Micro Credit Defaulter Project Report

Submitted by:

Shikha Chaudhary

ACKNOWLEDGMENT

I'd like to express my heartfelt gratitude to my SME (Subject Matter Expert) Shubham Yadav, as well as Flip Robo Technologies, for allowing me to work on this project on Micro Credit Defaulter Project, as well as for assisting me in conducting extensive research that allowed me to learn a lot of new things.

In addition, I used a few outside resources to help me finish the project. I made sure to learn from the samples and adjust things to fit my project's needs. The following are all of the external resources that were utilised to create this project:

1) https://www.google.com/
2) https://www.youtube.com/
3) https://scikit-learn.org/stable/user_guide.html
4) https://github.com/
5) https://www.kaggle.com/
6) https://medium.com/
7) https://towardsdatascience.com/
8) https://www.analyticsvidhya.com/

INTRODUCTION

- **Business Problem Framing**

  Microfinance Institutions (MFIs) are businesses that provide financial services to low-income people. When addressing unbanked poor families living in rural places with few sources of income, MFS becomes quite effective. Group Loans, Agricultural Loans, Individual Business Loans, and other Microfinance Services (MFS) are some of the Microfinance Services (MFS) provided by MFI. Many microfinance institutions (MFI), experts, and donors promote the use of mobile financial services (MFS), which they believe are more convenient, efficient, and cost-effective than the traditional high-touch strategy used to deliver microfinance services for a long time. Despite the fact that the MFI industry focuses primarily on low-income households and is extremely beneficial in these areas, MFS implementation has been unequal, with both considerable obstacles and triumphs.

- **Conceptual Background of the Domain Problem**

  Microfinance is now widely recognised as a strategy for poverty reduction, with $70 billion in outstanding loans and a global client base of 200 million people. We are now working with a client in the telecom industry. They are a provider of fixed wireless telecommunications networks. They've released a number of products and built their business and organisation around the budget operator

model, which entails providing better products at lower prices to all value-conscious clients via a disruptive innovation strategy that focuses on the subscriber. They recognise the value of communication and how it influences a person's life, thus they focus on giving low-income families and impoverished consumers with services and products that can assist them in their time of need. They've teamed up with a microfinance institution to offer micro-credit on mobile balances that must be paid back in five days. If the Consumer deviates from the course of repaying the loaned amount within the time period of 5 days, he is considered a defaulter. The payback amount for a loan of 5 (in Indonesian Rupiah) should be 6 (in Indonesian Rupiah), whereas the payback amount for a loan of 10 (in Indonesian Rupiah) should be 12. (in Indonesian Rupiah).

- **Review of Literature**

1. What is Microfinance?

Microfinance is frequently thought of as financial services for the poor and low-income. In practise, the term "microfinance institution" is generally used more strictly to refer to loans and other services provided by providers that identify themselves as such (MFIs). Microfinance can also be defined as a collection of different operators working together to meet the needs of the financially underserved in terms of poverty alleviation, social promotion, emancipation, and inclusion. Microfinance organisations use cutting-edge methods to reach and serve their target customers. Microfinance operations are fundamentally different from traditional financial disciplines such as general and entrepreneurial finance. This disparity can be explained by the fact that microcredit loans are often too little to fund growth-oriented business

projects. The following are some of the unique characteristics of microfinance:

i. Delivery of very small loans to unsalaried workers.

ii. Little or no collateral requirements.

iii. Group lending and liability.

iv. Pre-loan savings requirement.

v. Gradually increasing loan sizes.

If current loans are repaid in whole and on time, there is an implicit assurance of easy access to future loans. Microfinance is considered as a catalyst for poverty alleviation, especially in developing nations, when it is delivered in innovative and sustainable methods to help the underserved poor.

2. Default in Microfinance

In microfinance, default refers to a client's failure to repay a loan. The default could be in terms of the payment amount or the payment schedule.

- **Motivation for the Problem Undertaken**

The major goal of this research is to create a model that can predict whether or not consumers will pay their loans on time. Machine Learning methods will be used to predict.

We obtained the sample data from our client database. In order to improve the credit selection process, the client requests certain forecasts that will assist them in making future investments and improving customer selection.

Analytical Problem Framing

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

  On the basis of accounts that have been recharged in the recent 30 days, I have performed several analytics before moving further with exploratory study. I established a criteria that if a person does not recharge their main account within three months, I just delete their information because they are not important and may be old clients, but there is no revenue rotation. Then I investigated the date fields and discovered that the data is from 2016. I extracted the month and day from the date, separated the data into columns, then attempted to visualise the data using months and days.

  I looked at the maximum amount of loan taken by the folks and discovered that there were more outliers in the data. According to the client's description, the loan amount that the customer can pay is either rupiah 6 or 12, so I've

removed all the loan amounts that suggest the loan was accepted for more than 12 rupiah.

Then I segregated the defaulters' data and verified the network's valuable clients, discovering that their monthly revenue exceeds 10,000 rupiah. We deleted some columns despite the fact that the data is extremely unbalanced and many columns do not have the expected maximum value. We checked for skewed data and attempted to treat it before running the model, which resulted in NaN.

When we tried to remove the undesired data, i.e. the outliers, we discovered that about 40000 records were chopped. Despite the fact that the data provided by the client included about 37 columns and over 2 lakh columns, I did not want to risk losing any valuable data, therefore I skipped the outlier reduction step as well. After scaling my data, I ran it through a number of classification models and discovered that the Extra Trees Classifier Algorithm performed admirably.

- **Data Sources and their formats**

  One of our telecom industry clients gave the information. They are a fixed wireless telecommunications network provider that has launched a variety of products and built a business and organisation around the budget operator model, providing better products at lower prices to all value-conscious customers through a disruptive innovation strategy that focuses on the subscriber.

  The data was provided by an Indonesian telecom firm in the form of a CSV file with an excel sheet describing the data. They also gave a problem statement, outlining what they expect of us as well as the criteria that must be met.

Let's take a look at the facts right now. I've attached a screenshot to give you an idea of what I'm talking about.

```
import warnings
warnings.simplefilter("ignore")
warnings.filterwarnings("ignore")
import joblib

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import missingno
import pandas_profiling
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
import xgboost as xgb
import lightgbm as lgb

from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV
from scikitplot.metrics import plot_roc_curve
from sklearn.metrics import roc_curve, auc, roc_auc_score
```

```
df = pd.read_csv("Data_File.csv")
```

I am importing the dataset comma separated values file and storing it into our dataframe for further usage.

```
df # checking the first 5 and last 5 rows
```

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0
...	...	...	...	...	...	...	...	...	...	...	...	...
209588	209589	1	22758I85348	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	...	6.0
209589	209590	1	95583I84455	1075.0	36.936000	36.936000	1728.36	1728.36	4.0	0.0	...	6.0
209590	209591	1	28556I85350	1013.0	11843.111667	11904.350000	5861.83	8893.20	3.0	0.0	...	12.0
209591	209592	1	59712I82733	1732.0	12488.228333	12574.370000	411.83	984.58	2.0	38.0	...	12.0
209592	209593	1	65061I85339	1581.0	4489.362000	4534.820000	483.92	631.20	13.0	0.0	...	12.0

209593 rows × 37 columns

We're looking at the first five and last five rows of our dataset. It reveals that our data frame contains a total of 209593 rows and 37 columns. This is a Classification challenge because we have the label column that stores the defaulter and non-defaulter values labelled with 0 and 1.

- **Data Preprocessing Done**

  Checked for missing values to confirm the information of no null values present provided in the problem statement.

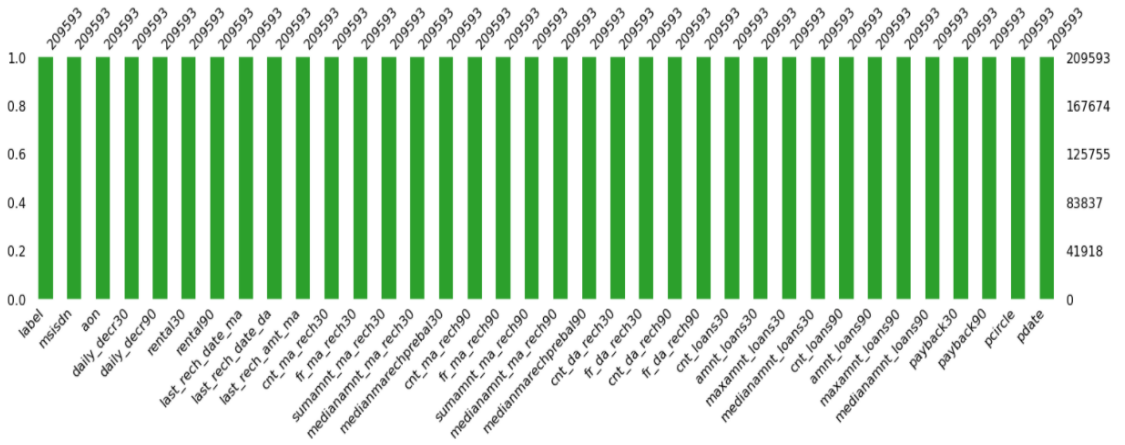
```
df.isna().sum() # checking for missing values
```

```
label                       0
msisdn                      0
aon                         0
daily_decr30                0
daily_decr90                0
rental30                    0
rental90                    0
last_rech_date_ma           0
last_rech_date_da           0
last_rech_amt_ma            0
cnt_ma_rech30               0
fr_ma_rech30                0
sumamnt_ma_rech30           0
medianamnt_ma_rech30        0
medianmarechprebal30        0
cnt_ma_rech90               0
fr_ma_rech90                0
sumamnt_ma_rech90           0
medianamnt_ma_rech90        0
medianmarechprebal90        0
cnt_da_rech30               0
fr_da_rech30                0
cnt_da_rech90               0
fr_da_rech90                0
cnt_loans30                 0
amnt_loans30                0
maxamnt_loans30             0
medianamnt_loans30          0
cnt_loans90                 0
amnt_loans90                0
maxamnt_loans90             0
medianamnt_loans90          0
payback30                   0
payback90                   0
pcircle                     0
pdate                       0
dtype: int64
```

Took a visual on the missing data information as well.

```
missingno.bar(df, figsize = (25,5), color="tab:green")
```

```
<AxesSubplot:>
```



We may confirm the non-null count details as well as the datatype information using the info method. We have 21 columns with float/decimal datatypes, 12 columns with integer datatypes, and three columns with object/categorical datatypes. Before we can use the information in our machine learning models, we'll need to convert the object datatype columns to numerical data.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 209592 entries, 0 to 209592
Data columns (total 36 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   label                   209592 non-null  int64
 1   msisdn                  209592 non-null  object
 2   aon                     209592 non-null  float64
 3   daily_decr30            209592 non-null  float64
 4   daily_decr90            209592 non-null  float64
 5   rental30                209592 non-null  float64
 6   rental90                209592 non-null  float64
 7   last_rech_date_ma       209592 non-null  float64
 8   last_rech_date_da       209592 non-null  float64
 9   last_rech_amt_ma        209592 non-null  int64
 10  cnt_ma_rech30           209592 non-null  int64
 11  fr_ma_rech30            209592 non-null  float64
 12  sumamnt_ma_rech30       209592 non-null  float64
 13  medianamnt_ma_rech30    209592 non-null  float64
 14  medianmarechprebal30    209592 non-null  float64
 15  cnt_ma_rech90           209592 non-null  int64
 16  fr_ma_rech90            209592 non-null  int64
 17  sumamnt_ma_rech90       209592 non-null  int64
 18  medianamnt_ma_rech90    209592 non-null  float64
 19  medianmarechprebal90    209592 non-null  float64
 20  cnt_da_rech30           209592 non-null  float64
 21  fr_da_rech30            209592 non-null  float64
 22  cnt_da_rech90           209592 non-null  int64
 23  fr_da_rech90            209592 non-null  int64
 24  cnt_loans30             209592 non-null  int64
 25  amnt_loans30            209592 non-null  int64
 26  maxamnt_loans30         209592 non-null  float64
 27  medianamnt_loans30      209592 non-null  float64
 28  cnt_loans90             209592 non-null  float64
 29  amnt_loans90            209592 non-null  int64
 30  maxamnt_loans90         209592 non-null  int64
 31  medianamnt_loans90      209592 non-null  float64
 32  payback30               209592 non-null  float64
 33  payback90               209592 non-null  float64
 34  pcircle                 209592 non-null  object
 35  pdate                   209592 non-null  object
dtypes: float64(21), int64(12), object(3)
```

- **Data Inputs- Logic- Output Relationships**

  Data description on each column present in our dataset.

  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 Indonasian Rupiah)

medianamnt_ma_rech90 : Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)

medianmarechprebal90 : Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)

cnt_da_rech30 : Number of times data account got recharged in last 30 days

fr_da_rech30 : Frequency of data account recharged in last 30 days

cnt_da_rech90 : Number of times data account got recharged in last 90 days

fr_da_rech90 : Frequency of data account recharged in last 90 days

cnt_loans30 : Number of loans taken by user in last 30 days

amnt_loans30 : Total amount of loans taken by user in last 30 days

maxamnt_loans30 : Maximum amount of loan taken by the user in last 30 days

medianamnt_loans30: Median of amounts of loan taken by the user in last 30 days

cnt_loans90 : Number of loans taken by user in last 90 days

amnt_loans90 : Total amount of loans taken by user in last 90 days

maxamnt_loans90 : Maximum amount of loan taken by the user in last 90 days

medianamnt_loans90: Median of amounts of loan taken by the user in last 90 days

payback30 : Average payback time in days over last 30 days

payback90 : Average payback time in days over last 90 days

pcircle : Telecom circle

pdate : Date

Data description in a tabular format:

Column Names	Column 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 user
aon	Age on cellular network in days
daily_decr30	Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)
daily_decr90	Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)
rental30	Average main account balance over last 30 days
rental90	Average main account balance over last 90 days
last_rech_date_ma	Number of days till last recharge of main account
last_rech_date_da	Number of days till last recharge of data account
last_rech_amt_ma	Amount of last recharge of main account (in Indonesian Rupiah)
cnt_ma_rech30	Number of times main account got recharged in last 30 days
fr_ma_rech30	Frequency of main account recharged in last 30 days
sumamnt_ma_rech30	Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)
medianamnt_ma_rech30	Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
medianmarechprebal30	Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
cnt_ma_rech90	Number of times main account got recharged in last 90 days
fr_ma_rech90	Frequency of main account recharged in last 90 days
sumamnt_ma_rech90	Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)
medianamnt_ma_rech90	Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)
medianmarechprebal90	Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)
cnt_da_rech30	Number of times data account got recharged in last 30 days
fr_da_rech30	Frequency of data account recharged in last 30 days
cnt_da_rech90	Number of times data account got recharged in last 90 days
fr_da_rech90	Frequency of data account recharged in last 90 days
cnt_loans30	Number of loans taken by user in last 30 days
amnt_loans30	Total amount of loans taken by user in last 30 days
maxamnt_loans30	Maximum amount of loan taken by the user in last 30 days
medianamnt_loans30	Median of amounts of loan taken by the user in last 30 days
cnt_loans90	Number of loans taken by user in last 90 days
amnt_loans90	Total amount of loans taken by user in last 90 days
maxamnt_loans90	Maximum amount of loan taken by the user in last 90 days
medianamnt_loans90	Median of amounts of loan taken by the user in last 90 days
payback30	Average payback time in days over last 30 days
payback90	Average payback time in days over last 90 days
pcircle	Telecom circle
pdate	Date

- **State the set of assumptions (if any) related to the problem under consideration**

  I had assumed that any telecom operator would keep customer data for three months, therefore I had cut off my data based on that assumption.

  Because the data is from 2016, just the date and months are different, I removed the 2016 year from the pdate columns. Months and days were divided into separate columns.

  Then I looked at the data of defaulters separately and discovered that many valuable consumers are defaulters

because they may have forgotten to pay or are too preoccupied with their lives. I separated them so that the company could deal with them courteously, as we cannot afford to lose these consumers.

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

  Hardware technology being used.

  RAM : 8 GB

  CPU : Intel(R) Core(TM) i3-7100U CPU @ 2.40GHz 2.40 GHz

  Software technology being used.

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

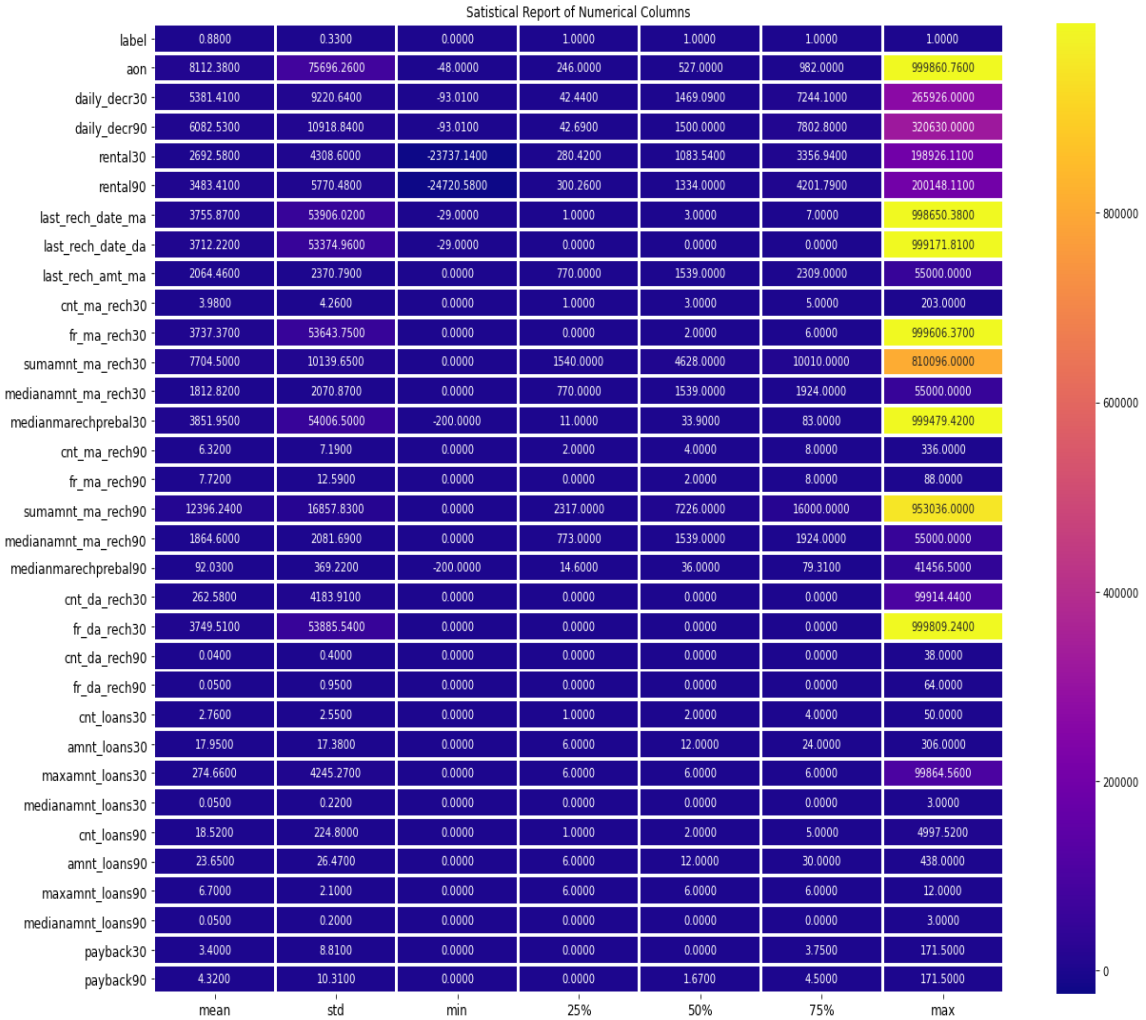
  Libraries/Packages specifically being used.

  Pandas , NumPy, matplotlib, seaborn, scikit-learn, pandas-profiling, missingno

Model/s Development and Evaluation

- **Identification of possible problem-solving approaches (methods)**

  To check the numerical data specifics, we used the describe technique. There are 33 numerical values in the columns, and it appears that the count, mean, standard deviation, minimum value, 25% quartile, 50% quartile, 75% quartile, and maximum value are all mostly properly distributed in terms of data points, but I do see some abnormality that we will confirm with a visual.



In the above report we can see that the maximum value for columns aon, daily_decr30, daily_decr90, rental30, rental90, last_rech_date_ma, last_rech_date_da, fr_ma_rech30, sumamnt_ma_rech30, medianmarechprebal30, sumamnt_ma_rech90 and fr_da_rech30 have quite a high number than the other column values.

- **Testing of Identified Approaches (Algorithms)**

  Listing down all the 8 classification machine learning algorithms used for the training and testing.

```
LR = LogisticRegression()
ETC = ExtraTreesClassifier()
SVCM = SVC(C=1.0, kernel='rbf', gamma='auto', random_state=42)
DTC = DecisionTreeClassifier(max_depth=15, random_state=21)
RFC = RandomForestClassifier(max_depth=15, random_state=111)
KNN = KNeighborsClassifier(n_neighbors=15)
XGB = xgb.XGBClassifier(verbosity=0)
LGBM = lgb.LGBMClassifier()

models = {'Logistic Regression' : LR,
          'Extra Trees Classifier' : ETC,
          'Support Vector Classifier' : SVCM,
          'Decision Tree Classifier' : DTC,
          'Random Forest Classfier' : RFC,
          'K Nearest Neighbors Classifier' : KNN,
          'XGB Classifier' : XGB,
          'LGBM Classifier' : LGBM}
```

- **Run and evaluate selected models**

  I created a Classification Model function incorporating the evaluation metrics so that we can get the required data for all the models.

Machine Learning Model for Classification with Evaluation Metrics

```
# Classification Model Function

def classify(model_func):

    for model_name, model in model_func.items():

        # Training the model
        model.fit(X_train, Y_train)

        # Predicting Y_test
        pred = model.predict(X_test)

        print('\n##############################',model_name,'##############################')

        # Classification Report
        class_report = classification_report(Y_test, pred)
        print("\nClassification Report for {}:\n".format(model_name), class_report)

        # Accuracy Score
        acc_score = (accuracy_score(Y_test, pred))*100
        print("Accuracy Score for {}:".format(model_name), acc_score)

        # Cross Validation Score
        cv_score = (cross_val_score(model, X, Y, cv=5).mean())*100
        print("Cross Validation Score for {}:".format(model_name), cv_score)

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

- **Key Metrics for success in solving problem under consideration**

  The accuracy score, cross val score, classification report, auc score, and confusion matrix were the main metrics employed in this study. We used Hyperparameter Tuning to identify the optimal parameters and to improve our scores, and we'll be using the GridSearchCV method to do it.

  1. Cross Validation:

  Cross-validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross-validation the 1st part (20%) of the 5 parts will be kept out as a holdout

set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the
dataset. In the similar way further iterations are made for the second 20% of the dataset is held as a holdout set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality.

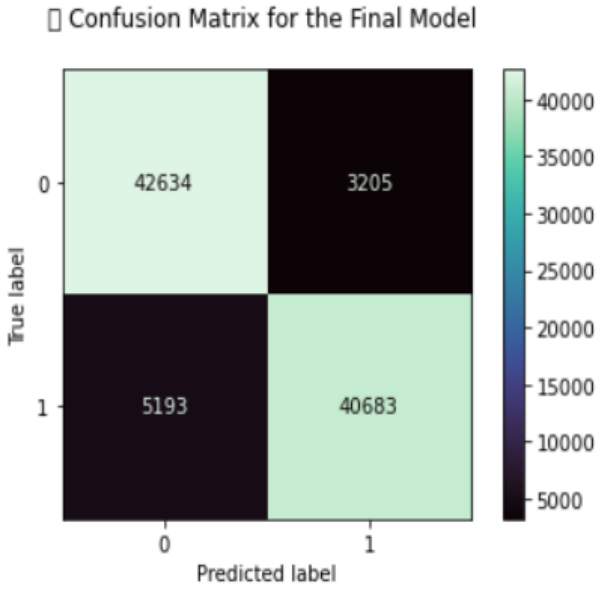
2. Confusion Matrix:

A confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e., commonly mislabelling one as another).

It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

Confusion Matrix

```
metrics.plot_confusion_matrix(Classifier, X_test, Y_test, cmap='mako')
plt.title('\t Confusion Matrix for the Final Model \n')
plt.show()
```



3. Classification Report:

The classification report visualizer displays the precision, recall, F1, and support scores for the model. There are four ways to check if the predictions are right or wrong: 1. TN / True Negative: the case was negative and predicted negative 2. TP / True Positive: the case was positive and predicted positive 3. FN / False Negative: the case was positive but predicted negative 4. FP / False Positive: the case was negative but predicted positive

Precision: Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive. It is the accuracy of positive predictions. The formula of precision is given below: Precision = TP/ (TP + FP)

Recall: Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. It is also the fraction of positives that were correctly identified. The formula of recall is given below: Recall = TP/(TP+FN)
F1 score: The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy. The formula is: F1 Score = 2*(Recall * Precision) / (Recall + Precision)
Support: Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process.

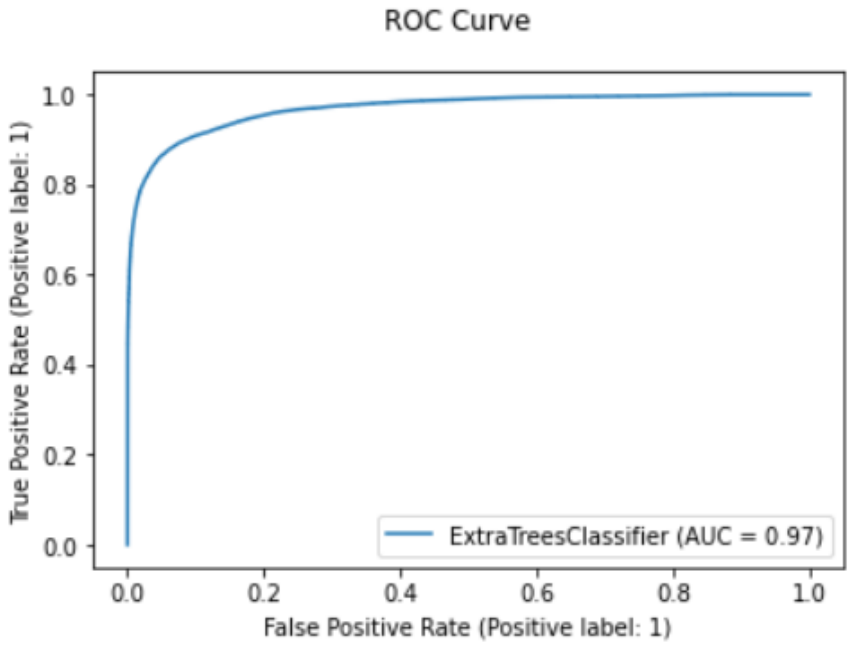
```
############################## Extra Trees Classifier ##############################

Classification Report for Extra Trees Classifier:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95     45930
           1       0.95      0.95      0.95     45785

    accuracy                           0.95     91715
   macro avg       0.95      0.95      0.95     91715
weighted avg       0.95      0.95      0.95     91715

Accuracy Score for Extra Trees Classifier: 95.1207545112577
Cross Validation Score for Extra Trees Classifier: 94.83481437060458

Accuracy Score - Cross Validation Score is 0.2859401406531248
```

4. AUC-ROC Curve and score:

AUC (Area Under the Curve) - ROC (Receiver Operating Characteristics) curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represent the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s. By analogy, the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.
The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis. Score is the area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

AUC ROC Curve

```
disp = metrics.plot_roc_curve(Final_Model, X_test, Y_test)
disp.figure_.suptitle("ROC Curve")
plt.show()
```



5. Hyperparameter Tuning:

There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as Hyperparameters. These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.

We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as Hyperparameter Tuning. We can do tuning by using GridSearchCV.
GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

Hyper parameter tuning on the best Classification ML Model

```
# Choosing Extra Trees Classifier

fmod_param = {'criterion' : ["gini", "entropy"],
              'max_depth' : [30, 40],
              'n_estimators' : [300, 350],
              'min_samples_split' : [3, 4],
              'random_state' : [42, 72]
             }

GSCV = GridSearchCV(ExtraTreesClassifier(), fmod_param, cv=5)
GSCV.fit(X_train,Y_train)
GSCV.best_params_
```

```
Final_Model = ExtraTreesClassifier(criterion="entropy", max_depth=30, min_samples_split=3,
                                   n_estimators=350, random_state=72)
Classifier = Final_Model.fit(X_train, Y_train)
fmod_pred = Final_Model.predict(X_test)
fmod_acc = (accuracy_score(Y_test, fmod_pred))*100
print("Accuracy score for the Best Model is:", fmod_acc)
```

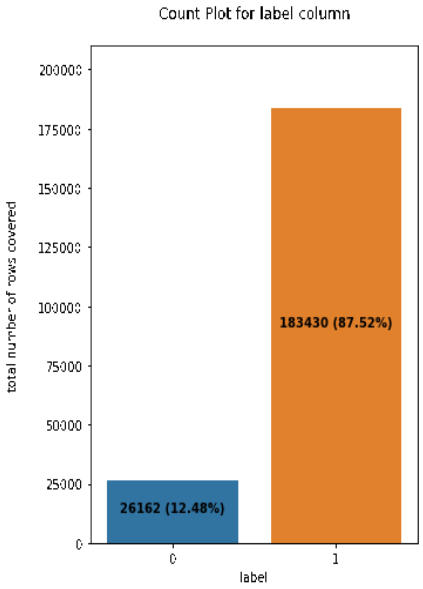
```
Accuracy score for the Best Model is: 90.8433734939759
```

- **Visualizations**

  Now we'll look at the various plots created with this dataset in order to gain a better understanding of the data. The plots' codes, as well as the results, are listed below:

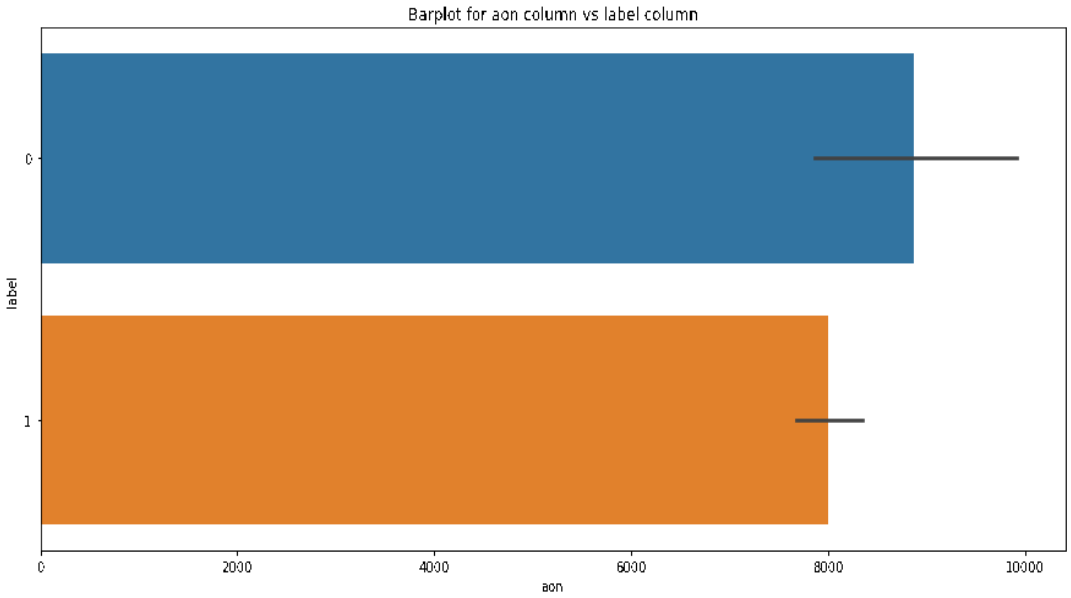
Univariate Analysis

```
try:
    x = 'label'
    k=0
    plt.figure(figsize=[5,7])
    axes = sns.countplot(df[x])
    for i in axes.patches:
        ht = i.get_height()
        mr = len(df[x])
        st = f"{ht} ({round(ht*100/mr,2)}%)"
        plt.text(k, ht/2, st, ha='center', fontweight='bold')
        k += 1
    plt.ylim(0,210000)
    plt.title(f'Count Plot for {x} column\n')
    plt.ylabel(f'total number of rows covered\n')
    plt.show()

except Exception as e:
    print("Error:", e)
    pass
```

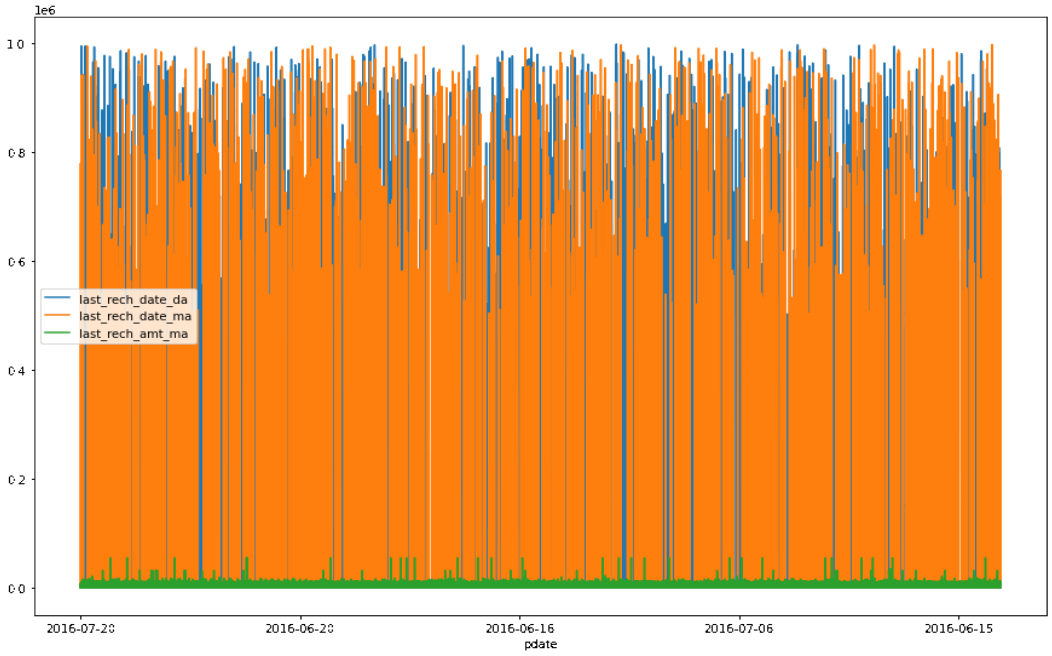


Bivariate Analysis

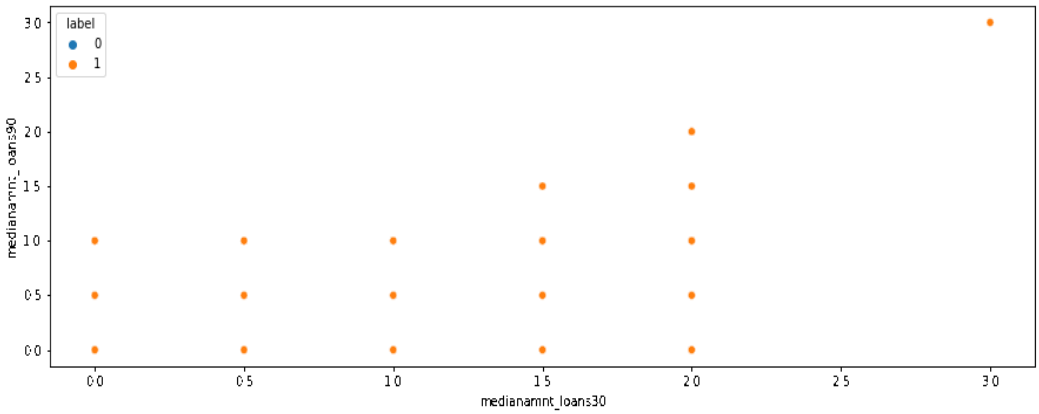
```
y = 'label'

x = 'aon'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df,orient='h')
plt.title(f"Barplot for {x} column vs {y} column")
plt.show()

x = 'last_rech_date_da'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df,orient='h')
plt.title(f"Barplot for {x} column vs {y} column")
plt.show()

x = 'last_rech_date_ma'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df,orient='h')
plt.title(f"Barplot for {x} column vs {y} column")
plt.show()

x = 'last_rech_amt_ma'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df,orient='h')
plt.title(f"Barplot for {x} column vs {y} column")
plt.show()
```




```
df.plot(kind="line", x="pdate", y=["last_rech_date_da", "last_rech_date_ma", "last_rech_amt_ma"], figsize=[15,10])
df.plot(kind="line", x="msisdn", y=["last_rech_date_da", "last_rech_date_ma", "last_rech_amt_ma"], figsize=[15,10])
```



```
plt.figure(figsize=(15,5))
sns.scatterplot(x='medianamnt_loans30', y='medianamnt_loans90', data=df, hue='label')

plt.figure(figsize=(15,5))
sns.scatterplot(x='maxamnt_loans30', y='maxamnt_loans90', data=df, hue='label')

plt.figure(figsize=(15,5))
sns.scatterplot(x='cnt_da_rech30', y='cnt_da_rech90', data=df, hue='label')

plt.figure(figsize=(15,5))
sns.scatterplot(x='cnt_loans30', y='cnt_loans90', data=df, hue='label')

plt.figure(figsize=(15,5))
sns.scatterplot(x='amnt_loans30', y='amnt_loans90', data=df, hue='label')

plt.figure(figsize=(15,5))
sns.scatterplot(x='cnt_ma_rech30', y='cnt_ma_rech90', data=df, hue='label')

plt.figure(figsize=(15,5))
sns.scatterplot(x='fr_da_rech30', y='fr_da_rech90', data=df, hue='label')

plt.figure(figsize=(15,5))
sns.scatterplot(x='fr_ma_rech30', y='fr_ma_rech90', data=df, hue='label')

plt.figure(figsize=(15,5))
sns.scatterplot(x='medianamnt_ma_rech30', y='medianamnt_ma_rech90', data=df, hue='label')

plt.figure(figsize=(15,5))
sns.scatterplot(x='daily_decr30', y='daily_decr90', data=df, hue='label')

plt.figure(figsize=(15,5))
sns.scatterplot(x='rental30', y='rental90', data=df, hue='label')

plt.figure(figsize=(15,5))
sns.scatterplot(x='payback30', y='payback90', data=df, hue='label')

plt.figure(figsize=(15,5))
sns.scatterplot(x='medianmarechprebal30', y='medianmarechprebal90', data=df, hue='label')

plt.figure(figsize=(15,5))
sns.scatterplot(x='sumamnt_ma_rech30', y='sumamnt_ma_rech90', data=df, hue='label')
```



- **Interpretation of the Results**

  for feature aon:

  Data ranges from -48 to 999860 with Mean value of 8112.34.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature daily_descr30:

Data ranges from -93 to 265926 with Mean value of 5381.4.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature daily_descr90:

Data ranges from -93 to 320630 with Mean value of 6082.52.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature rental30:

Data ranges from -23737.14 to 198926 with Mean value of 2692.58.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature rental90:

Data ranges from -24720 to 200148 with Mean value of 3483.41.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature last_rech_date_ma:

Data ranges from -29 to 998650 with Mean value of 3755.85.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature last_rech_date_da:

Data ranges from -29 to 999178 with Mean value of 3712.2.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature last_rech_amt_ma:

Data ranges from 0 to 55000 with Mean value of 2064.45.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature cnt_ma_rech30:

Data ranges from 0 to 203 with Mean value of 3.98.

Data is not distributed normally or in well curve.

Data is spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature fr_ma_rech30:

Data ranges from 0 to 999606 with Mean value of 3737.36.

Data is not distributed normally or in well curve.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature sumamnt_ma_rech30:

Data ranges from 0 to 810096 with Mean value of 7704.5.

Data is not distributed normally or in well curve.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature medianamnt_ma_rech30:

Data ranges from 0 to 55000 with Mean value of 1812.82.

Data is not distributed normally or in well curve.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature medianmarechprebal30:

Data ranges from -200 to 999479 with Mean value of 3851.93.

Data is not distributed normally or in well curve.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature cnt_ma_rech90:

Data ranges from 0 to 336 with Mean value of 6.32.

Data is not distributed normally or in well curve.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature fr_ma_rech90:

Data ranges from 0 to 88 with Mean value of 7.72.

Data is not distributed normally or in well curve.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature sumamnt_ma_rech90:

Data ranges from 0 to 953036 with Mean value of 12396.22.

Data is not distributed normally or in well curve.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature medianamnt_ma_rech90:

Data ranges from 0 to 55000 with Mean value of 1864.6.

Data is not distributed normally or in well curve.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature medianmarechprebal90:

Data ranges from -200 to 41456 with Mean value of 92.03.

Data is not distributed normally or in well curve.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature cnt_da_rech30:

Data ranges from 0 to 99914 with Mean value of 262.58.

Data is not distributed normally or in well curve.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature fr_da_rech30:

Data ranges from 0 to 999809 with Mean value of 3749.49.

Data is not distributed normally or in well curve.

Data is highly spreaded and needs to be treated accordingly.

Data is positively skewed and needs to be treated accordingly.

for feature cnt_da_rech90:

Data ranges from 0 to 38 with Mean value of 0.04.

Data is distributed normally but not in well curve.

Data is positively skewed and needs to be treated accordingly.

for feature fr_da_rech90:

Data ranges from 0 to 64 with Mean value of 0.05.

Data is not distributed normally or in well curve.

Data is positively skewed and needs to be treated accordingly.

for feature cnt_loans30:

Data ranges from 0 to 50 with Mean value of 2.76.

Data is not distributed normally or in well curve.

Data is positively skewed and needs to be treated accordingly.

for feature amnt_loans30:

Data ranges from 0 to 306 with Mean value of 17.95.

Data is not distributed normally or in well curve.

Data is positively skewed and needs to be treated accordingly.

for feature maxamnt_loans30:

Data ranges from 0 to 99864 with Mean value of 274.66.

Data is not distributed normally or in well curve.

Data is positively skewed and needs to be treated accordingly.

for feature medianamnt_loans30:

Data ranges from 0 to 3 with Mean value of 0.05.

Data is not distributed normally or in well curve and it is understandable as feature has only limited set of values.

Data is positively skewed and needs to be treated accordingly.

for feature cnt_loans90:

Data ranges from 0 to 4997.52 with Mean value of 18.52.

Data is not distributed normally or in well curve.

Data is positively skewed and needs to be treated accordingly.

for feature amnt_loans90:

Data ranges from 0 to 438 with Mean value of 23.65.

Data is not distributed normally or in well curve.

Data is positively skewed and needs to be treated accordingly.

for feature maxamnt_loans90:

Data ranges from 0 to 12 with Mean value of 6.7.

Data is not distributed normally or in well curve and it understandable as user has two option for loans i.e., 5 and 10 for with 6 and 12 has to be paid.

Data is positively skewed and needs to be treated accordingly.

for feature medianamnt_loans90:

Data ranges from 0 to 3 with Mean value of 0.05.

Data is not distributed normally or in well curve.

Data is positively skewed and needs to be treated accordingly.

for feature payback30:

Data ranges from 0 to 171.5 with Mean value of 3.4.

Data is not distributed normally or in well curve.

Data is positively skewed and needs to be treated accordingly.

for feature payback90:

Data ranges from 0 to 171.5 with Mean value of 4.32.

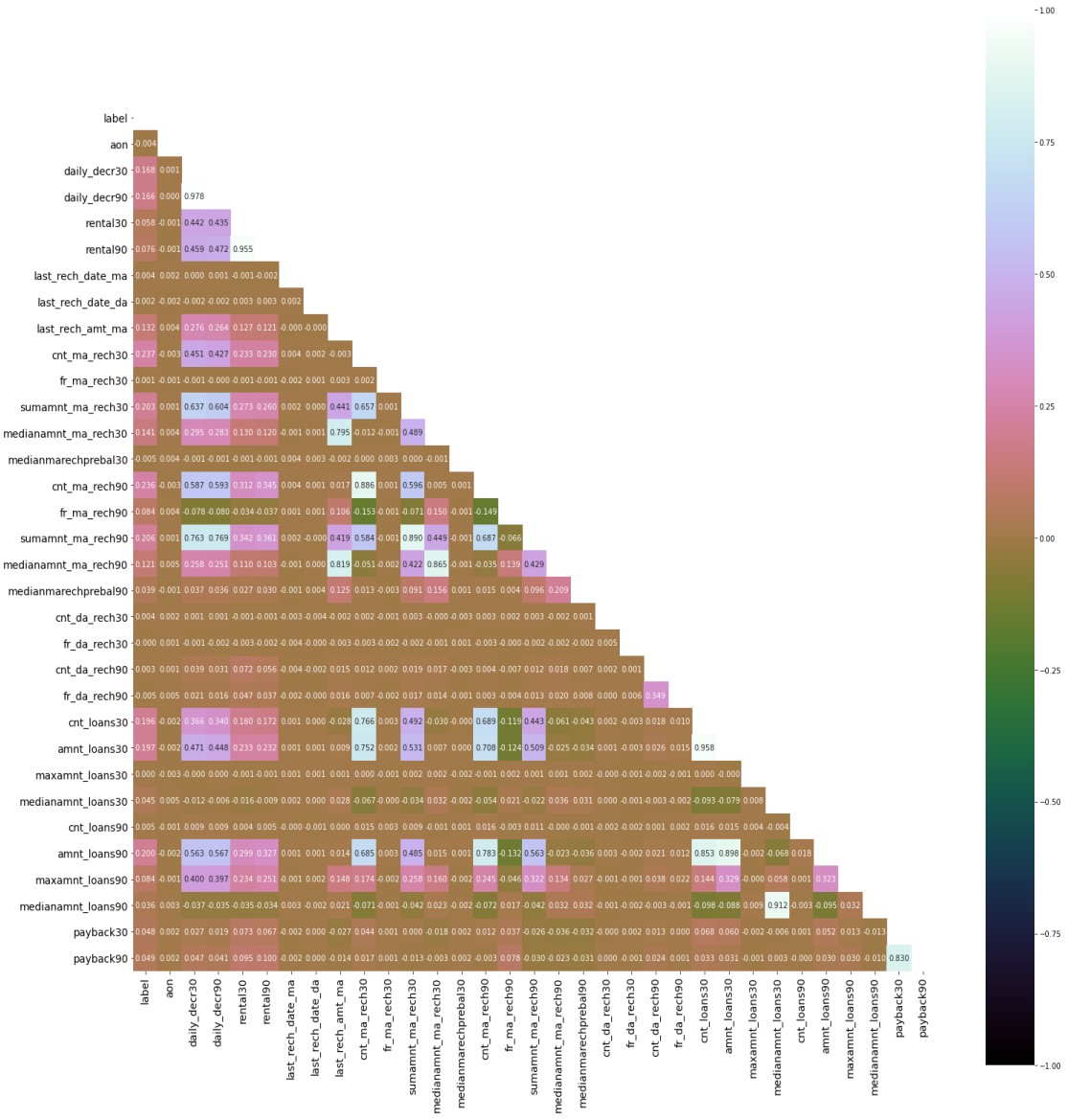
Data is not distributed normally or in well curve.

Data is positively skewed and needs to be treated accordingly.

Correlation using a Heatmap

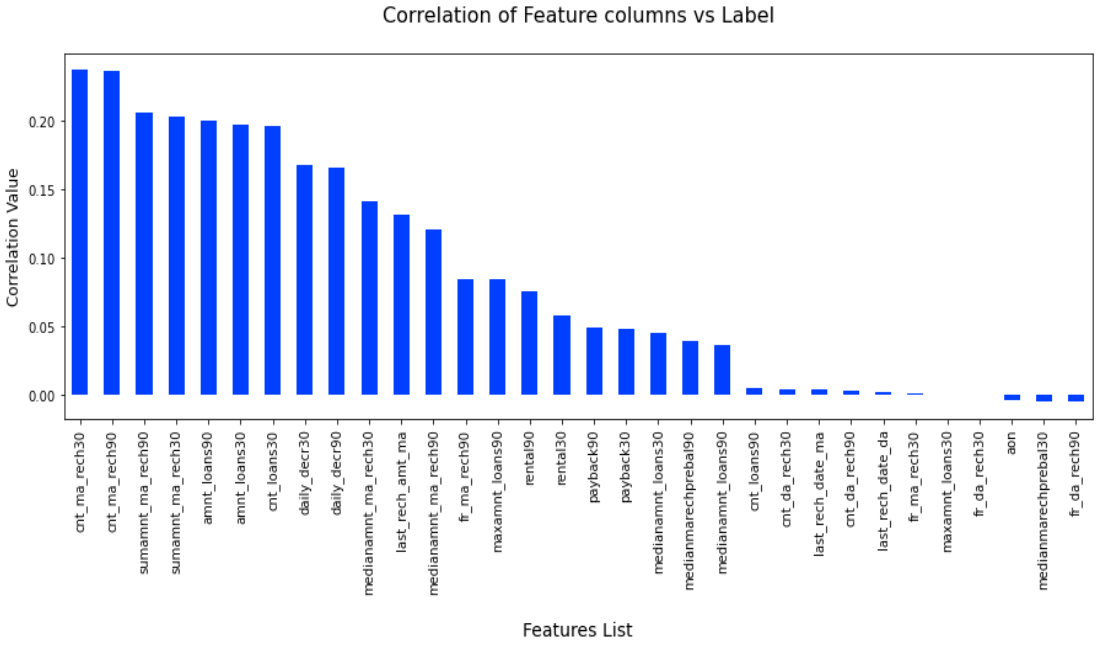
- Positive correlation - A correlation of +1 indicates a perfect positive correlation, meaning that both variables move in the same direction together.
- Negative correlation - A correlation of –1 indicates a perfect negative correlation, meaning that as one variable goes up, the other goes down.

```
upper_triangle = np.triu(df.corr())
plt.figure(figsize=(25,25))
sns.heatmap(df.corr(), vmin=-1, vmax=1, annot=True, square=True, fmt='0.3f',
            annot_kws={'size':10}, cmap="cubehelix", mask=upper_triangle)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```



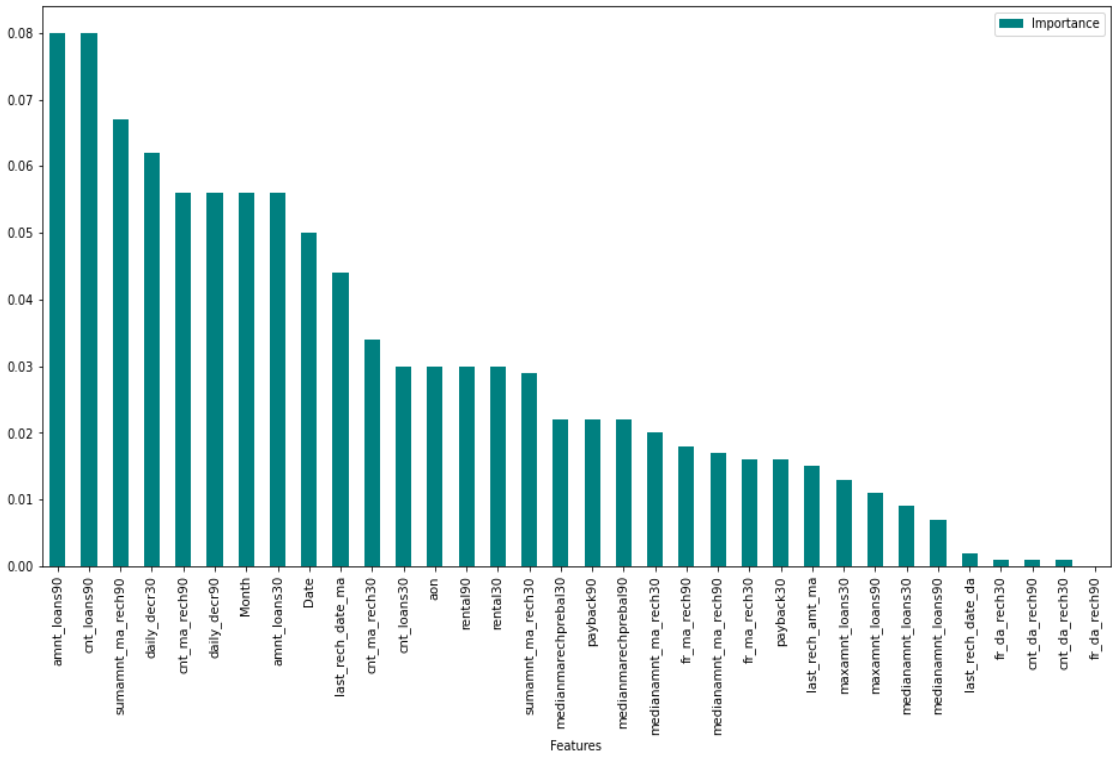
Correlation Bar Plot comparing Gender column with the remaining columns

```
df_corr = df.corr()
plt.figure(figsize=(15,5))
df_corr['label'].sort_values(ascending=False).drop('label').plot.bar()
plt.title("Correlation of Feature columns vs Label\n", fontsize=16)
plt.xlabel("\nFeatures List", fontsize=14)
plt.ylabel("Correlation Value", fontsize=12)
plt.show()
```



Feature importance bar graph

```
rf=RandomForestClassifier()
rf.fit(X_train, Y_train)
importances = pd.DataFrame({'Features':X.columns, 'Importance':np.round(rf.feature_importances_,3)})
importances = importances.sort_values('Importance', ascending=False).set_index('Features')
plt.rcParams["figure.figsize"] = (16,8)
importances.plot.bar(color='teal')
importances
```



CONCLUSION

- **Key Findings and Conclusions of the Study**

  Based on the numerous variables taken into account, an MFI can determine whether or not a person will repay money and whether or not an MFI should issue a load to that individual.

- **Learning Outcomes of the Study in respect of Data Science**

  For greater accuracy, I developed many categorization models rather than relying on a single model, and I used cross validation to guarantee that the model did not suffer from overfitting or underfitting. To improve the scores, I chose the best one and applied hyper parameter tuning to it.

- **Limitations of this work and Scope for Future Work**

  The limitation is that it will only function for this specific use case, and it will need to be tweaked if used in a new scenario on a similar scale. The scope of this dataset is that we can use it in companies to determine whether we should provide a loan to a person or not, and we can also make predictions about a person purchasing an expensive service based on their personal details that we have in this dataset, such as the number of times their data account has been recharged in the last 30 days and the daily amount spent from their main account, averaged over the last 30 days (in Indonesian Rupiah), so even a marketing firm can use it.

Thank You

Thanks for your attention