FLIP ROBO

Project Report On

"Micro-Credit Defaulter Model"

Submitted by:

Abhilasha Nagaraj Bhat

ACKNOWLEDGMENT

I would like to express my sincere thanks of gratitude to my SME as well as “Flip Robo Technologies” team for letting me work on “Micro-Credit Loan Defaulter Model” project also huge thanks to my academic team “DataTrained”. Their suggestions and directions have helped me in the completion of this project successfully. This project also helped me in doing lots of research wherein I came to know about so many new things.

References:

I have also used few external resources that helped me to complete this project successfully. Below are the external resources that were used to create this project.

1. https://www.google.com/
2. https://scikit-learn.org/stable/index.html
3. https://www.researchgate.net/
4. https://journal-archieves31.webs.com
5. https://github.com/
6. https://towardsdatascience.com/
7. https://www.analyticsvidhya.com/

TABLE OF CONTENTS:

1. INTRODUCTION

Credit defaulter risk is one of the most important risks to be managed by a financial institution. Without loan repayment there is no profit, hence the problem of credit defaulter risk management is relevant to all financial institutions involved in lending to individuals and legal entries. This is even more true with microcredit organizations who have only one product-loans.

Banks have a diverse portfolio so the risk is somewhat mitigated but credit risk is still the most important to manage. Credit risk is economic loss that emanates from the failure of a counterparty to fulfil its contractual obligations or from the increased risk of default during the term of transaction.

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.

Business goal: This case study aims to develop a basic understanding of credit loan defaulter risk analytics in banking and financial services and understand how data is used to minimise the risk of losing money while lending to customers. The loan providing companies find it hard to give loans to the people due to their insufficient or non-existent credit history. Because of that, some consumers use it as their advantage by becoming a defaulter.

A client in Telecom Industry is collaborating with an MFI (Microfinance Institution) 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. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

In this project 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".

Conceptual Background of the Domain Problem

Microfinance is a banking service provided to unemployed or low-income individuals or groups who otherwise would have no other access to financial services. Indonesia is renowned for its large-scale microfinance sector, with a range of commercial banks. Some rural communities in Indonesia have no choice but to seek out loans from unregulated moneylenders. Micro lenders, particularly those operating under Indonesian banks, as well as social enterprise start-ups, are also targeting these communities through their high mobile penetration rates and are developing the right digital platforms to reach out to them.

Generally, Credit Scores plays a vital role for loan approvals, and is very important in today's financial analysis for an individual, Most of the loan lending vendors rely heavily on it, so in our case users has 5 days' time to pay back the loan or else they are listed as defaulters which will impact the loan the credit score heavily, so there are few thing to lookout in this dataset as users who are taking extensive loans, user who have most frequent recharges in their main account have a good chance of 100% payback rate, and user who never recharged their main account for them loan should have never been approved as there is high chance for single user or default user taking multiple connections in name or documents of the family members.

Review of Literature

Literature review covers relevant literature with the aim of gaining insight into the factors that cause loans default within micro finance institutions. The main aim of micro finance is to provide funds for investment in micro businesses that is expected to increase income to investor households and hence improve

their livelihood. It has been observed that most borrowers use micro credit finances on food, shelter and clothing to meet their basic needs rather than investment.

In order to overcome challenges of loan defaults, micro finance institutions use various credit lending models. One of the models in micro finance is rotating savings and credit associations (ROSCA). ROSCAs form groups of individuals who pay into an account on a monthly basis. Each individual then earns an opportunity to receive a relatively large loan with to invest. The group decides who receives the loan each term, often based on rotating schedule. The initial money is either accumulation of the group members' individual deposits or more frequently, by an outside donation. Loan repayment is ensured through peer pressure. Anyone who does not repay the loan amount risks the privilege to borrow in the future.

Motivation for the Problem Undertaken

The main objective of this study is to investigate which method from a chosen set of machine learning techniques performs the best default prediction. This project was highly motivated project as it includes the real time problem for Microfinance Institution (MFI), and to the poor families in remote areas with low income, and it is related to financial sectors, as I believe that with growing technologies and Idea can make a difference, there are so much in the financial market to explore and analyse and with Data Science the financial world becomes more interesting.

The project gives an insight to identify major factors that lead to credit risk portfolio in microfinance banks and provide recommendations aimed at mitigating credit risks in microfinance banks. With the help of independent variables available in the dataset we need to model the micro credit defaulters' level in the micro finance institution. This model will help the management to understand how the users considered as defaulter or non-defaulter based on the attributes available. The model will be a good way for the management to understand whether the customer will be paying back the loaned amount within 5 days of issuing loan. We are provided with sample data, in order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

2. ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modelling of the Problem:

We need to build a Machine Learning 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 the dataset, the 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.

Clearly it is a binary classification problem where we need to use classification algorithms to predict the results. There were no null values in the dataset. There were some unwanted entries like more than 90% of zero values present in some of the columns which means these customers have no loan history so, I have dropped those columns. I found some negative values while summarizing the statistics of the dataset, I have converted them into positive. To get better insights on features I have used some plots like pie plot, count plot, bar plot, distribution plot, box plots etc. There were lots of skewness and outliers present in our dataset which need to be cleaned using appropriate techniques and balanced the data. At last, I have built many classification models to predict the defaulter level at the institution.

Data Sources and their formats

Data set provided by Flip Robo was in the format of CSV (Comma Separated Values). The dimension of the dataset is 209593 rows and 37 columns including target variable "label". In the particular dataset maximum number of columns are of Float type and Integer type and very few are of object type. The attribution information is as follows:

Variables	Definition
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 users
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 Indonesian Rupiah)
medianamnt_ma_rech90	Median of amount of recharges done in main account over last 90 days at user level (in Indonesian Rupiah)
medianmarechprebal90	Median of main account balance just before recharge in last 90 days at user level (in Indonesian 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
payback90	Average payback time in days over last 90 days
pcircle	telecom circle
pdate	date
Unnamed:0	User Identifier

Data Pre-processing Done

Data pre-processing is the process of converting raw data into a well-readable format to be used by Machine Learning model. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre-process our data before feeding it into our model. I have used following pre-processing steps:

- Importing necessary libraries and loading dataset as a data frame.
- Used pandas to set display maximum columns ensuring not to find any truncated information.
- Checked some statistical information like shape, number of unique values present, info, finding zero values etc.
- Checked for null values and did not find any null values.
- Dropped some unwanted columns like Unnamed:0, pcircle, msisdn as they are of no use for prediction.
- Dealt with zero values by verifying the percentage of zero values in each column and decided to discard the columns having more than 90% of zero values.
- Converted time variable “pdate” from object into datetime and extracted Day, Month and Year for better understanding. Checked value counts for each and dropped Year column as it contains unique value throughout the dataset.
- Checked unique values and value counts of target variable.
- Converted the data having values other than 6, 12 & 0 into 0 in the column maxamnt_loans30. As it is specified in the problem statement that we should have only 0, 6 & 12 values. Also, discarded some rows in the column amnt_loans90 as it gives the sum of loans taken by the user in 90 days
- While checking the statistical summary of the dataset, I found some columns having negative values which were invalid and unrealistic so decided to convert negative values into positive using absolute command.
- Visualized each feature using seaborn and matplotlib libraries by plotting several categorical and numerical plots like pie plot, count plot, bar plot, distribution plot, box plots etc.
- Identified outliers using box plots and I tried to remove them using both Zscore and IQR method and got huge data loss of around 18% and 62% respectively, so removed outliers using percentile method by setting data loss to 2%.

- Checked for skewness and removed skewness in numerical columns using power transformation method (yeo-johnson).
- Used Pearson's correlation coefficient to check the correlation between label and features. With the help of heatmap, correlation bar graph was able to understand the Feature vs Label relativity and insights on multicollinearity amongst the feature columns.
- Separated feature and label data and feature scaling is performed using MinMaxScalar method to avoid any kind of data biasness.
- Since the dataset was imbalanced. Label '1' had approximately 87.5% records, while, label '0' had approximately 12.5% records. So, performed Oversampling method using SMOTE to balance the data.
- Checked for the best random state to be used on our Classification Machine Learning model pertaining to the feature importance details.
- Finally created classification model along with evaluation metrics.

Data Inputs- Logic- Output Relationships

The dataset consists of label and features. The features are independent and label is dependent as the values of our independent variables changes as our label varies.

- Since we had only numerical columns so, I checked the distribution of skewness using dist plots as a part of univariate analysis.
- To analyse the relation between features and label I have used many plotting techniques where I found some of the columns having strong relation with label.
- The visualization helped me to understand that maximum distribution is for non-defaulter for all the features & maximum defaulter list are from people who have Average payback time in days over last 30 & 90 days, also frequency of recharge done in the main account since last 90 days. So, the features, which I have kept after dropping few are having some kind of relationship with the output.
- I have checked the correlation between the label and features using heat map and bar plot. Where I got the positive correlation between the label and features and there was no much relation.

Hardware & Software Requirements & Tools Used

To build the machine learning projects it is important to have the following hardware and software requirements and tools.

Hardware required:

- Processor: core i5 or above
- RAM: 8 GB or above
- ROM/SSD: 250 GB or above

Software required:

- Distribution: Anaconda Navigator
- Programming language: Python
- Browser based language shell: Jupyter Notebook

Libraries required:

```
# Preprocessing
import numpy as np
import pandas as pd
# Visualization
import seaborn as sns
import matplotlib.pyplot as plt

import os
import scipy as stats

import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
```

- ✓ **import numpy as np:** It is defined as a Python package used for performing the various numerical computations and processing of the multidimensional and single dimensional array elements. The calculations using Numpy arrays are faster than the normal Python array.

- ✓ **import pandas as pd:** Pandas is a Python library that is used for faster data analysis, data cleaning and data pre-processing. The data-frame term is coming from Pandas only.

- ✓ **import matplotlib.pyplot as plt:** Matplotlib and Seaborn acts as the backbone of data visualization through Python.

  Matplotlib: It is a Python library used for plotting graphs with the help of other libraries like Numpy and Pandas. It is a powerful tool for visualizing data in Python. It is used for creating statical interferences and plotting 2D graphs of arrays.

- ✓ **import seaborn as sns: Seaborn** is also a Python library used for plotting graphs with the help of Matplotlib, Pandas, and Numpy. It is built on the roof of Matplotlib and is considered as a superset of the Matplotlib library. It helps in visualizing univariate and bivariate data.
- ✓ from scipy.stats import zscore
- ✓ from sklearn.preprocessing import PowerTransformer
- ✓ from sklearn.preprocessing import MinMaxScaler
- ✓ from statsmodel.starts.outliers_influence import variance_inflation_factor
- ✓ from imblearn.over_sampling import SMOTE

With the above sufficient libraries, we can perform pre-processing and data cleaning. For building my ML models Below libraries are required.

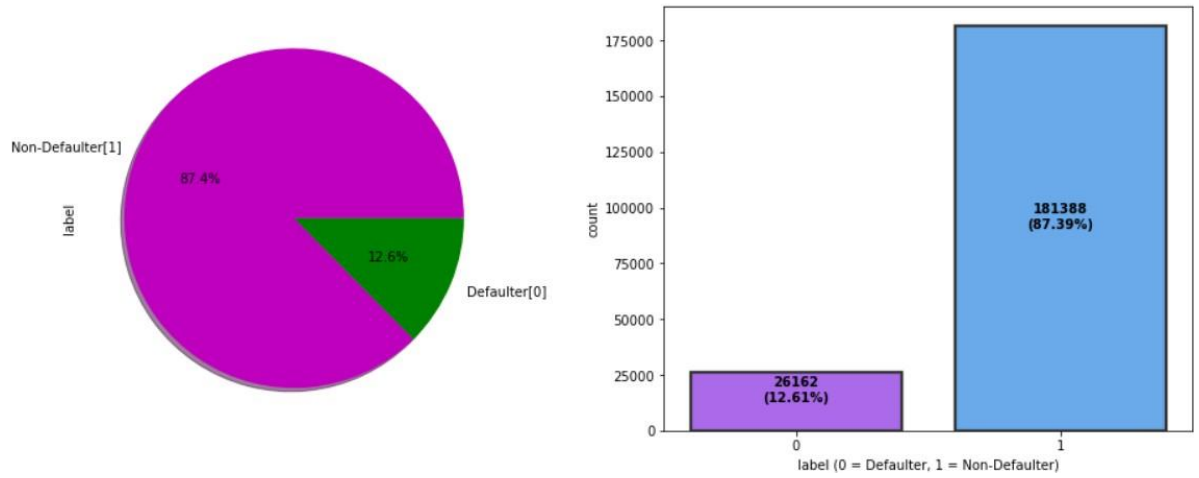
- ✓ from sklearn.model_selection import train_test_split
- ✓ from sklearn.tree import DecisionTreeClassifier
- ✓ from sklearn.ensemble import RandomForest Classifier, ExtraTreesClassifier
- ✓ from sklearn.ensemble import GradientBoostingClassifier, BaggingClassifier
- ✓ from xgboost import XGBClassifier as xgb
- ✓ from sklearn.metrics import classification_report, confusion_matrix, roc_curve, accuracy_score, roc_auc_score
- ✓ from sklearn.model_selection import cross_val_score

3. MODEL/S DEVELOPMENT AND EVALUATION

Identification of possible Problem-solving approaches (Methods):

I have used both statistical and analytical approaches to solve the problem which mainly includes the pre-processing of the data also used EDA techniques and heat map to check the correlation of independent and dependent features. Also, before building the model, I made sure that the input data is cleaned and scaled before it was fed into the machine learning models. The data mainly had class imbalancing issue which looks like below.

```
1    181388
0     26162
Name: label, dtype: int64
```



From the above we can see that the data set is highly imbalanced, so applied SMOTET method to balance the dataset.

For this particular project we need to predict whether the user paid back the credit loan amount within 5 days of issuing the loan. In this dataset, label is the target variable, which consists of two categories, defaulters and non-defaulters. Which means our target column is categorical in nature so this is a classification problem.

I have used many classification algorithms and got the prediction results. By doing various evaluations I have selected Gradient Boosting Classifier as best suitable algorithm to create our final model as it is giving least difference in accuracy score and cross validation score among all the algorithms used.

In order to get good performance and to check whether my model getting over-fitting and under-fitting I have made use of the K-Fold cross validation and then hyper parameter tuning on best model. Then I saved my final model and loaded the same for predictions.

Testing of Identified Approaches (Algorithms)

Since label is my target variable which is categorical in nature, from this I can conclude that it is a classification type problem hence I have used following classification algorithms. After the pre-processing and data cleaning I left with 27 columns including target and with the help of feature importance bar graph I used these independent features for model building and prediction. The algorithms used on training the data are as follows:

1. Decision Tree Classifier
2. Random Forest Classifier
3. Extra Trees Classifier
4. Gradient Boosting Classifier
5. Extreme Gradient Boosting Classifier (XGB)
6. Bagging Classifier

Run and evaluate selected models

I have used 6 classification algorithms after choosing random state amongst 1-1000 number. I have used Decision Tree Classifier to find best random state and the code is as below:

Finding best random state for building Classification Models

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
maxAccu=0
maxRS=0
for i in range(1,200):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state =i)
    DTC = DecisionTreeClassifier()
    DTC.fit(x_train, y_train)
    pred = DTC.predict(x_test)
    acc=accuracy_score(y_test, pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best accuracy is ",maxAccu," on Random_state ",maxRS)
```

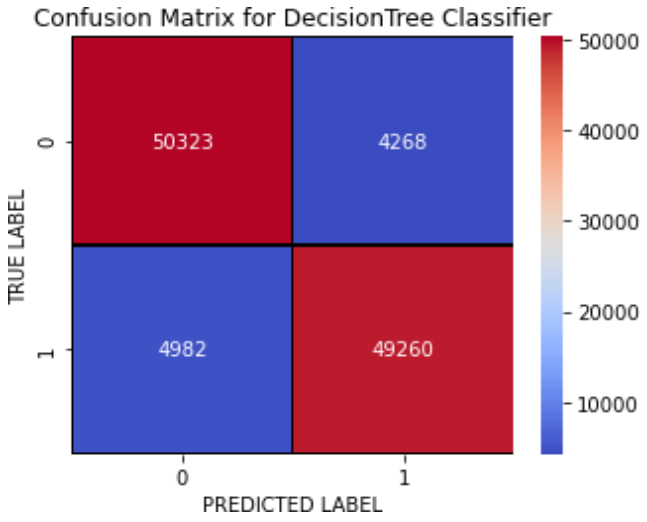
```
Best accuracy is  0.9150992805490982  on Random_state  97
```

Model Building:

1. DecisionTreeClassifier:

```
# Checking Accuracy and evaluation metrics for Decision Tree Classifier
DTC = DecisionTreeClassifier()
DTC.fit(x_train,y_train)                          # Training the model
predDTC = DTC.predict(x_test)                     #Predicting y_test

DTC_score = accuracy_score(y_test, predDTC)*100   # Accuracy Score
print("Accuracy Score:", DTC_score)

from sklearn.metrics import roc_auc_score
roc_auc_score = roc_auc_score(y_test,predDTC)*100  # ROC AUC Score
print("\nroc_auc_score:", roc_auc_score)
conf_matrix = confusion_matrix(y_test, predDTC)    # Confusion Matrix
print("\nConfusion Matrix:\n",conf_matrix)
class_report = classification_report(y_test,predDTC) # Classification Report
print("\nClassification Report:\n", class_report)

# Cross Validation Score
cv_score = (cross_val_score(DTC, x, y, cv=5).mean())*100
print("Cross Validation Score:", cv_score)

# Result of accuracy minus cv scores
Result = DTC_score - cv_score
print("\nAccuracy Score - Cross Validation Score is", Result)
```

```
Accuracy Score: 91.50073966535885

roc_auc_score: 91.49854845271034

Confusion Matrix:
 [[50323  4268]
 [ 4982 49260]]

Classification Report:
               precision    recall  f1-score   support

           0       0.91      0.92      0.92     54591
           1       0.92      0.91      0.91     54242

    accuracy                           0.92    108833
   macro avg       0.92      0.91      0.91    108833
weighted avg       0.92      0.92      0.92    108833

Cross Validation Score: 90.87234894174678

Accuracy Score - Cross Validation Score is 0.6283907236120712
```



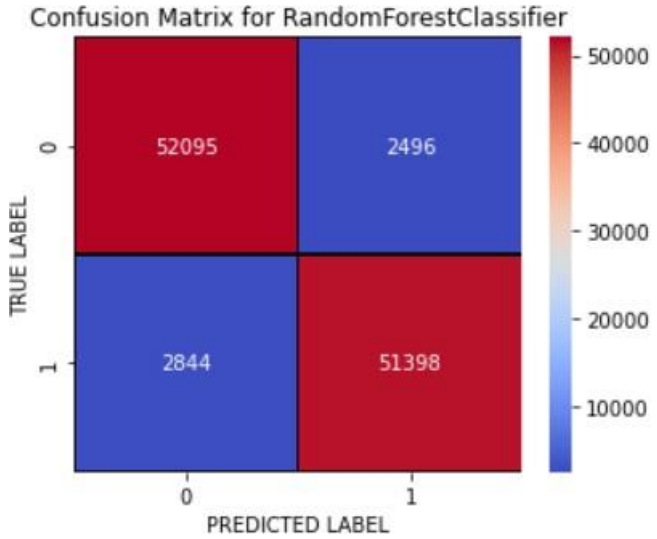
♣ Created Decision Tree Classifier model and checked for its evaluation metrics and it is giving accuracy 91.50%.

♣ We can see the true values and predicted values in Decision Tree Classifier model using confusion matrix.

2. RandomForestClassifier:

```
# Checking Accuracy and evaluation metrics for Random Forest Classifier
RFC = RandomForestClassifier()

RFC.fit(x_train,y_train)                              # Training the model
predRFC = RFC.predict(x_test)                         #Predicting y_test

RFC_score = accuracy_score(y_test, predRFC)*100       # Accuracy Score
print("Accuracy Score:", RFC_score)
roc_auc_score2 = roc_auc_score(y_test,predRFC)*100    #ROC AUC Score
print("\nroc_auc_score:", roc_auc_score2)
conf_matrix = confusion_matrix(y_test, predRFC)       # Confusion Matrix
print("\nConfusion Matrix:\n",conf_matrix)
class_report = classification_report(y_test,predRFC) # Classification Report
print("\nClassification Report:\n", class_report)

# Cross Validation Score
cv_score2 = (cross_val_score(RFC, x, y, cv=5).mean())*100
print("Cross Validation Score:", cv_score2)

# Result of accuracy minus cv scores
Result = RFC_score - cv_score2
print("\nAccuracy Score - Cross Validation Score is", Result)
```

```
Accuracy Score: 95.09339997978554

roc_auc_score: 95.09232413625527

Confusion Matrix:
 [[52095  2496]
 [ 2844 51398]]

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95     54591
           1       0.95      0.95      0.95     54242

    accuracy                           0.95    108833
   macro avg       0.95      0.95      0.95    108833
weighted avg       0.95      0.95      0.95    108833

Cross Validation Score: 94.91313431042722

Accuracy Score - Cross Validation Score is 0.18026566935832022
```



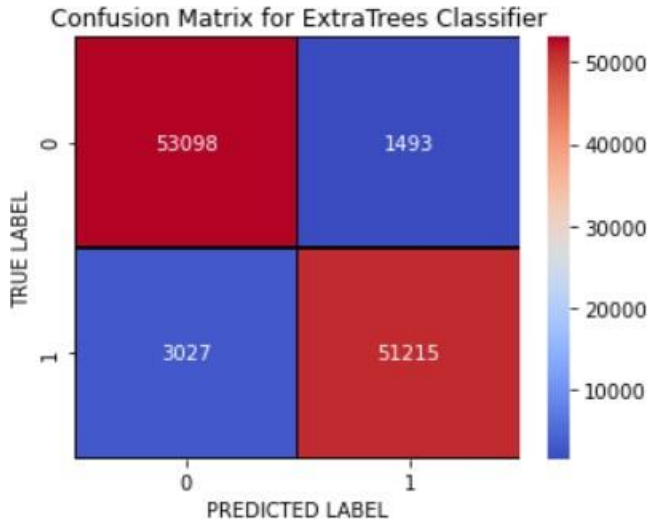
♣ Created RandomForestClassifier model and checked for its evaluation metrics. The model giving 95.09% accuracy.

♣ With the help of confusion matrix, we can able to observe the true positive values and predicted values in RandomForest Classifier model.

3. ExtraTreesClassifier:

```
# Checking Accuracy and evaluation metrics for ExtraTrees Classifier
XT = ExtraTreesClassifier()
XT.fit(x_train,y_train)                              # Training the model
predXT = XT.predict(x_test)                         #Predicting y_test
XT_score = accuracy_score(y_test, predXT)*100      # Accuracy Score
print("Accuracy Score:", XT_score)
roc_auc_score3 = roc_auc_score(y_test,predXT)*100       # ROC AUC Score
print("\nroc_auc_score:", roc_auc_score3)
conf_matrix = confusion_matrix(y_test, predXT)      # Confusion Matrix
print("\nConfusion Matrix:\n",conf_matrix)
class_report = classification_report(y_test,predXT) # Classification Report
print("\nClassification Report:\n", class_report)

# Cross Validation Score
cv_score3 = (cross_val_score(XT, x, y, cv=5).mean())*100
print("Cross Validation Score:", cv_score3)

# Result of accuracy minus cv scores
Result = XT_score - cv_score3
print("\nAccuracy Score - Cross Validation Score is", Result)
```

```
Accuracy Score: 95.84684792296454

roc_auc_score: 95.84228526032157

Confusion Matrix:
 [[53098  1493]
 [ 3027 51215]]

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96     54591
           1       0.97      0.94      0.96     54242

    accuracy                           0.96    108833
   macro avg       0.96      0.96      0.96    108833
weighted avg       0.96      0.96      0.96    108833

Cross Validation Score: 96.31067393951886

Accuracy Score - Cross Validation Score is -0.46382601655432154
```



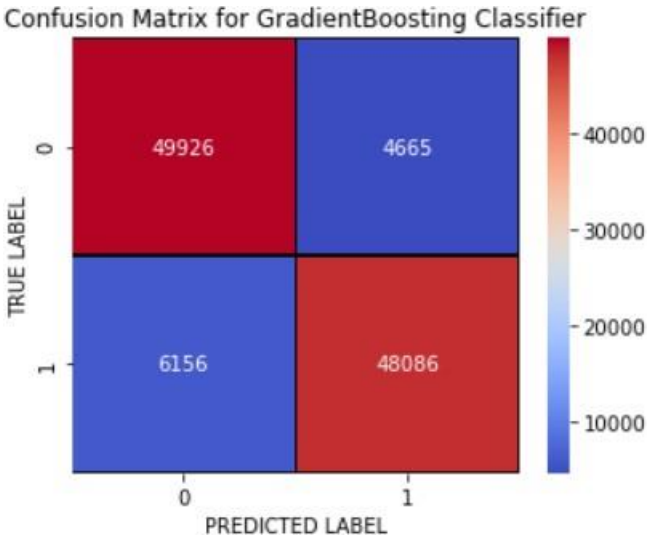
♣ Created the ExtraTrees Classifier model and checked for its evaluation metrics. The ExtraTrees Classifier model giving 95.84% accuracy.

♣ With the help of confusion matrix, we can able observe the true positive values and predicted values in Extra Trees Classifier model.

4. GradientBoostingClassifier:

```
# Checking Accuracy and evaluation metrics for GradientBoosting Classifier
GB = GradientBoostingClassifier()

GB.fit(x_train,y_train)                              # Training the model
predGB = GB.predict(x_test)                         #Predicting y_test

GB_score = accuracy_score(y_test, predGB)*100      # Accuracy Score
print("Accuracy Score:", GB_score)

roc_auc_score4 = roc_auc_score(y_test,predGB)*100       # ROC AUC Score
print("\nroc_auc_score:", roc_auc_score4)

conf_matrix = confusion_matrix(y_test, predGB)      # Confusion Matrix
print("\nConfusion Matrix:\n",conf_matrix)

class_report = classification_report(y_test,predGB) # Classification Report
print("\nClassification Report:\n", class_report)

cv_score4 = (cross_val_score(GB, x, y, cv=5).mean())*100  # Cross Validation Score
print("Cross Validation Score:", cv_score4)

# Result of accuracy minus cv scores
Result = GB_score - cv_score4
print("\nAccuracy Score - Cross Validation Score is", Result)
```

```
Accuracy Score: 90.0572436669025

roc_auc_score: 90.05274816795628

Confusion Matrix:
 [[49926  4665]
 [ 6156 48086]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.91      0.90     54591
           1       0.91      0.89      0.90     54242

    accuracy                           0.90    108833
   macro avg       0.90      0.90      0.90    108833
weighted avg       0.90      0.90      0.90    108833

Cross Validation Score: 89.90425411080916

Accuracy Score - Cross Validation Score is 0.1529895560933312
```

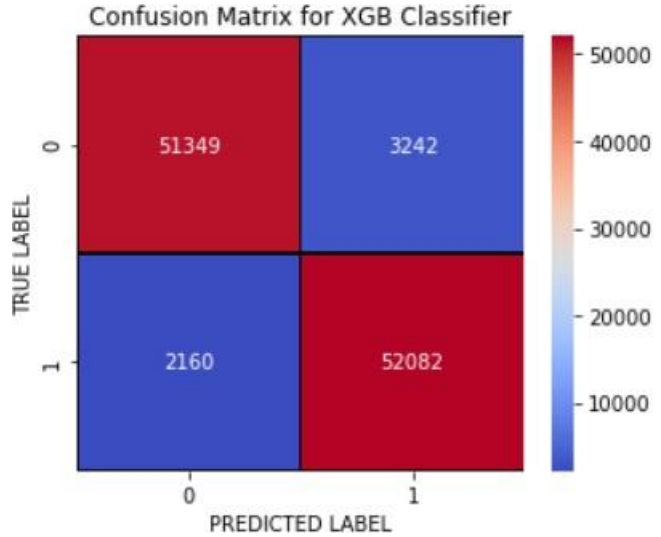


- Created Gradient Boosting Classifier and checked for its evaluation metrics. The model giving 90.05% accuracy.
- With the help of confusion matrix, we can able observe the true positive values and predicted values in Gradient Boosting Classifier model.

5. Extreme Gradient Boosting Classifier (XGB Classifier)

```
# Checking Accuracy and evaluation metrics for XGB Classifier
XGB = xgb(verbosity=0)

XGB.fit(x_train,y_train)                              # Training the model
predXGB = XGB.predict(x_test)                         #Predicting y_test

XGB_score = accuracy_score(y_test, predXGB)*100       # Accuracy Score
print("Accuracy Score:", XGB_score)

roc_auc_score5 = roc_auc_score(y_test,predXGB)*100        # ROC AUC Score
print("\nroc_auc_score:", roc_auc_score5)

conf_matrix = confusion_matrix(y_test, predXGB)       # Confusion Matrix
print("\nConfusion Matrix:\n",conf_matrix)
class_report = classification_report(y_test,predXGB) # Classification Report
print("\nClassification Report:\n", class_report)

cv_score5 = (cross_val_score(XGB, x, y, cv=5).mean())*100  # Cross Validation Score
print("Cross Validation Score:", cv_score5)

# Result of accuracy minus cv scores
Result = XGB_score - cv_score5
print("\nAccuracy Score - Cross Validation Score is", Result)
```

```
Accuracy Score: 95.03643196456957

roc_auc_score: 95.03956905200877

Confusion Matrix:
 [[51349  3242]
 [ 2160 52082]]

Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.94      0.95     54591
           1       0.94      0.96      0.95     54242

    accuracy                           0.95    108833
   macro avg       0.95      0.95      0.95    108833
weighted avg       0.95      0.95      0.95    108833

Cross Validation Score: 93.61703980816385

Accuracy Score - Cross Validation Score is 1.4193921564057206
```



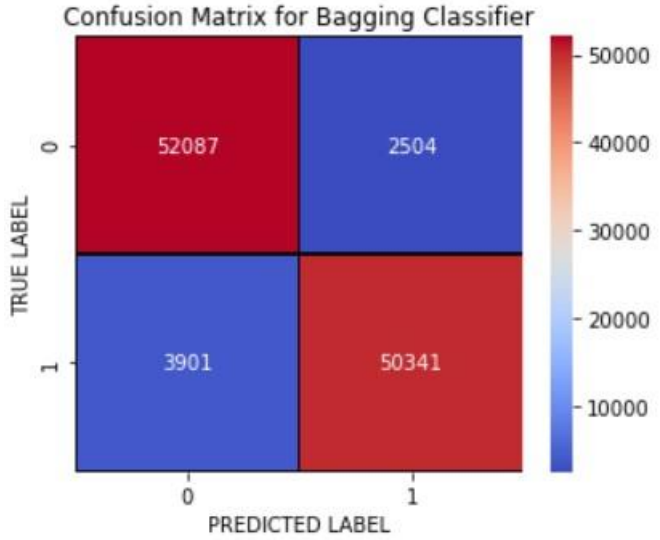
♣ Created XGB Classifier and checked for its evaluation metrics. The model giving 95.03% accuracy.

♣ With the help of confusion matrix, we can able observe the true positive values and predicted values in XGB Classifier model.

6. BaggingClassifier:

```
# Checking Accuracy and evaluation metrics for Bagging Classifier
BC = BaggingClassifier()

BC.fit(x_train,y_train)                              # Training the model
predBC = BC.predict(x_test)                         #Predicting y_test

BC_score = accuracy_score(y_test, predBC)*100      # Accuracy Score
print("Accuracy Score:", BC_score)

roc_auc_score6 = roc_auc_score(y_test,predBC)*100       # ROC AUC Score
print("\nroc_auc_score:", roc_auc_score6)

conf_matrix = confusion_matrix(y_test, predBC)      # Confusion Matrix
print("\nConfusion Matrix:\n",conf_matrix)

class_report = classification_report(y_test,predBC) # Classification Report
print("\nClassification Report:\n", class_report)

cv_score6 = (cross_val_score(BC, x, y, cv=5).mean())*100  # Cross Validation Score
print("Cross Validation Score:", cv_score6)

# Result of accuracy minus cv scores
Result = BC_score - cv_score6
print("\nAccuracy Score - Cross Validation Score is", Result)
```

```
Accuracy Score: 94.11483649260795

roc_auc_score: 94.11065969169336

Confusion Matrix:
 [[52087  2504]
 [ 3901 50341]]

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.95      0.94     54591
           1       0.95      0.93      0.94     54242

    accuracy                           0.94    108833
   macro avg       0.94      0.94      0.94    108833
weighted avg       0.94      0.94      0.94    108833

Cross Validation Score: 93.64761563823926

Accuracy Score - Cross Validation Score is 0.46722085436869065
```



♣ Created Bagging Classifier and checked for its evaluation metrics. The model giving 94.11% accuracy.

♣ With the help of confusion matrix, we can able observe the true positive values and predicted values in Bagging Classifier model.

Model Selection:

Model	Accuracy_Score	Cross_Validation_Score	Difference
DecisionTreeClassifier	91.05	90.872	0.628
RandomForestClassifier	95.093	94.913	0.180
ExtraTreesClassifier	95.846	96.310	-0.463
GradientBoostingClassifier	90.057	89.904	0.152
XGBClassifier	95.036	93.617	1.419
BaggingClassifier	94.114	93.647	0.467

From the difference between accuracy and cross validation score, GradientBoostingClassifier has least difference compared to other models. So, we can conclude that GradientBoostingClassifier as our best fitting model. Performed Hyperparameter tuning to increase the best model accuracy.

Hyper Parameter Tuning:

```
from sklearn.model_selection import RandomizedSearchCV

# GradientBoosting Classifier
parameters = {'max_depth': [3,6,9],
              'max_features':['auto', 'sqrt', 'log2'],
              'learning_rate':[0.1,0.25,0.5],
              'min_samples_leaf': [1,50,100]}
```

Have used 4 GradientBoostingClassifier parameters to be saved under the variable "parameters" that will be used in RandomizedSearchCV for finding the best output.

```
RCV=RandomizedSearchCV(GradientBoostingClassifier(),parameters,cv=5,n_iter =10)
```

Assigning a variable to the RandomizedSearchCV function after entering all the necessary inputs.

```
RCV.fit(x_train,y_train)
```

```
RandomizedSearchCV(cv=5, estimator=GradientBoostingClassifier(),
                   param_distributions={'learning_rate': [0.1, 0.25, 0.5],
                                        'max_depth': [3, 6, 9],
                                        'max_features': ['auto', 'sqrt',
                                                         'log2'],
                                        'min_samples_leaf': [1, 50, 100]})
```

Now we use our training data set to make the RandomizedSearchCV aware of all the hyper parameters that needs to be applied on our best model.

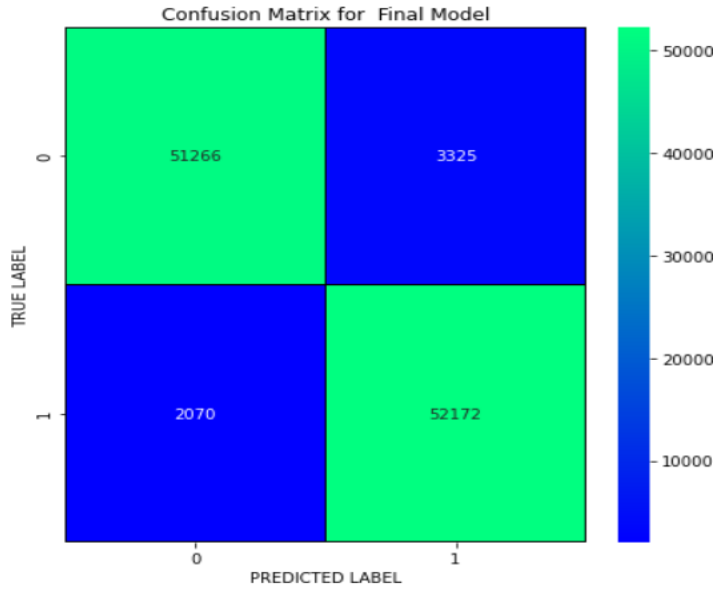
```
RCV.best_params_
```

```
{'min_samples_leaf': 100,
 'max_features': 'auto',
 'max_depth': 6,
 'learning_rate': 0.5}
```

This gives us the list of parameters which will be used further in our final model creation.

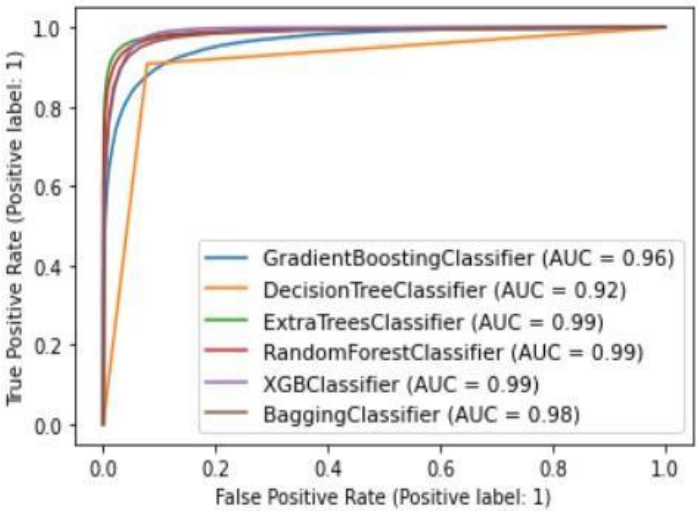
```
# Creating final model
MicroCredit_model = GradientBoostingClassifier(min_samples_leaf = 100, max_features = "auto", max_depth = 6, learning_rate = 0.5)
MicroCredit_model.fit(x_train, y_train)
pred = MicroCredit_model.predict(x_test)
acc_score = accuracy_score(y_test,pred)
print("Accuracy score for the Best Model is:", acc_score*100)
```

```
Accuracy score for the Best Model is: 95.04286383725524
```

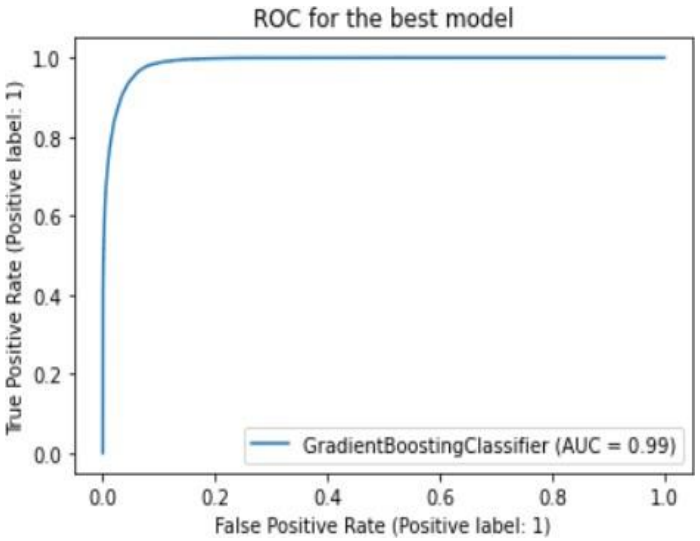


I have successfully incorporated the hyper parameter tuning using best parameters of GradientBoostingClassifier and the accuracy of the model has been increased by 5% after hyperparameter tuning and received the accuracy score as 95.04% which is very good. With the help of confusion matrix, we can able to see actual and predicted values for the final model. And also, we can understand the number of times we got the correct outputs and the number of times my model missed to provide the correct prediction.

ROC-AUC Curve for all the models used and for best model:



ROC-AUC Curve for all the models



ROC-AUC Curve for final model

I have generated the ROC Curve for all the models and for the best model and it shows the AUC score for the models. The AUC score for my final model to be of 99% which is increased after tuning the model.

Key Metrics for success in solving problem under consideration

The key metrics used here were Accuracy Score, Precision, Recall, F1 score, Cross Validation Score, Roc Auc Score and Confusion Matrix. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning and used RandomizedSearchCV method.

- **Accuracy score** means how accurate our model is that is the degree of closeness of the measured value to a standard or true value. It is one metric for evaluating classification models. Accuracy is the ratio of number of correct predictions into number of predictions.
- **Precision** is the degree to which repeated measurements under the same conditions are unchanged. It is amount of information that is conveyed by a value. It refers to the data that is correctly classified by the classification algorithm.
- **Recall** is how many of the true positives were recalled (found). Recall refers to the percentage of data that is relevant to the class. In binary classification problem recall is calculated as below:
  Recall = Number of True Positives/ (Total number of True Positives + Total number of False Negatives)
- **F1 Score** is used to express the performance of the machine learning model (or classifier). It gives the combined information about the precision and recall of a model. This means a high F1-score indicates a high value for both recall and precision.
- **Cross Validation Score** is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set. It is used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited. In cross-validation, you make a fixed number of folds (or partitions) of the data, run the analysis on each fold, and then average the overall error estimate. It is used to estimate the performance of ML models.
- **Roc Auc Score:** The **Receiver Operator Characteristic (ROC)** curve is an evaluation metric for binary classification problems. It is a probability curve that plots the **TPR** against **FPR** at various threshold values.
  The **Area Under Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

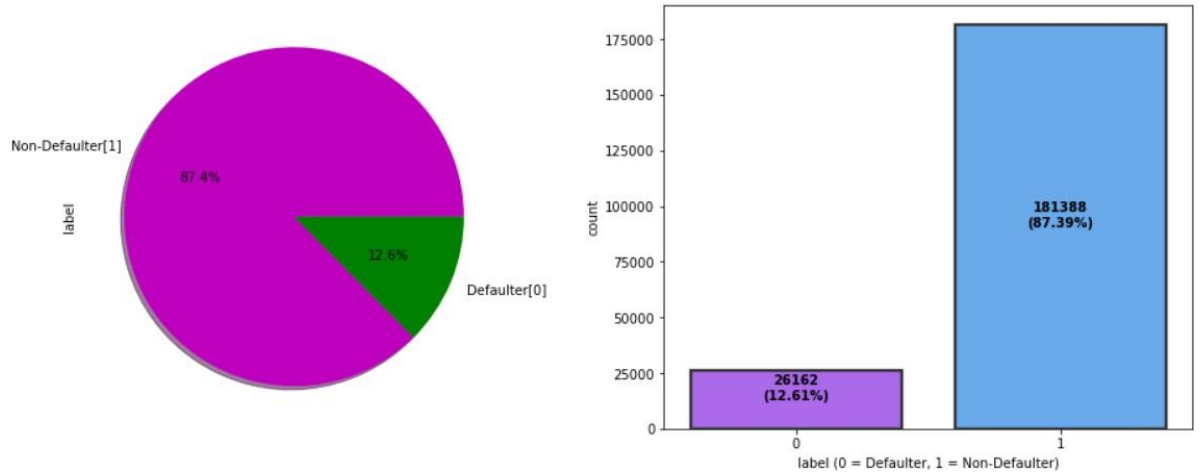
- **Confusion Matrix** is one of the evaluation metrics for machine learning classification problems, where a trained model is being evaluated for accuracy and other performance measures. And this matrix is called the confusion matrix since it results in an output that shows how the system is confused between the two classes.

Visualizations

I used pandas profiling to get the over viewed visualization on the pre-processed data. Pandas is an open-source Python module with which we can do an exploratory data analysis to get detailed description of the features and it helps in visualizing and understanding the distribution of each variable. I have analysed the data using both univariate and bivariate analysis. In univariate analysis I have used distribution plot, pie plot and count plot and in bivariate analysis I have used bar plots. These plots have given good pattern.

Univariate Analysis: Visualizing label whether the user paid back the credit amount within 5 days of issuing the loan or not.

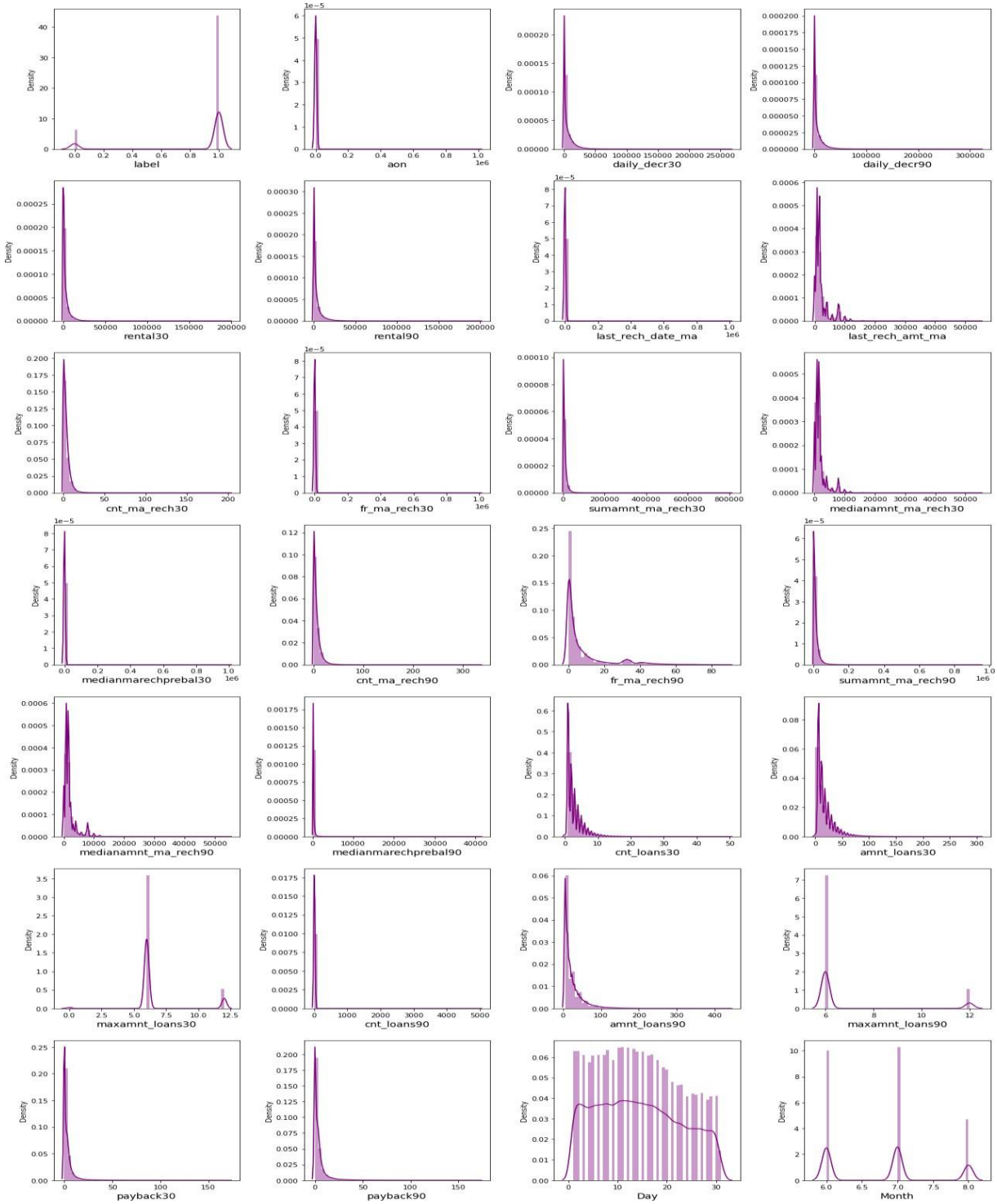
```
1    181388
0     26162
Name: label, dtype: int64
```



Observations:

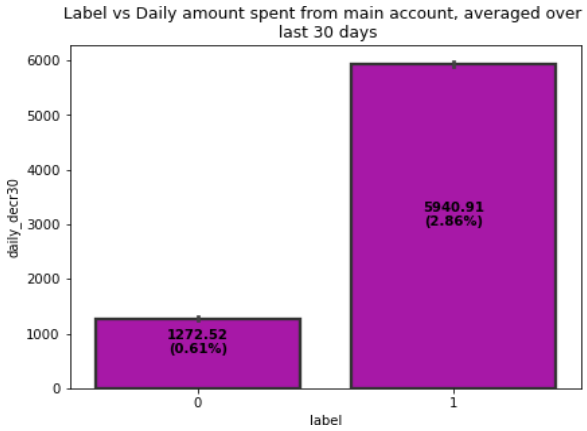
- ✓ From the above plots we can observe around 87% of the loan has been paid by the user and only 12% of the loan failed to pay. Since the data was not balanced, I have used SMOTE method to balance it that I already have been mentioned.

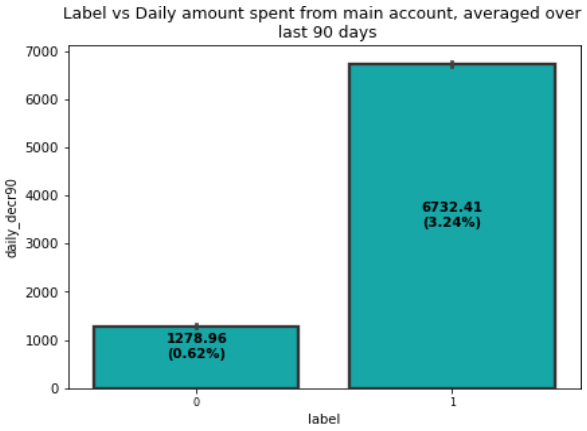
Distribution plots to check the skewness in numerical columns



- ✓ From the above distribution plot, I can observe most of the columns are not normally distributed only Day column somewhat distributed normally.
- ✓ All the columns have skewness and are skewed to right since the mean is greater than the median in these columns. We need to remove this skewness before building our machine learning models.

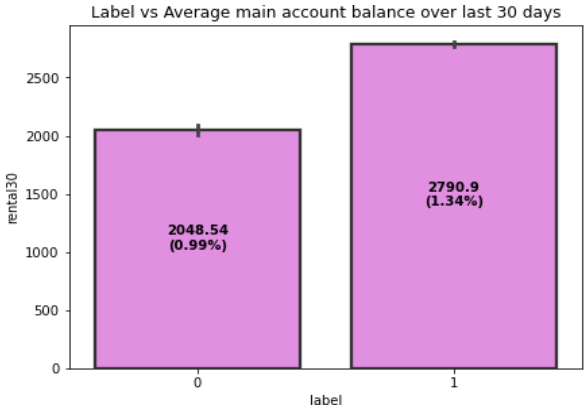
Bivariate Analysis: Comparing label with remaining features:

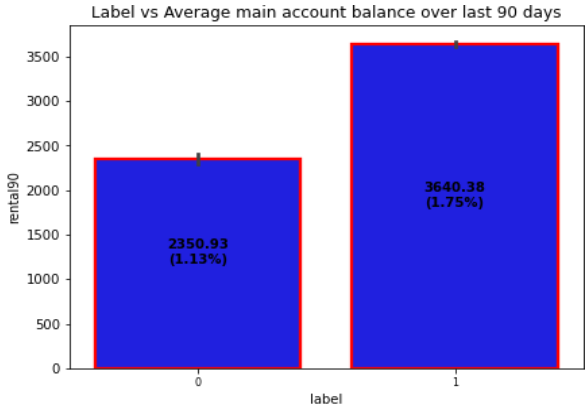




Observations:

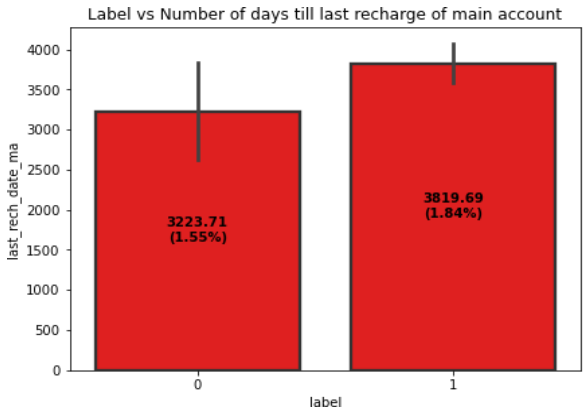
- ✓ Most of the users who have paid back the credit amount within 5 days of issuing loan, they have high rate of daily amount spent from the account over last 30 days and 90 days which have the count around 5940 and 6732 respectively.
- ✓ The users who have spent daily amount from main account over last 30 days and 90 days have always paid back the loan amount within 5 days. Around 0.6% of the users failed to pay back the loan within due date.

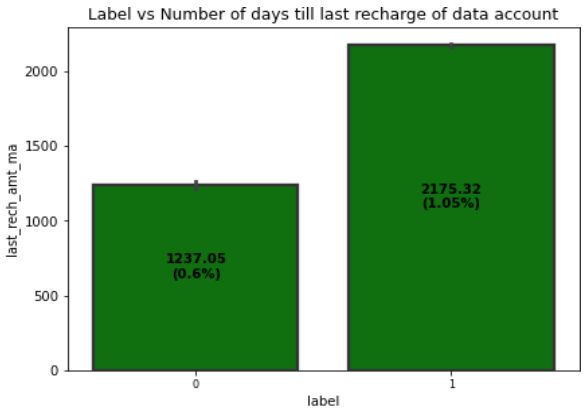




Observations:

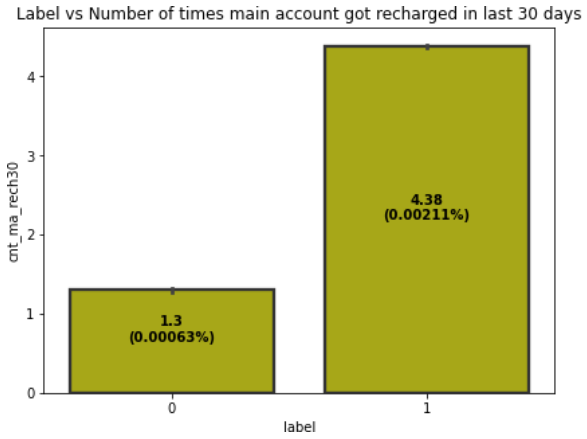
- ✓ Non defaulter users have average main account balance over last 30 days and 90 days which have count around 2790 & 3640 compared to defaulter.
- ✓ That means the users who have average main account balance always pays back the credit amounts within 5 days. And around 1% of the users either failed to pay back the loan amount within the due date or they are not paying the loan.

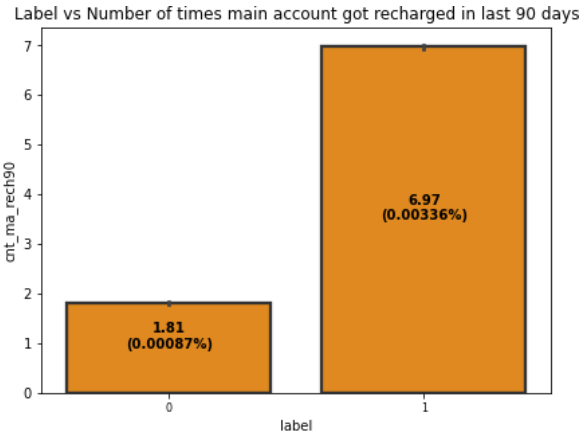




Observations:

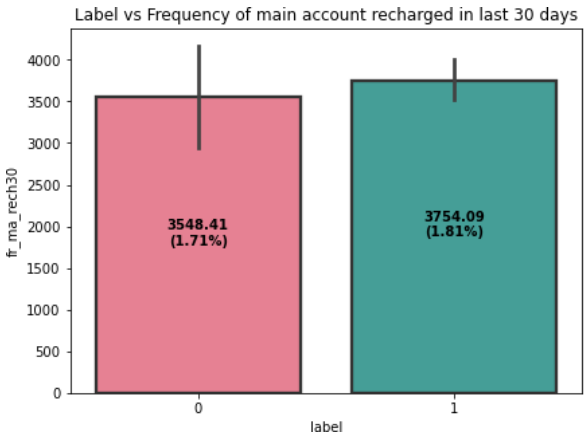
- ✓ The users who have recharged their main account on time are most likely to pay back their loan amount within 5 days. Also, some of the users who have not paid back their loan within 5 days they also recharged their main account on time.
- ✓ Looking at above plot of last_rech_amt_ma, we can say that if the amount of last recharge of main account is around 2000 then a greater number of people will pay back the loan amount.

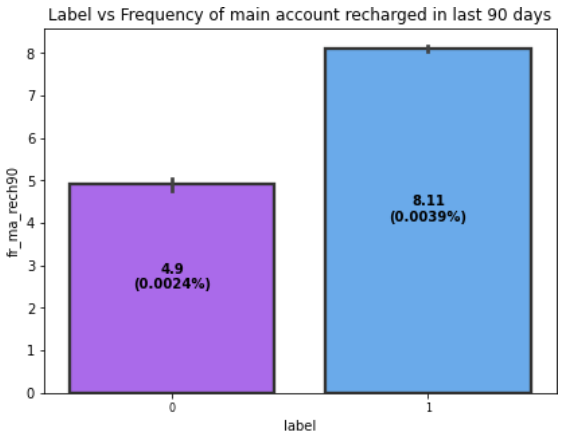




Observations:

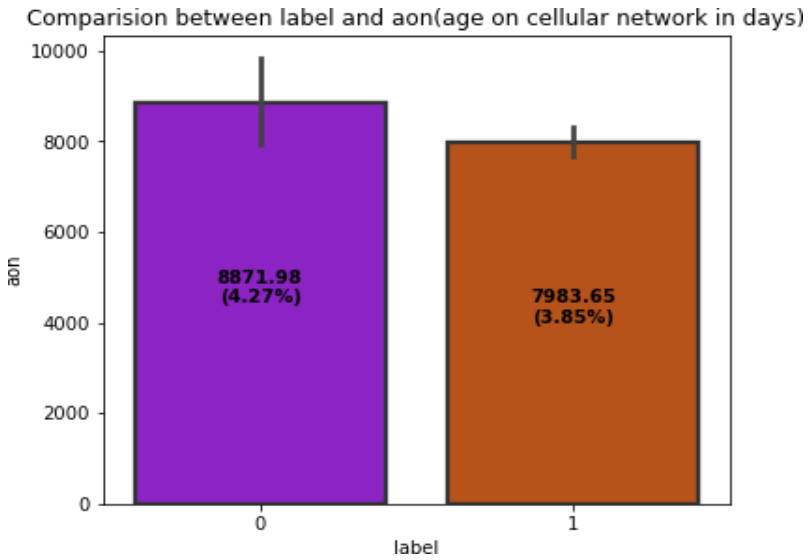
- ✓ The non-defaulters got recharged their main account more than 4 times in last 30 days and defaulters used to recharge their main account 1 time.
- ✓ The users who have paid back their loan within 5 days have got recharged their main account up to 7 times in last 90 days and the users who have not been paid loan within due date, they have got recharged their main account twice in last 90 days.
- ✓ From both the plots we can say that the users who got recharged their main account maximum times, they are able to pay back their loan amount within 5 days compared to the users who got their main account recharged less than 2 times.





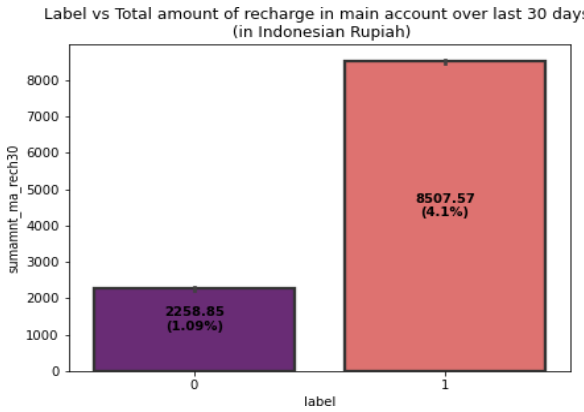
Observations:

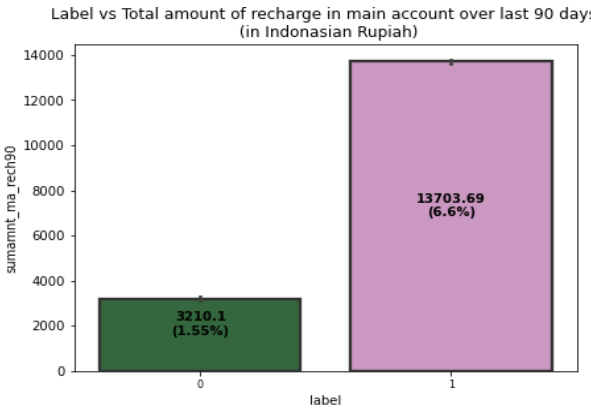
- ✓ The count of defaulters and non-defaulters is almost similar for the frequency of main account recharged in last 30 days. They didn't pay back the loan within 5 days. Which means there it is not contributing more for prediction.
- ✓ The frequency of main account recharged in last 90 days is increased for non-defaulters compared to defaulters.
- ✓ From the frequency of main account recharged in last 30 days & 90 days we have seen the users with low frequency are causing huge losses, company should implement some kind of strategies to reduce that like send SMS alerts for notification.



Observations:

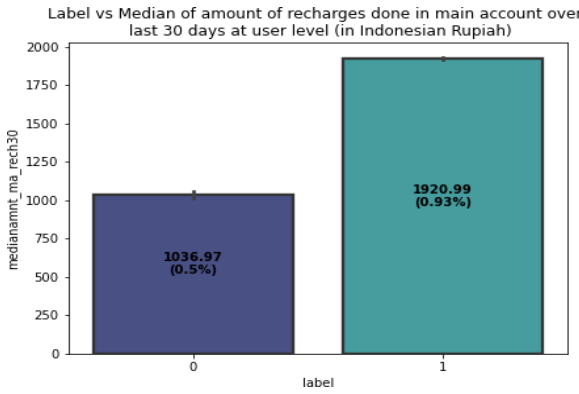
From the above bar plot we can observe that the defaulter rate is higher where the user age on cellular network in days is high which has around 8871 counts (in days).

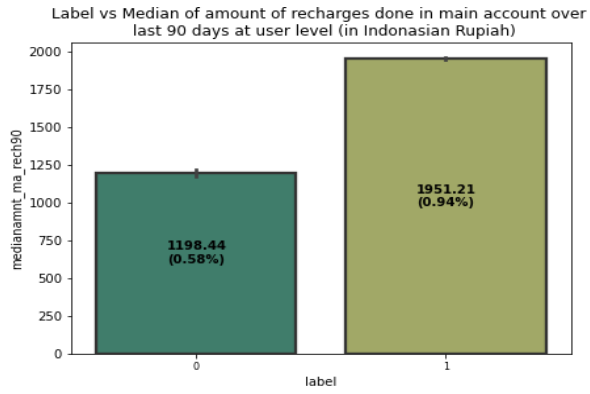




Observations:

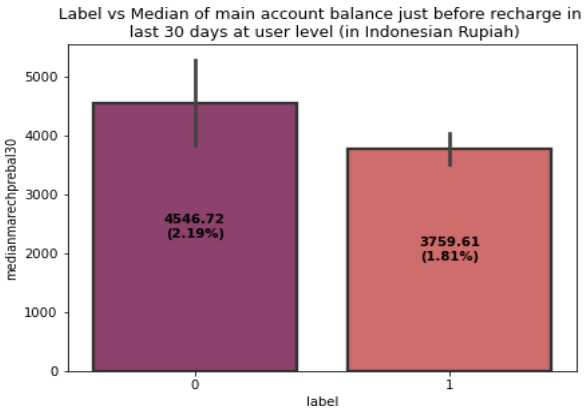
- ✓ The users who failed to pay back the loan within 5 days have less amount of recharge in their main account over last 30 days which is around 2000-2400 (in Indonesian Rupiah). And the users who paid back their loan within 5 days, they are recharging their main account more than 8000 (in Indonesian Rupiah) in last 30 days.
- ✓ The users who have paid their loan amount within 5 days have the total amount of recharge in their main account around 13700 (Indonesian Rupiah) in last 90 days while the defaulters have their total amount of recharge around 3200 (Indonesian Rupiah) over last 90 days.

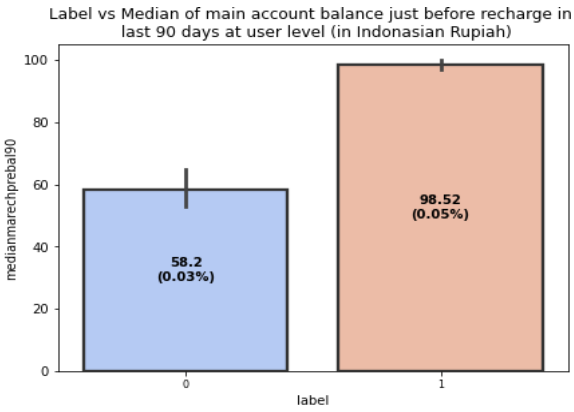




Observations:

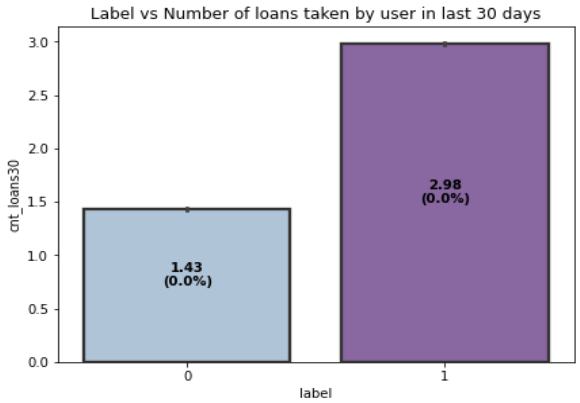
- ✓ The users who have done their median amount of recharge of 1920 (Indonesian Rupiah) in main account over last 30 days have successfully paid their credit amount within 5 days of issuing loan while the users who have done amount recharge of 1036 have failed to pay back the loan within due date.
- ✓ Similar to 30 days data, here also the users who have done their median amount recharge of 1950 in their main account over last 90 days they have paid back their credit amount within 5 days while the users having their median amount 1198 have not paid the loan within 5 days.

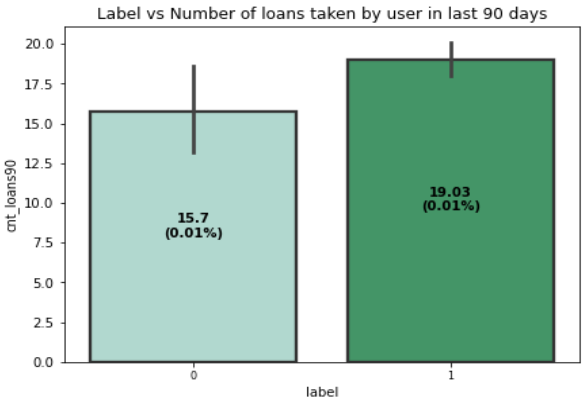




Observations:

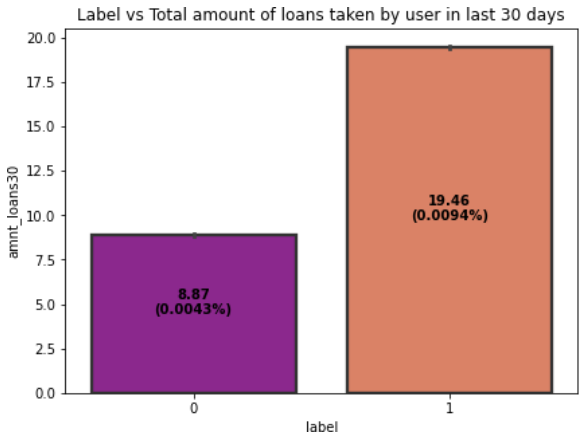
- ✓ In 30 days data, the median of main account balance for defaulters are around 4500 (Indonesian Rupiah) which is high compared to non-defaulters. Which means increasing median of main account balance just before recharge in last 30 days at user level, increasing the probability to being defaulter.
- ✓ In last 90 days data, the median of main account balance for non-defaulters are around 100 (Indonesian Rupiah) which is high compared to defaulters. Which means increasing median of main account balance just before recharge in last 90 days at user level, increasing the probability of being non-defaulters.

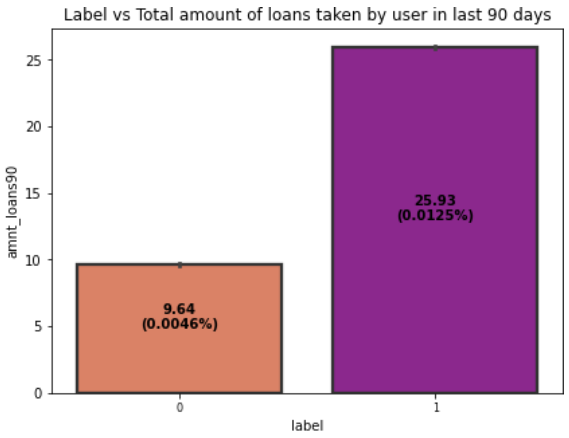




Observations:

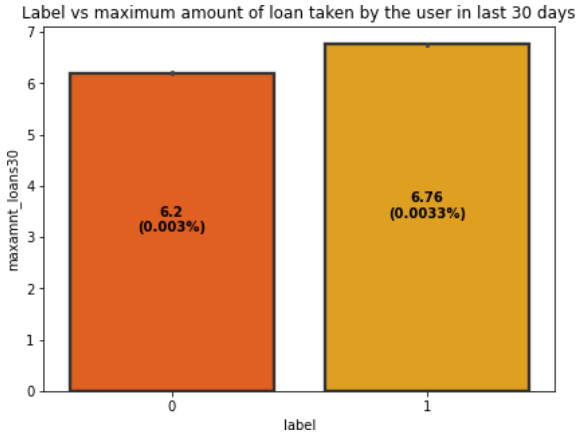
- ✓ Defaulters have taken 1 loan in last 30 days that is when a person takes loan amount for 1 time in last 30 days the chances of not paying back the credit amount are higher. And the users who have paid back the loan, they have taken maximum number of 3 loans in last 30 days data.
- ✓ In 90 days data, the number of loans taken by the defaulters are highly increasing also increasing the probability to being defaulter. Also, the number of loans taken by non-defaulters being decreased in last 90 days when compared to 30 days data.

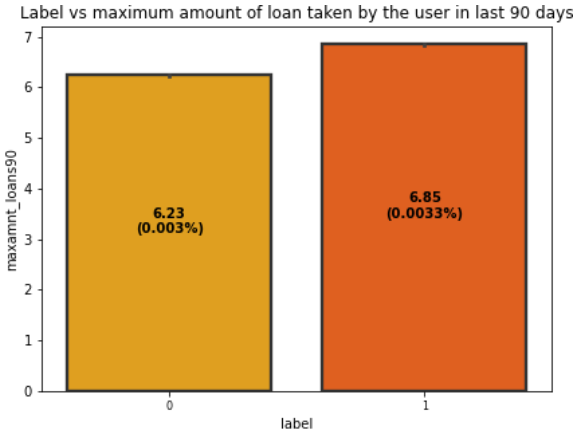




Observations:

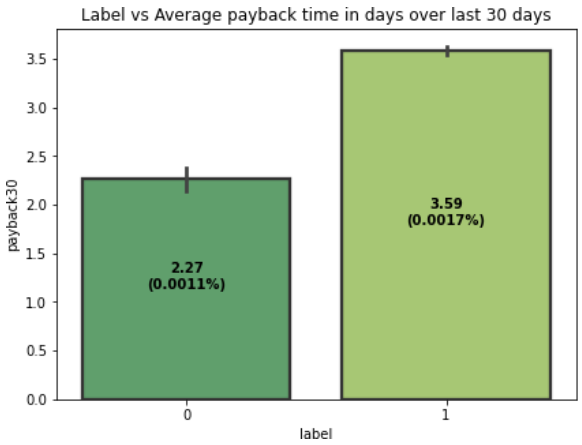
- ✓ The total amount of loans taken by the defaulters in last 30 days are in the range of 7.5-10 while the non-defaulters have taken up to 20 loans in last 30 days.
- ✓ The total amount of loans taken by the defaulters in last 90 days are up to 10 and the non- defaulters have taken total amount of loans up to 26 in last 90 days.
- ✓ So, from the above plot we can conclude that when the total number of loans taken by the users in last 90 days is below 10, then the chances of not paying back the loan amount are high.

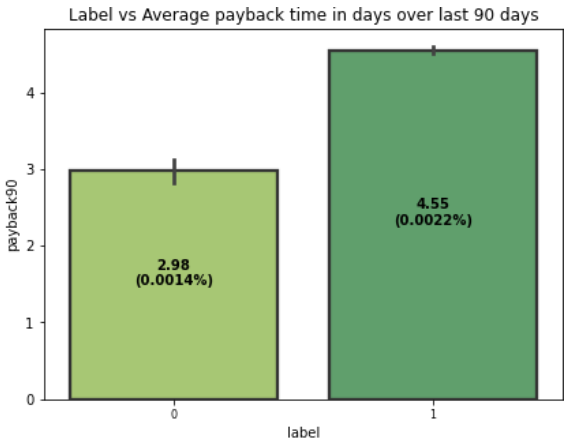




Observations:

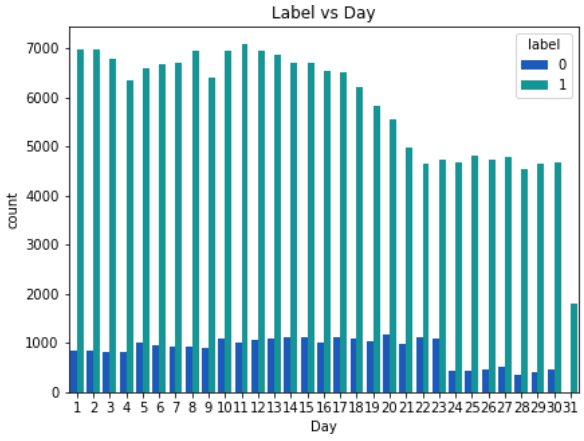
- ✓ The maximum amount of loan taken by the user in last 30 days and 90 days are almost same. The maximum amount of loan taken by the defaulters and non-defaulters are up to 6 and 7 respectively in last 30 and 90 days. So, from the plot we can say that whenever the user takes the maximum loan amount of 6, then only some users may not pay back the loan amount.

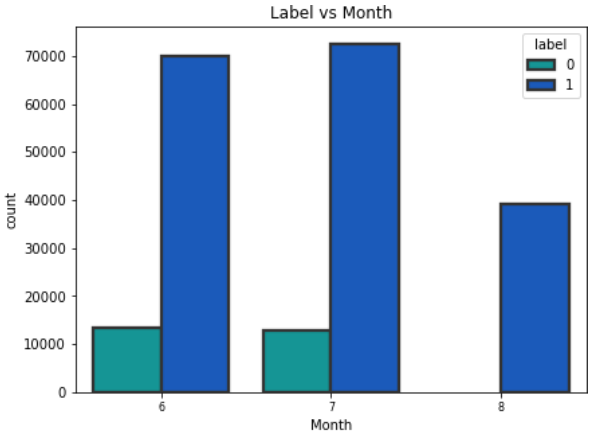




Observations:

- ✓ The defaulters are paying back their loan in an average of 2-2.5 days and the non-defaulters are paying back their loan in an average of 3 days over last 30 days.
- ✓ The defaulters in last 90 days, are paying back their loan in an average of 3 days and non-defaulters are paying back their loan in 4-5 days over last 90 days.
- ✓ It is seen from the plot that when an average payback time is below 3 days over last 30 & 90 days, then defaulters' rate is high.





Observations:

- ✓ The users who have taken loans in the month of august, they seem paying back their loan within 5 days.

Interpretation of the Results

Visualizations: I have used distribution plot to visualize the numerical variables. Used bar plots to check the relation between label and the features. The heat map and bar plot helped me to understand the correlation between dependent and independent features. Also, heat map helped to detect the multicollinearity problem and feature importance. Detected outliers and skewness with the help of box plots and distribution plots respectively. And I found some of the features skewed to right. I got to know the count of each column using bar plots.

Pre-processing: The dataset should be cleaned and scaled to build the ML models to get good predictions. I have performed few processing steps which I have already mentioned in the pre-processing steps where all the important features are present in the dataset and ready for model building.

Model building: After cleaning and processing data, I performed train test split to build the model. I have built multiple classification models to get the accurate accuracy score, and evaluation metrics like precision, recall, confusion matrix, f1 score. I got Gradient Boosting Classifier as best model which gives 90% accuracy score. I checked the cross-validation score ensuring there will be no overfitting. After tuning the best model Gradient Boosting Classifier, I got 95% accuracy score and also got increment in AUC-ROC curve. Finally, I saved my final model and got the good predictions results for defaulters.

4. CONCLUSION

Key Findings and Conclusions of the Study

This case study aims to give an idea of applying EDA in a real business scenario. In this case study, apart from applying the techniques that we have learnt in the EDA module, we will also develop a basic understanding of risk analytics in banking and financial services and understand how data is used to minimise the risk of losing money while lending to customers. From this dataset we were able to understand that the selection of customers for the credit to know whether they are defaulters or non-defaulters are done on the basis of different features.

In this study, we have used multiple machine learning models to predict the micro credit defaulters' rate. We have gone through the data analysis by performing feature engineering, finding the relation between features and label through visualizations. And got the important feature and we used these features to predict the defaulters' rate by building ML models. After training the model we checked CV score to overcome with the overfitting issue. Performed hyper parameter tuning, on the best model and the best model accuracy increased by 5% and the accuracy score was 95%. We have also got good prediction results.

Findings: From the whole study we found that the MFIs have provided loan to the user who have no recharge or balance in their account which needs to be stopped. Also, the frequency of main account recharged in last 30 days & 90 days we have seen the users with low frequency are causing huge losses, company should implement some kind of strategies to reduce like sending SMS alerts for notification. We found the defaulting rate is higher in old customers list. We found outliers and removed them and couldn't remove all the outliers since the data is expensive so, proceeded the data with remaining outliers. Further, removed skewness. Looking at the heat map, I could see there were few features which were correlated with each other, yet I haven't removed them based on their correlation thinking multicollinearity will not affect prediction. Other insight from this study is the impact of SMOTE on the model performance as well as how the number of variables included in the models.

Learning Outcomes of the Study in respect of Data Science

While working on this project I learned many things about the micro credit loan banks and organizations and how the machine learning models have helped to predict the defaulters' rate which provides greater understanding into the many causes of loan defaults in Microfinance Banks. I found that the project was quite interesting as the dataset contains several types of data. I used several types of plotting to visualize the relation between target and features. This graphical representation helped me to understand which features are important and how these features describe defaulter and non-defaulter rate in the banks. Data cleaning was one of the important and crucial things in this project where I dealt with features having zero values, negative statistical summary and time variables.

One of the challenges I faced while data cleaning is outlier removal, in most of the scenario's Z-score will be used as outlier removal technique since it performs quite well with less data loss. In our data set, Z-score has caused 18% data loss. Then I tried another technique called InterQuartileRange it caused around 80% data loss. So, I used percentile method to handle the outliers.

Finally, our aim is achieved by predicting the defaulters' rate at the organization that could help the clients in further investment and improvement in selection of customers.

Limitations of this work and scope for future work

Limitations: The dataset contains the data of only 2016 year belonging to telecom industry, if we get the data of other years along with other telecom companies then dataset would be quite more interesting to handle and predict on varied scenario.

In the dataset our data is not properly distributed in some of the columns many of the values in the columns are 0's and negative values which are not realistic. Because I have seen in some of the columns even the person didn't take loan but the label says that he paid back the loan amount, which is not correct. So, because of that data our models may not make the right patterns and the performance of the model also reduces. So that issues need to be taken care. Due to the presence of huge outliers, we unsure that our model is going to perform well on the dataset. Due to the class imbalance we had to balance the class defaulter (0). This might also have some effect on model.

Future work: The potential future work for this project will be a further development of the model by deepening analysis on variables used in the models as well as creating new variables in order to make better predictions.

As a recommendation, the Microfinance Institutions (MFI's) should adopt the group loan policy as the main mode through which microcredit may be issued to suitable applicants. Regular Credit risk assessment and analysis should be undertaken by the MFIs. Microfinance Bank officials should personally visit the group clients, either in their shops or houses to ascertain that the group is genuinely formed and that all members are serious business people and not just members by name. This will also avoid customers giving fake addresses with the intention to run away with the Bank money.