MICRO CREDIT DEFAULTER PROJECT

By

DEEPTHI PRAKASHAN

ACKNOWLEDGMENT

I would like to sincerely thank Flip Robo Technologies for the opportunity to work in the project and the timely guidance provided.

I express my gratitude towards Data Trained for the training they have provided overtime and for the guidance and support.

I express my heartfelt support to my parents and friends who were constantly by my side providing me with motivation and inspiration to work through the completion of this project.

INTRODUCTION

Business Problem

A telecom company is 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).

However, it is risky to provide every individual with a loan. When an individual applies for micro credit loans, we need to assess if they would repay the loan. It is on this basis that we determine whether to approve or decline the application for loan.

Conceptual Background

Microfinance refers to giving small loans to individuals with lower socioeconomic background who typically lack collateral, employment or have a low-income job. Hence these schemes help them participate in the economy. 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.

Motivation of the problem undertaken

As we have discussed, microfinance arguably has a positive impact in enabling economic empowerment. But the provision of loans as mobile balances as a microcredit, which is a subset of microfinance, is a challenging task since it requires evaluating the creditworthiness of consumers. So, the institutional sustainability and its ability to provide with financial services depends on the reducing loan risk and providing loans to individuals who are considered “creditworthy”.

Hence, default prediction is an absolute necessity. In order to improve the selection of customers for the credit, we need to build a machine learning model which will make predictions based on the credit history and other information of the customers from the database provided to us.

Analytical Problem Framing

The objective of this project is to predict if the micro-credit applicant will be a defaulter or not so we are dealing with a classification problem. We have been given with 37 features that would help evaluate the creditworthiness of a customer to avail them with loan. Various statistical methods were applied to see the relation they share in terms of being able to pay the loan. After the analysis of the data, we used Machine learning algorithms to predict the micro-credit defaulters

- **Data Sources and their formats**

    - The data has been received from FlipRobo Technologies
    - The data is in csv file format
    - There as 209593 rows and 37 columns
    - Label is the target column where 1 denotes the person is a defaulter and 0 if they are not.
    - The dataset is imbalanced. Label '1' has approximately 87.5% records, while, label '0' has approximately 12.5% records.

- **Data Pre-processing**

    pdate was separated into month date and years columns

    Unnecessary columns that were dropped:

    Unnamed:0,

msisdn (mobile number),
pcircle,
year

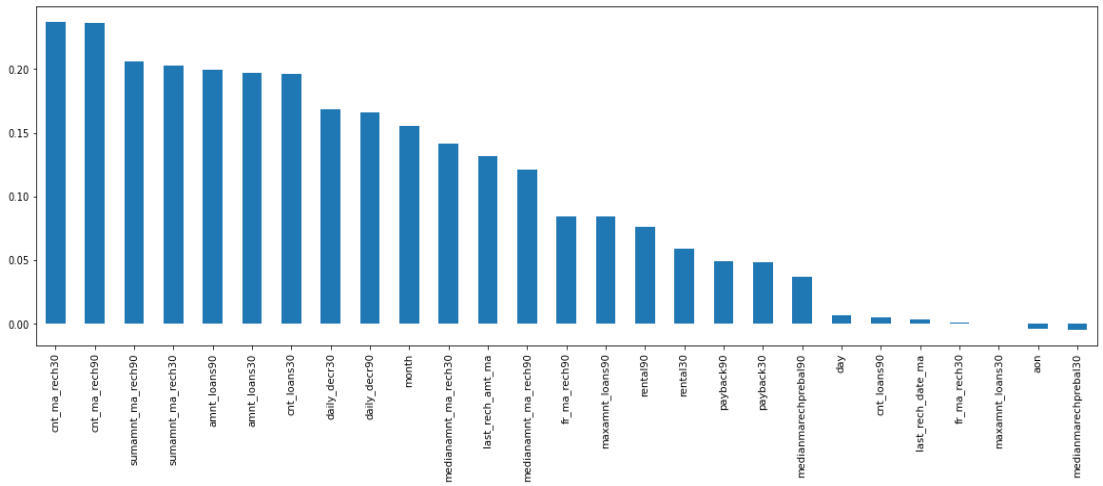
Columns were seen to contain negative values. The following column values were converted to positive values:
aon, daily_dec30, daily_decr90, rental30, rental90, last_rech_date_ma, last_rech_date_da, medianmarechprebal30, medianmarechprebal30.

medianamnt_loans90, medianamnt_loans30, fr_da_rech90, cnt_da_rech90, fr_da_rech30, cnt_da_rech30, last_rech_date_da, have above 90% data as zero. Due to the lack of variation in the data it will not contribute much to the prediction and was dropped.

Using boxplot, a lot of outliers were detected. Using IQR, the upper and lower limits were detected and proceeded to use impute the values using mean/median that went beyond the upper and lower limit.

- **Data Inputs- Logic- Output Relationships**
  In order to decide the data inputs that are necessary to build our machine learning model, we are required to understand the pattern/relationship that exists between the label and features. For the same we have conducted an in-depth exploratory data analysis through techniques like correlation analysis and data visualization techniques that included

Correlation Analysis was visualized through a heatmap and a bar plot. It was observed that most of the features were positively correlated, but we can see that all the features have value of correlation less than 0.3.



2.3 Hardware and Software Requirements and Tools Used

Hardware	
Processor	Dual Core or above
RAM	4GB or more
Cache	1MB or more
Hard Disk/SSD	180GB or more
Software	
OS	Window/Mac/Linux
Ide	Jupiter Notebook
Dataset	.csv file
Libraries	Pandas, numpy,matplotlib,seaborn, scikit.learn
Server	Web server with HTTP process

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

With the pre-processed data, we proceeded with training the models. To do the same, the data was split into train and test dataset. Here, 80% of the data was used for training and 20% for testing. Training the data allowed the models to learn the data and the underlying trend. Best random state was also checked, for which best r2 score was 0.95 at random state 39 (Using RandomForestClassifier), which was later used while training the models. Model's efficiency was then checked by predicting the house sales price from the test dataset. Evaluation metrics and cross-validation then helped us decide how good our prediction of default prediction was and if the predicted values were not as a result of overfitting or underfitting.

The models that were used in prediction were

- RandomForestClassifier
- DecisionTreeClassifier
- KNeighborsClassifier
- AdaBoostClassifier
- GradientBoostingClassifier.

Run and evaluate selected models

RandomForestClassification

```
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
predrf= rf.predict(x_test)
print('Accuracy',accuracy_score(y_test,predrf)*100)
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
Accuracy 95.4747029620126
[[34679  1392]
 [ 1853 33784]]
              precision    recall  f1-score   support

           0       0.95      0.96      0.96     36071
           1       0.96      0.95      0.95     35637

    accuracy                           0.95     71708
   macro avg       0.95      0.95      0.95     71708
weighted avg       0.95      0.95      0.95     71708
```

```
cv = cross_val_score(rf,x,y,cv=7)

print('Cross Validation Score: ', cv.mean())
print('Difference in cross val score', accuracy_score(y_test,predrf)- abs(cv.mean()))
print('**'*50)
```

```
Cross Validation Score:  0.9511768133350703
Difference in cross val score 0.003570216285055716
****************************************************************************************************
```

RandomForestClassifier is an ensemble techinque which is categorized under supervised machine learning. It evaluates by taking random subsets of data from the data sample and creating decision trees from each. The model's outcome is the value that was predicted by majority of the decision trees.

Here our model has predicted the test dataset with an accuracy of 95.11%.Further quatitative assessment for the performance of the model was made using other metrics such as precision,recall and ROC-Curve. Our model was able to correctly identify 95 defaulters from the 100 cases predicted as defaulters giving it a 95% precision. Out of all the defaulters, the model was able to catch 96% of the defaulters.

DecisionTreeClassifier

```
df = DecisionTreeClassifier()
df.fit(x_train,y_train)
preddf= df.predict(x_test)
print('Accuracy',accuracy_score(y_test,preddf)*100)
print(confusion_matrix(y_test,preddf))
print(classification_report(y_test,preddf))

print('**'*50)

cv = cross_val_score(df,x,y,cv=7)

print('Cross Validation Score: ', cv.mean())
print('Difference in cross val score', accuracy_score(y_test,preddf)- abs(cv.mean()))
```

```
Accuracy 91.27294025771184
[[33219  2852]
 [ 3406 32231]]
              precision    recall  f1-score   support

           0       0.91      0.92      0.91     36071
           1       0.92      0.90      0.91     35637

    accuracy                           0.91     71708
   macro avg       0.91      0.91      0.91     71708
weighted avg       0.91      0.91      0.91     71708

****************************************************************************************************
Cross Validation Score:  0.9087350309097711
Difference in cross val score 0.003994371667347174
```

DecisionTreeClassifier is a supervised machine learning model where it splits the dataset recursively until we are left with pure leaf nodes.

This model predicts with an accuracy of 91.27%. It correctly identified 91% defaulters from the total cases predicted as defaulters. Recall was found to be 92%, suggesting that it was able to catch 92% of the defaulters from the total number of defaulters. On cross validation, we did get a score close to the accuracy score, suggesting that it is not overfitting/underfitting. Overall, we see that RandomForestClassifier has performed better than DecisionTreeClassifier in all aspects.

KNeighborsClassifier

```
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
predknn= knn.predict(x_test)
print('Accuracy',accuracy_score(y_test,predknn)*100)
print(confusion_matrix(y_test,predknn))
print(classification_report(y_test,predknn))

print('**'*50)

cv = cross_val_score(knn,x,y,cv=7)

print('Cross Validation Score: ', cv.mean())
print('Difference in cross val score', accuracy_score(y_test,predknn)- abs(cv.mean()))
```

```
Accuracy 90.2632900094829
[[35823   248]
 [ 6734 28903]]
              precision    recall  f1-score   support

           0       0.84      0.99      0.91     36071
           1       0.99      0.81      0.89     35637

    accuracy                           0.90     71708
   macro avg       0.92      0.90      0.90     71708
weighted avg       0.92      0.90      0.90     71708

****************************************************************************************************
Cross Validation Score:  0.9041412679398385
Difference in cross val score -0.0015083678450095173
```

KNeighborsClassifier is a non-parametric supervised machine learning model that predicts the target value by averaging the values of data points that are found in its close distance.

KNeighborsClassifier has predicted with an accuracy of 90.26%. Among all the models, it has the lowest precision. So, it is only able to correctly predict 84% cases from total cases predicted as defaulters. However, the model overcompensates by catching 99% of the defaulters. While this model is beneficial for the company as it can catch most of the defaulters, yet from a customer point of view this can cause dissatisfaction among them as it categorizes relatively higher percentage of the non-defaulters as defaulters.

AdaBoost

```
ada = AdaBoostClassifier()
ada.fit(x_train,y_train)
predada= ada.predict(x_test)
print('Accuracy',accuracy_score(y_test,predada)*100)
print(confusion_matrix(y_test,predada))
print(classification_report(y_test,predada))

print('**'*50)

cv = cross_val_score(ada,x,y,cv=7)

print('Cross Validation Score: ', cv.mean())
print('Difference in cross val score', accuracy_score(y_test,predada)- abs(cv.mean()))
```

```
Accuracy 84.93473531544598
[[31183  4888]
 [ 5915 29722]]
              precision    recall  f1-score   support

           0       0.84      0.86      0.85     36071
           1       0.86      0.83      0.85     35637

    accuracy                           0.85     71708
   macro avg       0.85      0.85      0.85     71708
weighted avg       0.85      0.85      0.85     71708

****************************************************************************************************
Cross Validation Score:  0.8443875415054861
Difference in cross val score 0.004959811648973633
```

AdaBoostClassifier is an ensemble model. It utilises ‘stumps’ which is a tree with a node and two leaves. Each stump is made by taking the previous stump’s error into account so some stumps have more say in taking the decision than the other

AdaBoostClassifier predicts with an accuracy of 84.9%. It was able to correctly predict only 84% from the predicted defaulters. And the recall was found to be only 86%. Thus, it is comparatively a poorly performing model.

GradientBoosting

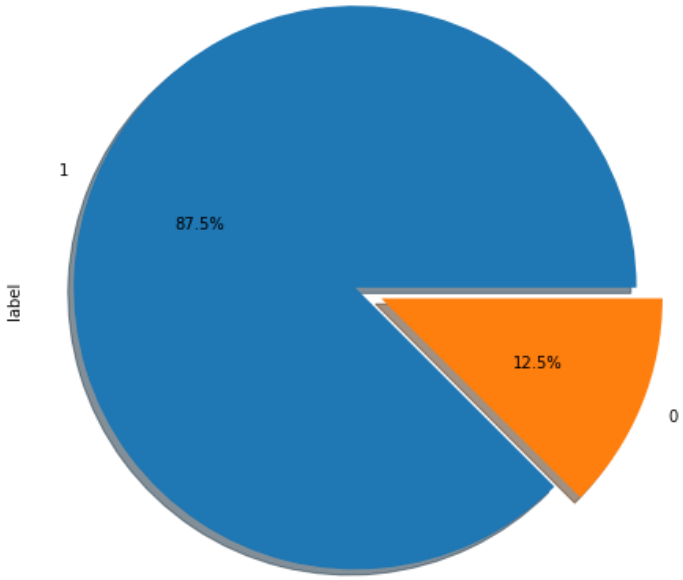
```
gbr = GradientBoostingClassifier()
gbr.fit(x_train,y_train)
predgbr= gbr.predict(x_test)
print('Accuracy',accuracy_score(y_test,predgbr)*100)
print(confusion_matrix(y_test,predgbr))
print(classification_report(y_test,predgbr))

print('**'*50)

cv = cross_val_score(gbr,x,y,cv=7)

print('Cross Validation Score: ', cv.mean())
print('Difference in cross val score', accuracy_score(y_test,predgbr)- abs(cv.mean()))
```

```
Accuracy 89.9216265967535
[[32993  3078]
 [ 4149 31488]]
              precision    recall  f1-score   support

           0       0.89      0.91      0.90     36071
           1       0.91      0.88      0.90     35637

    accuracy                           0.90     71708
   macro avg       0.90      0.90      0.90     71708
weighted avg       0.90      0.90      0.90     71708

****************************************************************************************************
Cross Validation Score:  0.8943181107814552
Difference in cross val score 0.004898155186079833
```

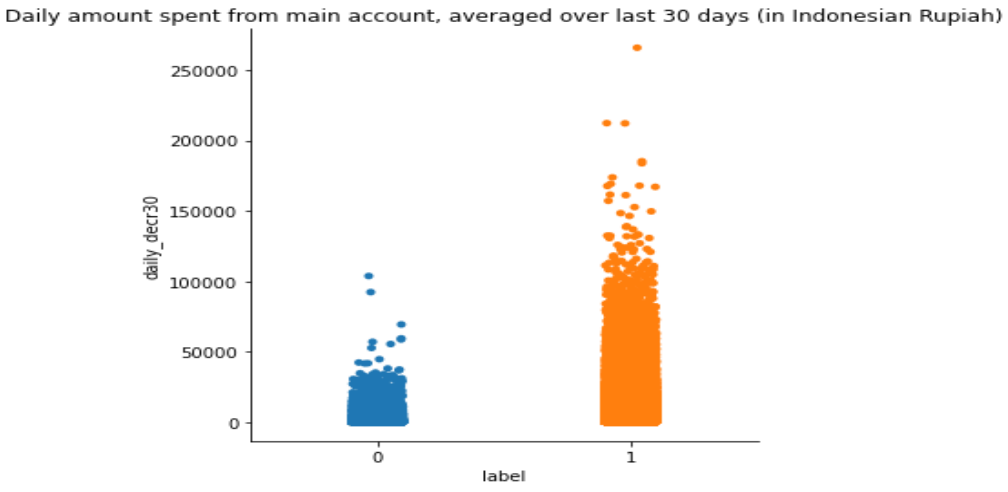
GradientBoostingClassifier is a non-parametric supervised machine learning model.It is an ensemble model that combines multiple models and obtains the prediction. It is also called an additive model as it keeps adding weak learners such that we get a stronger model from the final model.

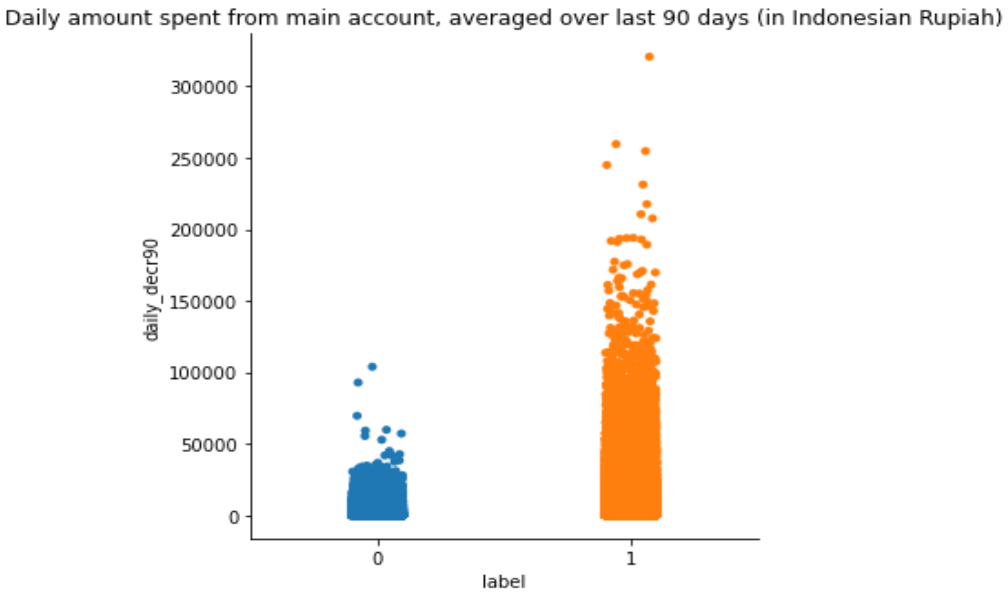
GradientBoostingClassifier has an accuracy score of 89.92%. It is able to correctly predict 89% from the predicted defaulters. Its recall is found to be 91%.

Visualization

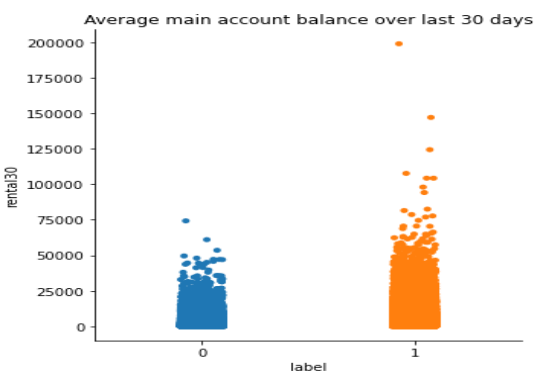


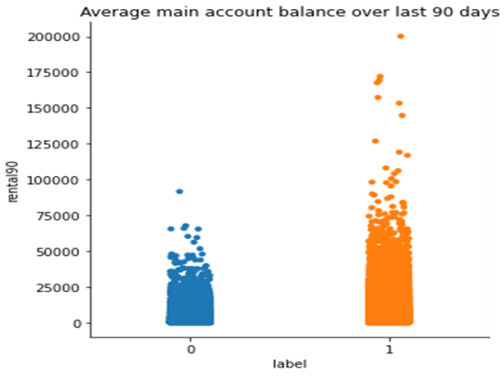
In the dataset we received, we see that 87.5% have paid the loan in time and 12.5% have failed to pay. The data, as we observe is also imbalanced. SMOTE method was applied to balance the dataset



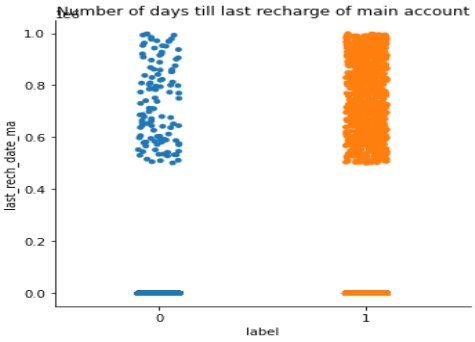


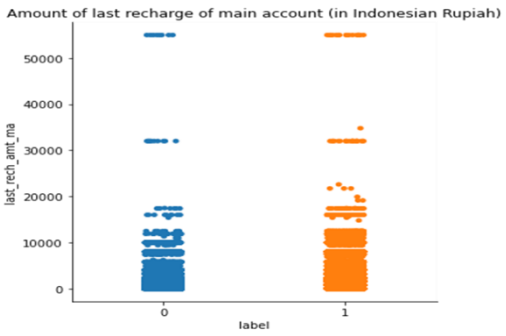
- Daily amount spent averaged over last 30 days for defaulters is less as compared to non-defaulters.
- Daily amount spent averaged over last 90 days for defaulters is less as compared to non-defaulters.



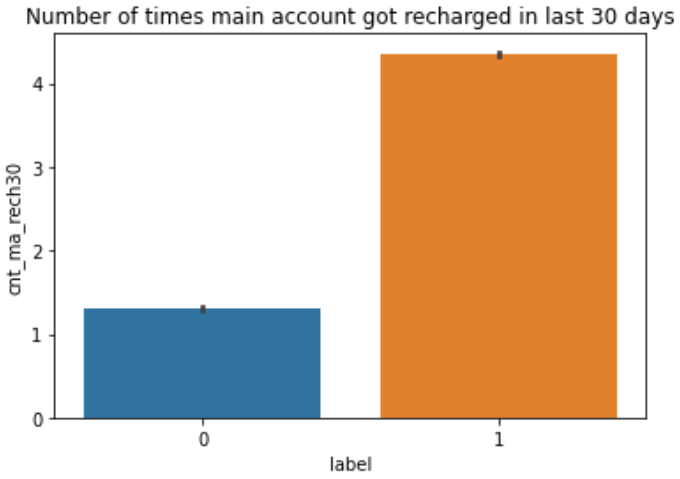


- Average main account balance 30 days for defaulters is less as compared to non-defaulters.
- Average main account balance 90 days for defaulters is less as compared to non-defaulters

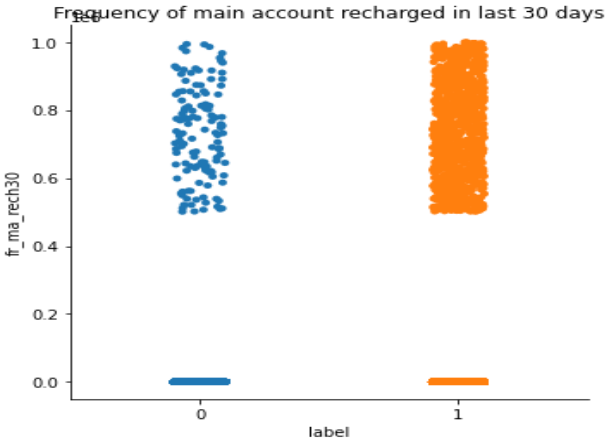




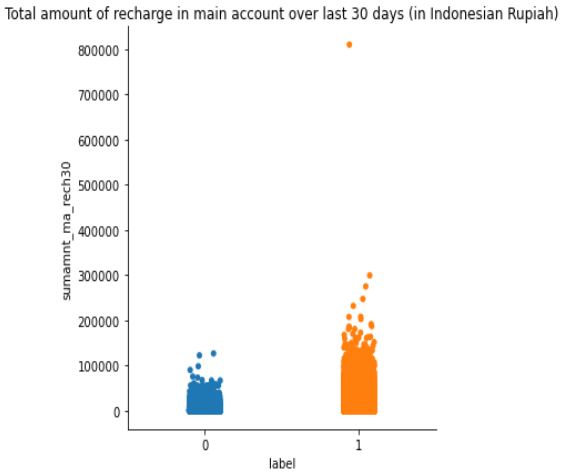
- No difference can be observed for 'last_rech_date_ma'
- No difference can be observed for 'last_amt_date_ma

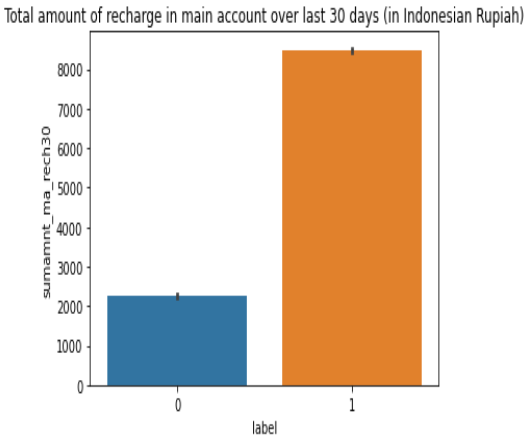


- The number of times defaulters' recharge is lesser as compared to non-defaulters
- Re-payers were seen to recharge their account almost 4x the number of times defaulters were seen to recharge.

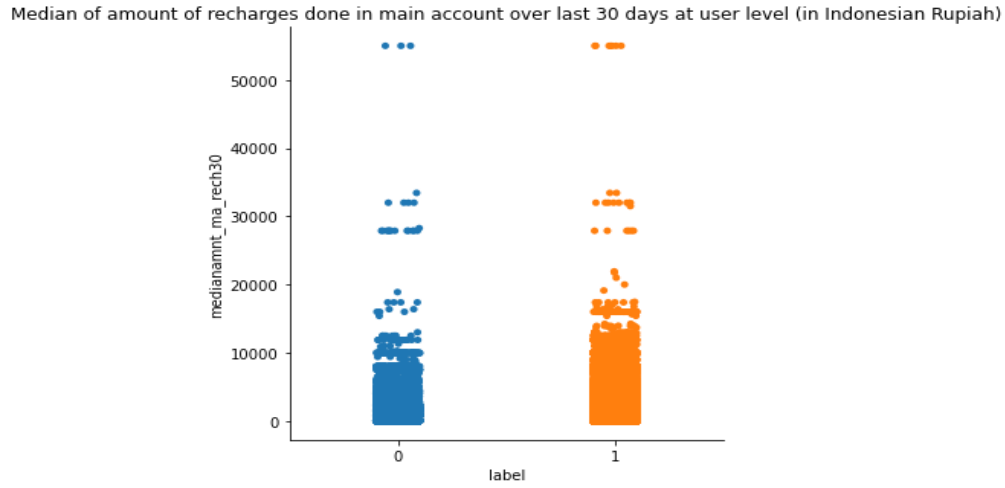


- Frequency of main account recharged in last 30 days show no difference in trend between defaulters and non-defaulters.



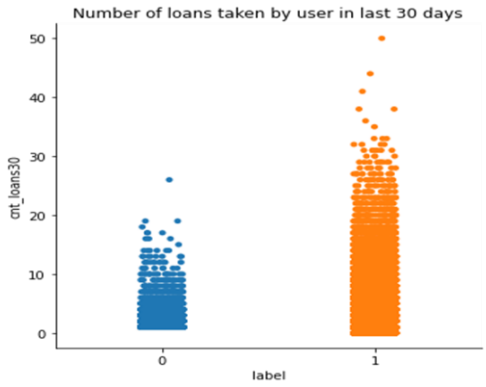


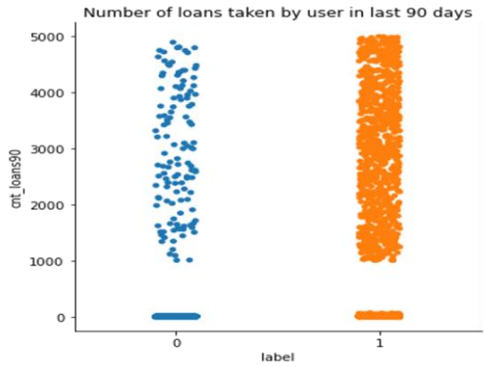
- Total amount of recharge in main account over last 30 days is comparatively less for defaulter's account
- On average, total amount of recharge for defaulters in main account is found to be 2000.
- For non-defaulters, this value is found to be 8000, which is projected slightly higher due to the presence of a seemingly impossible outlier.



.

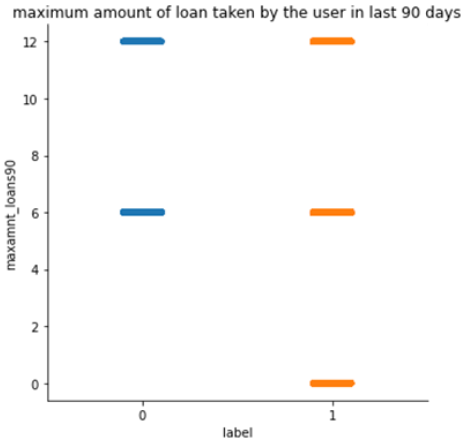
- There is no observable trend between defaulters and non-defaulters in case of Median of amount rechargers' done in main account over last 30 days at user level.



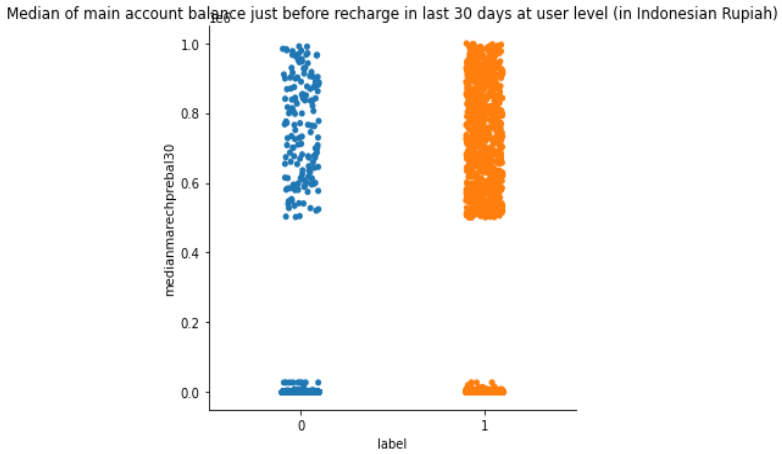


- Number of loans taken by defaulters in last 30 days is lesser as compared to non- defaulters

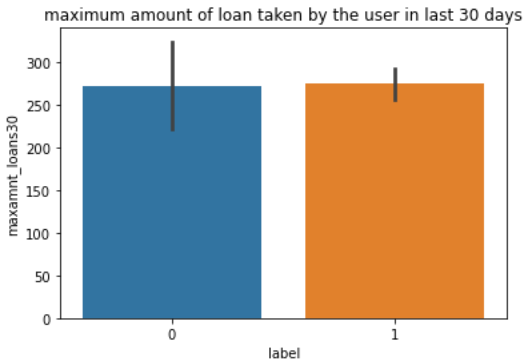
- However, number of loans taken by defaulters in last 90 days seem to have no difference in trend.

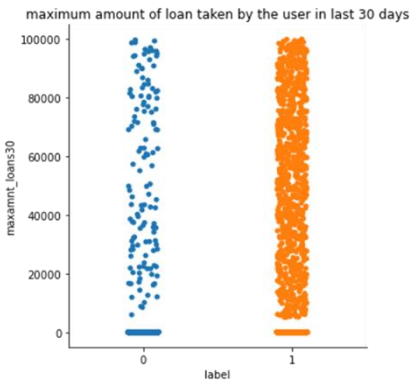


- Maximum amount of loans taken by the user in last 90 days is see to be either 6 or 12 for defaulters and non-defaulters. In case of non-defaulters there are some users who have never taken any loans.

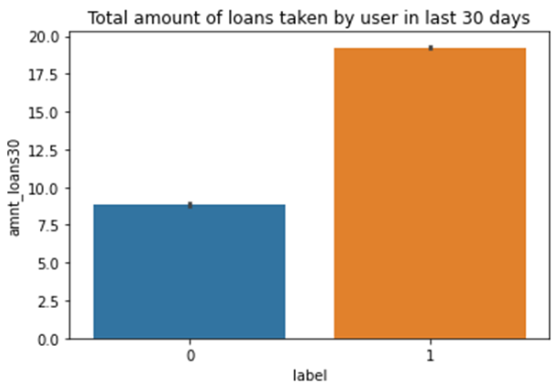


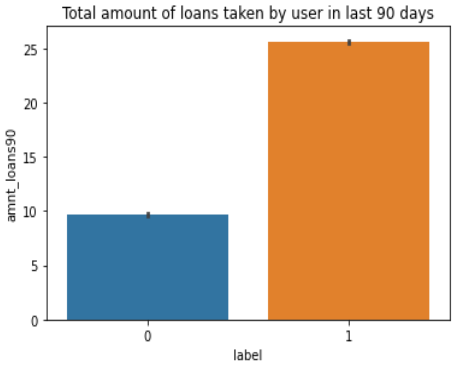
- Median of main account balance just before recharge in last 30 days does not show any trend difference.





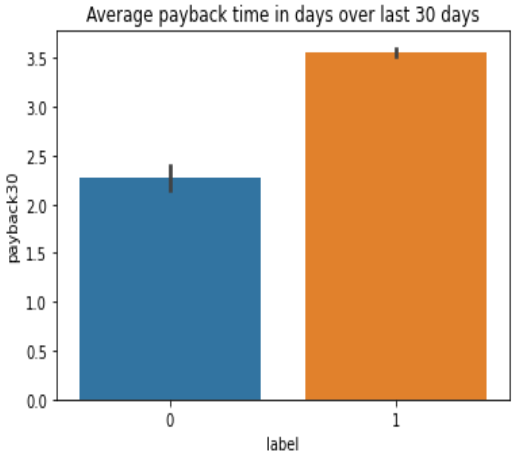
- Maximum amount of loan taken by defaulters and non-defaulters look almost similar.

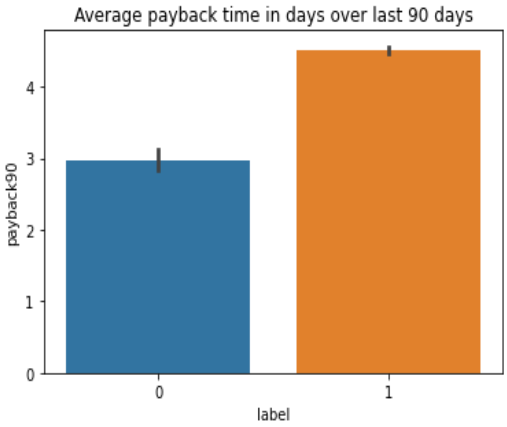




- Total amount of loans taken by defaulters in 30 days is less than non-defaulters. While defaulters on average only took 7-8 loans, non- defaulters are seen to take about 20 loans.

- Total amount of loans taken by defaulters in 90 days is far less than non-defaulters. While defaulters on average only took 10 loans, non- defaulters are seen to take about 25 loans. This is an indication that non- defaulters repay soon within those 3 months so they were able to avail the next loan.





- Average payback time in days over last 30 days is seen to be 2- 2.5 days for defaulters
- Average payback time in days over last 30 days is seen to be 3 days for defaulters
- Average payback time over last 90 days for defaulters is seen to be 3 days and non-defaulters it shows 4 days.

- **Interpretation of the Results**

  From the visualization, we observed that the dataset we dealt with an imbalanced dataset which consisted of 80% of non-defaulters. Hence, we observed a greater density of datapoints in label '1', while plotting each feature while conducting bivariant analysis. Also, our data consisted of a lot of outliers, hence the observations regarding the behaviour with respect to defaulter vs non-defaulter can be a little flawed. During bivariant analysis, we saw defaulters spent less amount from main account when checked for over last 30 days and for 90 days. Their average main account

balance also tends to be less both for 30 days and 90 days. Further, we observed that the number of times that they recharge and the total amount in their main account is lesser. Defaulter are seen to take fewer loans. The total amount of loans for last 30 days and 90 days is also seen to be less in comparison.

After pre-processing the data, we build various models using RandomForestClassifier, DecisionTreeClassifier, AdaBoostClassifier, KNearestNeighborsClassifier, and GradientBoostingClassifier.

Model	**Accuracy**	**Precision**	**Recall**	**ROC-AOC**	**Cross-Validation score**
RandomForestClassifier	95.47%	0.95	0.96	0.99	95.11%
DecisionTreeClassifier	91.21%	0.91	0.92	0.91	90.88%
AdaBoostClassifier	85.11%	0.84	0.87	0.93	84.47%
kNearestNeighborsClassifier	90.31%	0.84	0.99	0.96	90.41%
GradietBoostingClassifier	89.83%	0.89	0.91	0.96	89.47%

To evaluate the performance of the models a common way is by using confusion matrix. Positives are defaulters and Negatives are non- defaulters.

Confusion matrix classifies the cases into

True Positives: if the observed defaulters are same as the predicted defaulters,

False Positives: if non-defaulters are predicted as defaulters

False Negatives: if the defaulters are predicted as non-defaulters

True Negatives: if the non-defaulters are predicted as non-defaulters.

Using confusion matrix, we saw all the models performed with an accuracy above 80%. To make a better decision we used evaluation metrics like Precision, Recall and ROC-AUC curve.

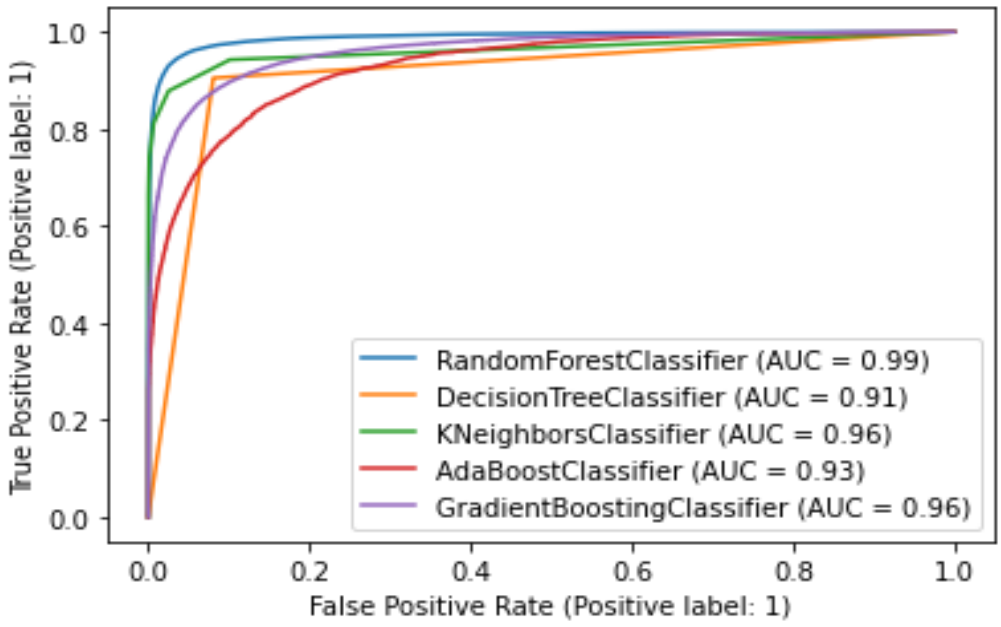
Precision: it tells us a how accurately the model was able to predict defaulters from the predicted defaulters.

$$\text{Precision} = \frac{TP}{TP+FP}$$

Recall: it tells us how many defaulters we were able to catch from the total defaulters

$$\text{Recall} = \frac{TP}{TP+FN}$$

ROC- AUC curve:



ROC graph summarizes all the confusion matrices that each threshold produces. In the graph, the x axis denotes the true positive rates (sensitivity) and y-axis denotes the false negative rates (1- specificity). In an ideal situation, the true positive rate would take the value of 1 and false positive would take the value 0. While ROC is the probability curve, AUC is the area under that curve and tells us how much the model is capable of distinguishing between the classes.

As we can observe, RandomForestClassifier is performing better than the rest and has an AUC of 0.99.

Finally, we also cross-validated the models and found that all models performed similarly as the cross-validation scores were close to the accuracy obtained

Thus, on comparing precision, recall and ROC- AUC curve, it was concluded that RandomForestClassifier should be selected as the best model.

CONCLUSION

Key Findings and Conclusions of the Study

- Defaulters were seen to recharge their main account a smaller number of times.
- They are also seen to take fewer number if loans.

Learning Outcomes of the Study in respect of Data Science

- The best model was selected to be RandomForestClassifier that gave an accuracy, precision, recall and ROC-AUC score of 95.47%, 95%, 96%, 99% respectively.
- However, depending on the business goal, if the company's focus is on having minimum defaulters, then kNearestNeighbors would be the best model with 99% sensitivity. However, the model has only a precision of 84% which may cause a dissatisfaction in customers who were declined from loans even after being re-payers.
- Few challenges that came across while solving the problem was: 1) solving the outliers. Since one of the objectives was to minimize loss of data techniques like Z-score method or IQR could not be used for removing outliers, Hence, the next best option was to replace them with the mean/median values.

Limitations of this work and Scope for Future Work

- Due to time constrain, outliers were replaced by mean/median values. However, a better method should be found in dealing with outliers with minimum data loss.

- The data we dealt with belong to the year 2016. It would be better if we deal with data ranging from other years and preferably from the past few years, as it will be more significant with the current population trend.