Accuracy score:100.00%
_______________________________________________________________
CLASSIFICATION REPORT:
                       0              1  accuracy      macro avg  \
precision       0.999952       1.000000  0.999976       0.999976
recall          1.000000       0.999952  0.999976       0.999976
f1-score        0.999976       0.999976  0.999976       0.999976
support    124925.000000  125084.000000  0.999976  250009.000000

            weighted avg
precision       0.999976
recall          0.999976
f1-score        0.999976
support    250009.000000
```

```python
dt_pred=dt.predict(x_test)
dt_clf_report = pd.DataFrame(classification_report(y_test,dt_pred,output_dict=True))
print("\n=======================Test Result of dt_clf===========================")
print(f"Accuracy score:{accuracy_score(y_test,dt_pred)*100:.2f}%")
print("_______________________________________________________________")
print(f"CLASSIFICATION REPORT:\n{dt_clf_report}")
print("________________________________________________________________")
print(f" Confusion Matrix:\n{confusion_matrix(y_test,dt_pred)}\n")
```

```
=======================Test Result of dt_clf===========================
Accuracy score:90.70%
_______________________________________________________________
CLASSIFICATION REPORT:
                      0             1  accuracy     macro avg  weighted avg
precision      0.900429      0.913831  0.907004      0.907130      0.907117
recall         0.915613      0.898363  0.907004      0.906988      0.907004
f1-score       0.907957      0.906031  0.907004      0.906994      0.906996
support    41748.000000  41589.000000  0.907004  83337.000000  83337.000000
________________________________________________________________
 Confusion Matrix:
[[38225  3523]
 [ 4227 37362]]
```

<u>Logistic Regression</u>- is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. It gave a score of 77% accuracy .

```python
lr = LogisticRegression()
lr.fit(x_train,y_train)
lr_pred = lr.predict(x_train)
lr_clf_report = pd.DataFrame(classification_report(y_train,lr_pred,output_dict=True))
print("\n=======================Train Result===========================")
print(f"Accuracy score:{accuracy_score(y_train,lr_pred)*100:.2f}%")
print("_______________________________________________________________")
print(f"CLASSIFICATION REPORT:\n{lr_clf_report}")
print("________________________________________________________________")
print(f" Confusion Matrix:\n{confusion_matrix(y_train,lr_pred)}\n")
```

```
=======================Train Result===========================
Accuracy score:77.39%
_______________________________________________________________
CLASSIFICATION REPORT:
                       0              1  accuracy      macro avg  \
precision       0.772081       0.775796  0.773928       0.773939
recall          0.776914       0.770946  0.773928       0.773930
f1-score        0.774490       0.773363  0.773928       0.773927
support    124925.000000  125084.000000  0.773928  250009.000000

            weighted avg
precision       0.773940
recall          0.773928
f1-score        0.773926
support    250009.000000
```

```python
lr_pred=lr.predict(x_test)
lr_clf_report = pd.DataFrame(classification_report(y_test,lr_pred,output_dict=True))
print("\n=======================Test Result of LR ===========================")
print(f"Accuracy score:{accuracy_score(y_test,lr_pred)*100:.2f}%")
print("_______________________________________________________________")
print(f"CLASSIFICATION REPORT:\n{lr_clf_report}")
print("________________________________________________________________")
print(f" Confusion Matrix:\n{confusion_matrix(y_test,lr_pred)}\n")
```

```
=======================Test Result of LR ===========================
Accuracy score:77.33%
_______________________________________________________________
CLASSIFICATION REPORT:
                      0             1  accuracy     macro avg  weighted avg
precision      0.771878      0.774832  0.773342      0.773355      0.773352
recall         0.777259      0.769410  0.773342      0.773334      0.773342
f1-score       0.774559      0.772111  0.773342      0.773335      0.773338
support    41748.000000  41589.000000  0.773342  83337.000000  83337.000000
________________________________________________________________
 Confusion Matrix:
[[32449  9299]
 [ 9590 31999]]
```

<u>Gradient Boosting Classifier</u>- gradient boosted trees use decision tress as estimators. Evaluate its gradient and approximates it with a simple tree(stage wisely , that minimizes the overall error). First it calculates the average of the target, for the first iteration it (average of actual target) is the predicted target, Then it calculates the pseudo-residual by subtracting the first predicted target(average of actual target) by actual target . It tries to reduce the error function as it creates a tree to predict the pseudo-residuals instead of a tree to predict for actual column values. Then When data is not following any pattern gradient

boosting trees are very helpful as it tries to optimize the error function. It gave a score of 90% accuracy .

```python
gbdt_clf = GradientBoostingClassifier()
gbdt_clf.fit(x_train,y_train)
pred=gbdt_clf.predict(x_train)
gbdt_clf_report = pd.DataFrame(classification_report(y_train,pred,output_dict=True))
print("\n=======================Train Result===========================")
print(f"Accuracy score:{accuracy_score(y_train,pred)*100:.2f}%")
print("_______________________________________________________________")
print(f"CLASSIFICATION REPORT:\n{gbdt_clf_report}")
print("________________________________________________________________")
print(f" Confusion Matrix:\n{confusion_matrix(y_train,pred)}\n")
```

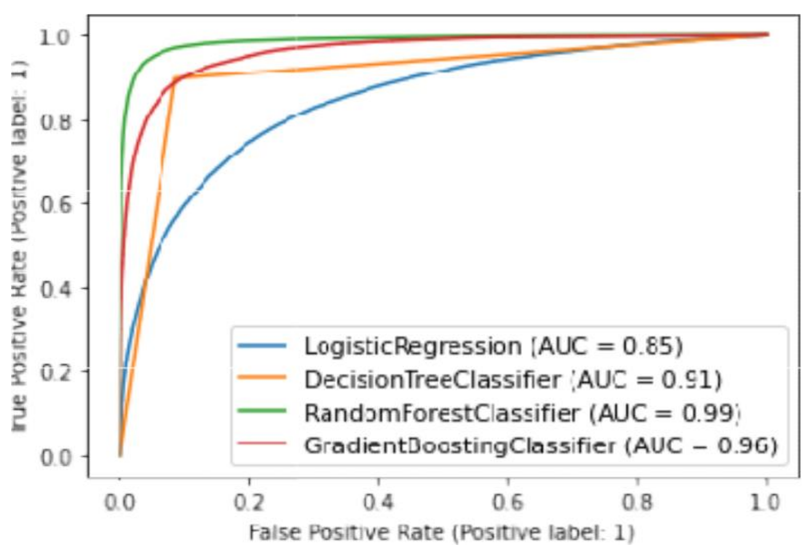
```
=======================Train Result===========================
Accuracy score:90.14%
_______________________________________________________________
CLASSIFICATION REPORT:
                       0              1  accuracy      macro avg  \
precision       0.893298       0.909819  0.901396       0.901559
recall          0.911547       0.891257  0.901396       0.901402
f1-score        0.902330       0.900443  0.901396       0.901387
support    124925.000000  125084.000000  0.901396  250009.000000

            weighted avg
precision       0.901564
recall          0.901396
f1-score        0.901386
support    250009.000000
```

```
pred=gbdt_clf.predict(x_test)
clf_report = pd.DataFrame(classification_report(y_test,pred,output_dict=True))
print("\n=======================Test Result============================")
print(f"Accuracy score:{accuracy_score(y_test,pred)*100:.2f}%")
print("_______________________________________________________________")
print(f"CLASSIFICATION REPORT:\n{clf_report}")
print("________________________________________________________________")
print(f" Confusion Matrix:\n{confusion_matrix(y_test,pred)}\n")
```

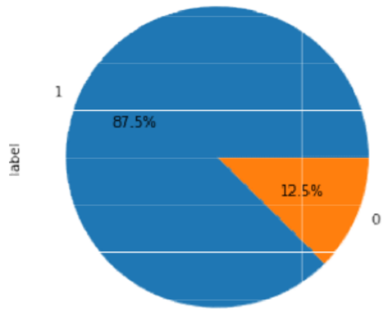
```
=======================Test Result===========================
Accuracy score:90.13%
_______________________________________________________________
CLASSIFICATION REPORT:
                      0             1  accuracy     macro avg  weighted avg
precision      0.893089      0.909962  0.901328      0.901526      0.901510
recall         0.912235      0.890380  0.901328      0.901307      0.901328
f1-score       0.902561      0.900064  0.901328      0.901313      0.901315
support    41748.000000  41589.000000  0.901328  83337.000000  83337.000000
________________________________________________________________
 Confusion Matrix:
[[38084  3664]
 [ 4559 37030]]
```

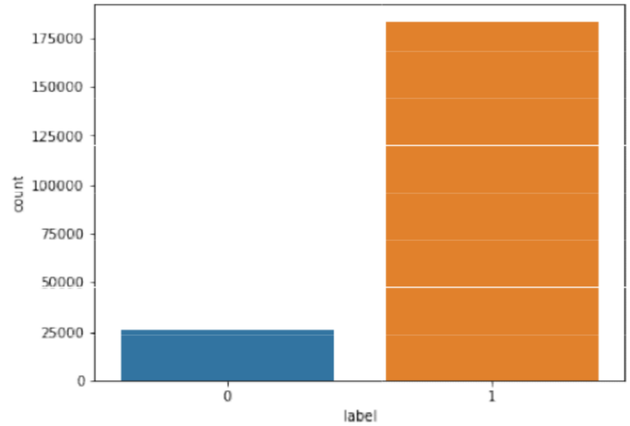
Key Metrics for success in solving problem under consideration

<u>AUC_ROC</u> - is the measurre of the ability of a classifier to distinguish betwween classes and is used as a summaryy of the roc curve. The higher the auc the betterr the performance of the model at distinguishing between the positive and neegative classes.

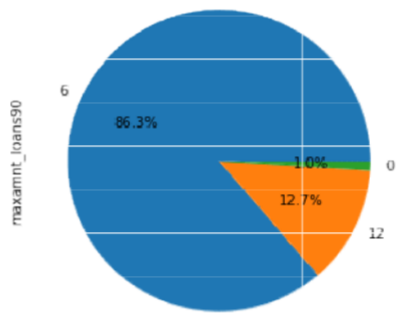


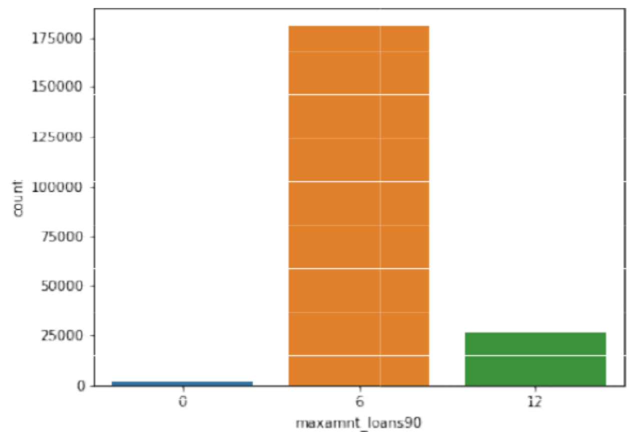
6. Visualizations



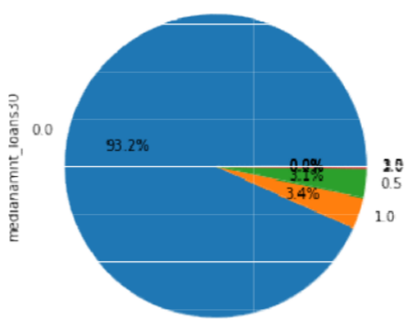


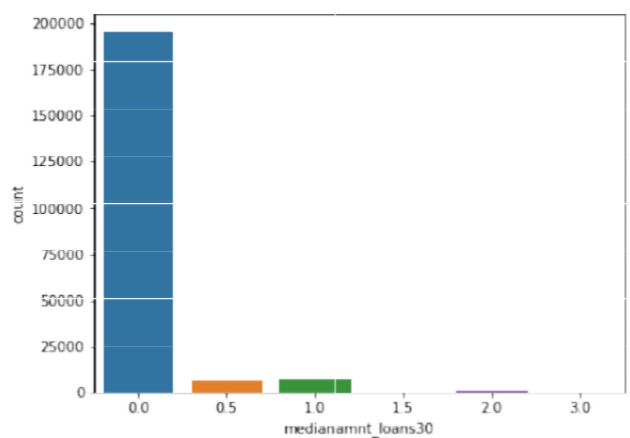
As we can see thee target data was having two categories 0 i.e. deefaulter and 1 non-defaulter. Annd the target was imbalanced.



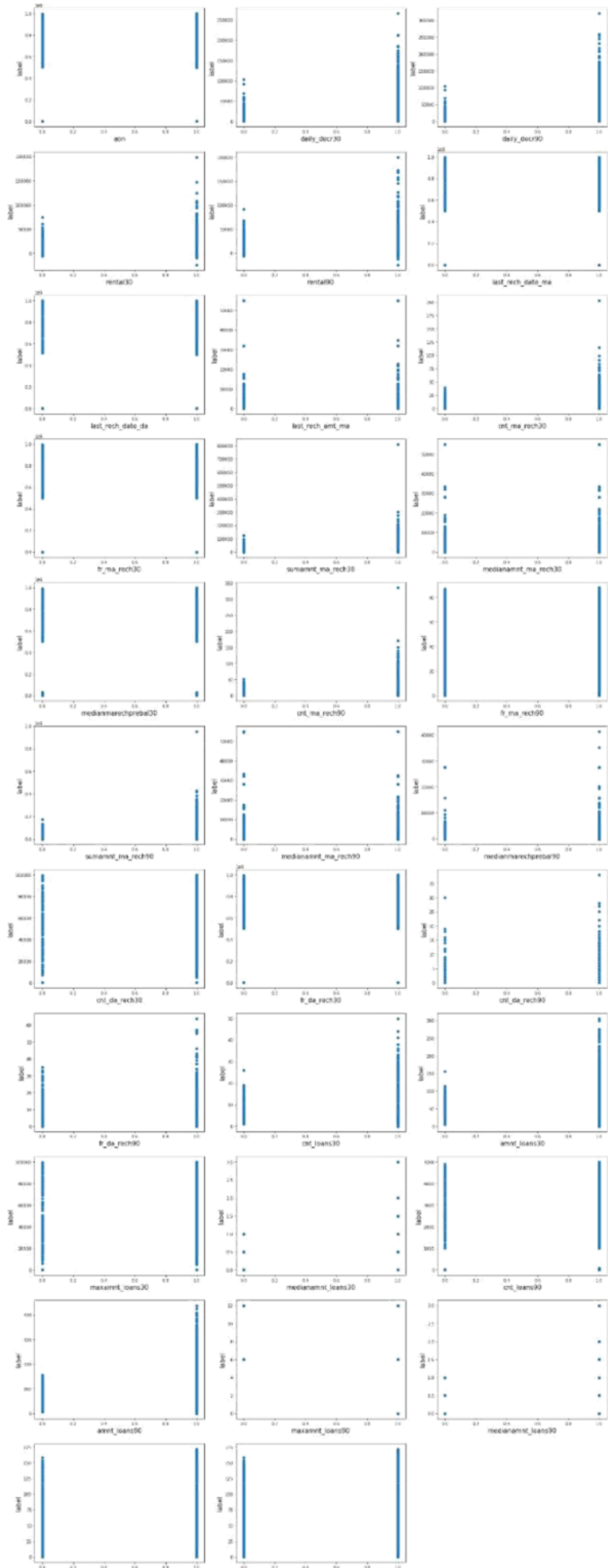


Maximum amounnt of loan taken by the user in last 90 days had 3 categories. Where highest count of people had taken the loan for 6 months.





Median of amounnts of loan taken by the user in last 30 days had 6 categories, where a highest count of people had taken loan 0.0



These scatterplots are showing the relationship between the target and the continuous features.

7. Interpretation of the Results

The dataset contained a lot of information for the same reason it also contained a lot of impurities in the form of outliers, imbalance-ness . It also contained lot information like the phone numbers of the users , the telecom circle , the pdates that though relevant to the company had to be dropped from the working dataset to obtain an optimal dataset based on which the predictions could be made. Treating the imbalanced dataset was priority. Treating the outliers via z-score resulted in removal of the anomalies but over 48000 rows were lost that's over 23% of the data as a result the outliers were not removed. Scaling was done using min-max scaler to bring the numerical data to a similar kind of range. Multiple classification algorithms were checked out of which only the random forest classifier gave the best score. With the best predicted values. Which proves that the random forest classifier trained model can best predict the outcome for this dataset with an accuracy of 95%.

CONCLUSION

8. Key Findings and Conclusions of the Study

It was a large dataset having - 209593 rows x 37 columns. The dataset is based on real world data hence it was having a lot of outliers. Removing the outliers would have resulted in removal of 20%+ of the data that would have resulted in the formation of a biased model.

Out of 4 evaluation metrics the random-forest classifier gave the best results. The reason for it being as it creates multiple decision trees as part of its ensemble techniques it can really work with a dataset that is having a huge number of outliers.

9. Learning Outcomes of the Study in respect of Data Science

While working with a diverse dataset such as this the main challenge that I faced was the system I was working with. The system was not up to speed while crunching the numbers and providing results. There were a few instances where the machine hanged and had to be restarted all together. So I was really challenging to work with

such a dataset. So the data relations had to be studied by plotting the distplot/ boxplot / heatmap. It was fun to observe that the values for each classifier was different which proves that each classifier is different from one another in case of challenging dataset such as this. It showed that there is a relationship among the behaviour of person and their ability/inability to pay. It also showed that how powerful the scikit learn library is when it comes to data-prediction and machine learning. It also came to my attention that how the sigmoid curve actually influences the categorical dataset.

10. Limitations of this work and Scope for Future Work

As the dataset was containing outliers during the testing and training phase of the data it might vary during actual ground work. The Random forest though ended up with a high score in this particular case it cannot be considered as the ultimate model for prediction as the ensemble techniques for the random forest allows it work with outliers. This project has many real world applications especially in the financial fields. The banking industry is always on the lookout for potential customers to give loan to but they would not want to end up on a trade that would result in losing out money. The debt situation in an economy always helps to move the economy forward but if there are a lot defaulters then it might prove to be a financial disaster . So this projects not only provides insight into the pattern of human behaviour but also provides a way out of a financial crisis. It will be interesting to observe the relationship between each user individually can be singled out with the help of their phone numbers. The dataset though gives insight into the inner working of the financial institutes but at the same time it also poses a question as to what might happen to the group of people who are considered as defaulters. Will they be denied loans altogether based on their habits or will the companies factor in more information while getting the predictions?