Micro-Credit Defaulter Model

Submitted by:

Aishwary Shukla

ACKNOWLEDGMENT

The aim of this prohect is 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,in label column, Label '1' indicates that the loan has been payed i.e. Non- defaulter, while, Label '0' indicates that the loan has not been payed i.e. defaulter.

From this dataset models were build in Jupyter notebook using Python and ML libraries. After a preliminary exploratory data analysis and data review, it became apparent that the problem fell under classification category. Thus, we have used classification algorithms namely logistic regression, decision tree, Random Forest, KNeighbours Regressor, Support Vector Machine, AdaBoost Classifier, XGBoost respectively. The major discovery is that the machine learning approach should be suitable for these types of problems due to many aspects. Python programming language and its libraries namely Pandas, Numpy, Matplotlib, Seaborn etc are also a good choice for a first step, not the least because of the easily grasped user interface, as well as the wide availability of algorithms within machine learning. Advanced methods such as hyper parameter tuning of best models is also available.

References:

- https://www.researchgate.net
- https://www.kaggle.com
- https://www.investopedia.com

Special thanks to:

Mr. Shankar Gaud Tegimanni, DataTrained

Mr. Shwetank Mishra, FlipRobo

Mr. Prateeek Rajvanshi, Clevered Institute

And all colleagues, mentors, trainers from DataTrained and FlipRobo for training me and giving me the opportunity to be an intern and expand my knowledge in the field of Data Science and Artificial Intelligence.

INTRODUCTION

Problem Statement: 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 donorsare 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 MFShas been uneven with both significant challenges and successes.

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

We are working with one such client that is in Telecom Industry. They are a fixed wireless telecommunications network provider. They have launched various products and have developed its business and organization based on the budget operator model, offering better products at Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

They understand the importance of communication and how it affects a person's life, thus, focusing on providing their services and products to low income families and poor customers that can help them in the need of hour.

They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6(in Indonesian Rupiah), while, for the loan amount of 10(in Indonesian Rupiah), the payback amount should be 12(in Indonesian Rupiah).

The sample data is provided to us from our client database. It is hereby given to you for this exercise. 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.

Conceptual Background of the Domain Problem

Khandani et al. (**2010**) have employed machine learning techniques to build nonlinear non-parametric forecasting approaches to measure consumer credit risk. To identify credit cardholders' defaults, the authors used a credit office data set and commercial bank-customer transactions to establish a forecast estimation. Their results indicate cost savings from 6% to 25% of total losses when machine learning forecasting techniques are employed to estimate the delinquency rates. Besides, their study opens up questions of whether aggregated customer credit risk analytics may improve systematic risk estimation.

Yap et al. (**2011**) used historical payment data from a recreational club and established credit scoring techniques to identify potential club member subscription defaulters. The study results demonstrated that no model outperforms the others among a credit scorecard model, logistic regression, and a decision tree model. Each model generated almost identical accuracy figures.

Zhao et al. (**2015**) examined a multi-layer perceptron (MLP) neural network's accuracy regarding estimating credit scores efficiently. The authors used a German credit dataset to train and estimate the model's accuracy. Their results indicated an MLP model containing nine hidden units achieved a classification accuracy of 87%, higher than other similar experiments. Their study results proved the trend of MLP models' scoring accuracy by increasing the number of hidden units.

In **Addo et al.** (**2018**) the authors examined credit risk scoring by employing various machine and deep learning techniques. The authors used binary classifiers in modeling loan default probability (DP) estimations by incorporating ten key features to test the classifiers' stability by evaluating performance on separate data. Their results indicated that the models such as the logistic regression, random forest, and gradient boosting modeling generated more accurate results than the models based on the neural network approach incorporating various technicalities.

Petropoulos et al. (**2019**) studied a dataset of loan-level data of the Greek economy of examining credit quality performance and quantification of probability default for an evaluating period of 10 years. The authors used an extended example of classifications of the incorporated machine learning models against traditional methods, such as logistic regression. Their results identified that machine learning

models had demonstrated superior performance and forecasting accuracy through the financial credit rating cycle.

Provenzano et al. (**2020**) introduced machine learning models to compose credit rating and default prediction estimation. They used financial instruments, such as historical balance sheets, bankruptcy statutes, and macroeconomic variables of a Moody's dataset. Using machine learning models, the authors observed excellent out-of-sample performance results to reduce the bankruptcy probability or improve credit rating.

Review of Literature

Steps followed to conduct the research of finding important features that positively affect the sales price of Houses.

- Exploratory data analysis
- Feature Engineering
- Relation between different features/amenities of house and target column i.e. sales price.
- Correlation and identification of multicollinearity problem.
- Identifying and Selecting best features that affect sales price.
- Data pre-processing.
- Model Initialization and evaluation.
- Model Validation
- Model Testing and Pipelining.

Motivation for the Problem Undertaken

My objective behind making this project is to implement techniques and methods ive learned and practiced during my PG programme in Data Science. Banking and financial institutions interest me thus working on this project will give me the opportunity to work in the domain of micro credit finance. Furthur I'll apply my knowledge and technical expertise in data analysis and will further deploy ml models that financial institutions can use to reduce repayment defaulter cases.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem
- Data Sources and their formats
- Data Preprocessing Done
- Data Inputs- Logic- Output Relationships
- Assumptions related to the problem under consideration

Hardware and software requirements:

- Python and its libraries relevant to machine learning
- Computer with adequate specification.

Tools, libraries and packages used with describe:

NumPy:

NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.

Pandas:

Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for grouping, combining and filtering data.

Matplotlib.pyplot and Seaborn

Matplotlib and Seaborn **act 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.

%matplotlib inline

> Matplotlib Inline command is a command that makes the plots generated by matplotlib show into the IPython shell that we are running and not in a separate output window.

statsmodels.stats.outliers_influence import variance_inflation_factor

The variance inflation factor is a measure for the increase of the variance of the parameter estimates if an additional variable, given by exog_idx is added to the linear regression. It is a measure for multicollinearity of the design matrix, exog.

One recommendation is that if VIF is greater than 5, then the explanatory variable given by exog_idx is highly collinear with the other explanatory variables, and the parameter estimates will have large standard errors because of this.

from sklearn.feature_selection import SelectKBest, f_classif

> Scikit-learn API provides SelectKBest class for extracting best features of given dataset. The SelectKBest method selects the features according to the k highest score. By changing the 'score_func' parameter we can apply the method for both classification and regression data. Selecting best features is important process when we prepare a large dataset for training. It helps us to eliminate less important part of the data and reduce a training time.

from sklearn.preprocessing import PowerTransformer

Apply a power transform feature wise to make data more Gaussian-like.

Power transforms are a family of parametric, monotonic transformations that are applied to make data more Gaussian-like. This is useful for modeling issues related to heteroscedasticity (non-constant variance), or other situations where normality is desired.

Currently, PowerTransformer supports the Box-Cox transform and the Yeo-Johnson transform. The optimal parameter for stabilizing variance and minimizing skewness is estimated through maximum likelihood.

from sklearn.decomposition import PCA

- **PCA Description**: PCA is an unsupervised pre-processing task that is carried out before applying any ML algorithm. PCA is based on "orthogonal linear transformation" which is a mathematical technique to project the attributes of a data set onto a new coordinate system. The attribute which describes the most variance is called the first principal component and is placed at the first coordinate.
- Similarly, the attribute which stands second in describing variance is called a second principal component and so on. In short, the complete dataset can be expressed in terms of principal components. Usually, more than 90% of the variance is explained by two/three principal components.
- Principal component analysis, or PCA, thus converts data from high dimensional space to low dimensional space by selecting the most important attributes that capture maximum information about the dataset.

Importing algorithms

Importing metrics

Removing warnings

```
import warnings
warnings.filterwarnings('ignore')
```

Spliting data and hyperparameter tuning

```
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

The train_test_split() method is used to split our data into train and test sets.

GridSearchCV is a technique to search through the best parameter values from the given set of the grid of parameters. It is basically a cross-validation method. the model and the parameters are required to be fed in. Best parameter values are extracted and then the predictions are made.

Randomized search on hyper parameters. RandomizedSearchCV implements a "fit" and a "score" method. It also implements "score_samples", "predict", "predict_proba", "decision_function", "transform" and "inverse_transform" if they are implemented in the estimator used.

Pipelining

from sklearn.pipeline import Pipeline

The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters. For this, it enables setting parameters of the various steps using their names and the parameter name separated by a '__', as in the example below. A step's estimator may be replaced entirely by setting the parameter with its name to another estimator, or a transformer removed by setting it to 'pass-through ' or None.

Model/s Development and Evaluation

- Problem-solving approach taken into consideration:

1. Exploratory data analysis:
   Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

   EDA is primarily used to see what data can reveal beyond the formal modeling or hypothesis testing task and provides a provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate. Originally developed by American mathematician John Tukey in the 1970s, EDA techniques continue to be a widely used method in the data discovery process today.

2. Feature Engineering:

Feature Engineering is done to improve the performance of the model. Steps involved in feature engineering are:

a) Feature Selection: is nothing but a selection of required independent features. Selecting the important independent features which have more relation with the dependent feature will help to build a good model.

b) Handling missing values: it includes handling missing values in the dataset.

c) Handling imbalanced data: we handle imbalanced data to reduce overfitting and underfitting problem.

d) Handling outliers: values that are out of bounds are called outliers. These cause skewness in data.

e) Encoding: A dataset may contain object datatypes. for building a model we need to have all features are in integer datatypes. so, Label Encoder and OneHotEncoder are used to convert object datatype to integer datatype.

f) Feature Scaling: scaling is applied to reduce the variance effect and to overcome the fitting problem

3. Analysis using Plots:
   - Univariate analysis: Univariate Analysis is a type of data visualization where we visualize only a single variable at a time. Univariate Analysis helps us to analyze the distribution of the variable present in the data so that we can perform further analysis.

   - Buvariate analysis: Bivariate analysis is the simultaneous analysis of two variables. It explores the concept of the relationship between two variable whether there exists an association and the strength of this association or whether there are differences between two variables and the significance of these differences.

- Multivariate analysis: It is an extension of bivariate analysis which means it involves multiple variables at the same time to find correlation between them. Multivariate Analysis is a set of statistical model that examine patterns in multidimensional data by considering at once, several data variable.

4. Corelation and multicollinearity analysis: Collinearity is a linear association between two predictors. Multicollinearity is a situation where two or more predictors are highly linearly related. In general, **an absolute correlation coefficient of >0.7 among two or more predictors indicates the presence of multicollinearity**

5. Best Feature selection and PCA: **Principal component analysis** (**PCA**) is a popular technique for analyzing large datasets containing a high number of dimensions/features per observation, increasing the interpretability of data while preserving the maximum amount of information, and enabling the visualization of multidimensional data. Formally, PCA is a statistical technique for reducing the dimensionality of a dataset. This is accomplished by linearly transforming the data into a new coordinate system where (most of) the variation in the data can be described with fewer dimensions than the initial data.

6. Model Selection, model training, and model evaluation: A machine learning model determines the output you get after running a machine learning algorithm on the collected data. It is important to choose a model which is relevant to the task at hand. Over the years, scientists and engineers developed various models suited for different tasks like speech recognition, image recognition, prediction, etc. Apart from this, you also have to see if your model is suited for numerical or categorical data and choose accordingly.

   Training is the most important step in machine learning. In training, you pass the prepared data to your machine learning model to find patterns and make predictions. It results in the model learning from the data so that it can accomplish the task set. Over time, with training, the model gets better at predicting.

After training your model, you have to check to see how it's performing. This is done by testing the performance of the model on previously unseen data. The unseen data used is the testing set that you split our data into earlier. If testing was done on the same data which is used for training, you will not get an accurate measure, as the model is already used to the data, and finds the same patterns in it, as it previously did. This will give you disproportionately high accuracy.

7. Parameter tuning: Once you have created and evaluated your model, see if its accuracy can be improved in any way. This is done by tuning the parameters present in your model. Parameters are the variables in the model that the programmer generally decides. At a particular value of your parameter, the accuracy will be the maximum. Parameter tuning refers to finding these values.

8. Best model selection, making predictions and Pipelining

Model selection is the process of choosing one of the models as the final model that addresses the problem.
Model selection is different from **model assessment**.
For example, we evaluate or assess candidate models in order to choose the best one, and this is model selection. Whereas once a model is chosen, it can be evaluated in order to communicate how well it is expected to perform in general; this is model assessment.

The process of evaluating a model's performance is known as model assessment, whereas the process of selecting the proper level of flexibility for a model is known as model selection.

"Prediction" refers to **the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome**, such as whether or not a customer will churn in 30 days.

A machine learning pipeline is a way to codify and automate the workflow it takes to produce a machine learning model. Machine learning pipelines consist of multiple

sequential steps that do everything from data extraction and preprocessing to model training and deployment.

For data science teams, the production pipeline should be the central product. It encapsulates all the learned best practices of producing a machine learning model for the organization's use-case and allows the team to execute at scale. Whether you are maintaining multiple models in production or supporting a single model that needs to be updated frequently, an end-to-end machine learning pipeline is a must.

Exploratory Data Analysis:

Functions and methods used:

1. Importing Libraries:

```
# Imp libs:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Standardizing and Normalizing data:
from sklearn.preprocessing import StandardScaler

# Spliting data
from sklearn.model_selection import train_test_split, GridSearchCV

# Algorithms
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
import itertools
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.naive_bayes import GaussianNB
# Importing metrics
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

from statsmodels.stats.outliers_influence import variance_inflation_factor

# Removing warnings
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

1. Reading csv files using pd.read_csv function and Inspecting 1st and last rows using .head and .tail method.

```
df = pd.read_csv(r'C:\Users\Lenovo\OneDrive\Desktop\Filp Robo worksheets and projects\Micro Credit Project\Data File.csv')
```

```
df
```

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	med
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

```
print('First 10 rows')
df.head(10)
```

First 10 rows

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	medianam
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0	
5	6	1	35819I70783	568.0	2257.362667	2261.460000	368.13	380.13	2.0	0.0	...	6.0	
6	7	1	96759I84459	545.0	2876.641667	2883.970000	335.75	402.90	13.0	0.0	...	6.0	
7	8	1	09832I90846	768.0	12905.000000	17804.150000	900.35	2549.11	4.0	55.0	...	6.0	
8	9	1	59772I84450	1191.0	90.695000	90.695000	2287.50	2287.50	1.0	0.0	...	6.0	
9	10	1	56331I70783	536.0	29.357333	29.357333	612.96	612.96	11.0	0.0	...	6.0	

10 rows × 37 columns

```
print('Last 10 rows')
df.tail(10)
```

Last 10 rows

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	me
209583	209584	1	30201I82732	935.0	15.550667	15.550667	140.44	140.44	4.0	0.0	...	6.0	
209584	209585	0	70387I89237	945.0	0.000000	0.000000	78.30	78.30	0.0	0.0	...	6.0	
209585	209586	1	12227I90843	793.0	5350.315333	5356.210000	594.80	640.20	2.0	0.0	...	6.0	
209586	209587	1	60331I70370	254.0	20079.157333	20228.090000	4003.12	5173.74	2.0	0.0	...	12.0	
209587	209588	1	19900I95200	239.0	14678.000000	14704.900000	12935.26	16775.60	8.0	0.0	...	12.0	
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	

10 rows × 37 columns

2. Inspecting rows and columns in dataset/Checking datatypes:

209593 rows × 37 columns

```
df.columns
```

```
Index(['Unnamed: 0', 'label', 'msisdn', 'aon', 'daily_decr30', 'daily_decr90',
       'rental30', 'rental90', 'last_rech_date_ma', 'last_rech_date_da',
       'last_rech_amt_ma', 'cnt_ma_rech30', 'fr_ma_rech30',
       'sumamnt_ma_rech30', 'medianamnt_ma_rech30', 'medianmarechprebal30',
       'cnt_ma_rech90', 'fr_ma_rech90', 'sumamnt_ma_rech90',
       'medianamnt_ma_rech90', 'medianmarechprebal90', 'cnt_da_rech30',
       'fr_da_rech30', 'cnt_da_rech90', 'fr_da_rech90', 'cnt_loans30',
       'amnt_loans30', 'maxamnt_loans30', 'medianamnt_loans30', 'cnt_loans90',
       'amnt_loans90', 'maxamnt_loans90', 'medianamnt_loans90', 'payback30',
       'payback90', 'pcircle', 'pdate'],
      dtype='object')
```

```
df.dtypes
```

```
Unnamed: 0                int64
label                     int64
msisdn                   object
aon                     float64
daily_decr30            float64
daily_decr90            float64
rental30                float64
rental90                float64
last_rech_date_ma       float64
last_rech_date_da       float64
last_rech_amt_ma          int64
cnt_ma_rech30             int64
fr_ma_rech30            float64
sumamnt_ma_rech30       float64
medianamnt_ma_rech30    float64
medianmarechprebal30    float64
cnt_ma_rech90             int64
fr_ma_rech90              int64
sumamnt_ma_rech90         int64
medianamnt_ma_rech90    float64
medianmarechprebal90    float64
cnt_da_rech30           float64
fr_da_rech30            float64
cnt_da_rech90             int64
fr_da_rech90              int64
cnt_loans30               int64
amnt_loans30              int64
maxamnt_loans30         float64
medianamnt_loans30      float64
cnt_loans90             float64
amnt_loans90              int64
maxamnt_loans90           int64
medianamnt_loans90      float64
payback30               float64
payback90               float64
pcircle                  object
pdate                    object
dtype: object
```

3. Checking null values in dataset:

```
: null_data = df.isnull().sum(
  null_data.sort_values(ascend
```

```
: Unnamed: 0                0
  medianamnt_ma_rech90      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
  medianmarechprebal90      0
  sumamnt_ma_rech90         0
  label                     0
  fr_ma_rech90              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
  pdate                     0
  dtype: int64
```

4. Feature Engineering,/Selecting categorical and numerical columns:

```
df.drop(columns=['Unnamed: 0'],axis=0,inplace=True)
```

```
# Feature Engineering

# spliting pdate into dd/mm/yy and changing dtype:

date = []
month = []
year = []

for i in df['pdate']:
    year.append( i.split('-')[0])

for i in df['pdate']:
    month.append( i.split('-')[1])

for i in df['pdate']:
    date.append( i.split('-')[2])

df['Date'] = date
df['Month'] = month
df['Year'] = year

df['Date'] = df['Date'].astype('int64')
df['Month'] = df['Month'].astype('int64')
df['Year'] = df['Year'].astype('int64')

df.drop(columns = ['pdate'],axis=0,inplace=True)
```

```
new_msisdn = []

for i in df['msisdn']:
    new_msisdn.append(i[0:5]+i[6:11])

df['msisdn'] = new_msisdn

df['msisdn'] = df['msisdn'].astype('int64')
```

```
new_cols = ['Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure
'mobile number of user',
'age on cellular network in days',
'Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)',
'Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)',
'Average main account balance over last 30 days',
'Average main account balance over last 90 days',
'Number of days till last recharge of main account',
'Number of days till last recharge of data account',
'Amount of last recharge of main account (in Indonesian Rupiah)',
'Number of times main account got recharged in last 30 days',
'Frequency of main account recharged in last 30 days',
'Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)',
'Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)',
'Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)',
'Number of times main account got recharged in last 90 days',
'Frequency of main account recharged in last 90 days',
'Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)',
'Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)',
'Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)',
'Number of times data account got recharged in last 30 days',
'Frequency of data account recharged in last 30 days',
'Number of times data account got recharged in last 90 days',
'Frequency of data account recharged in last 90 days',
'Number of loans taken by user in last 30 days',
'Total amount of loans taken by user in last 30 days',
'maximum amount of loan taken by the user in last 30 days',
'Median of amounts of loan taken by the user in last 30 days',
'Number of loans taken by user in last 90 days',
'Total amount of loans taken by user in last 90 days',
'maximum amount of loan taken by the user in last 90 days',
'Median of amounts of loan taken by the user in last 90 days',
'Average payback time in days over last 30 days',
'Average payback time in days over last 90 days',
'telecom circle',
'Date', 'Month', 'Year']

df.columns = new_cols
```

5. Data description: of numerical type columns and categorical columns:

```
desc = df.describe().T
desc['range']=desc['max']-desc['min']
desc
```

	count	mean	std	min	25%	50%	75%	max	range
Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}	209593.0	8.751771e-01	3.305187e-01	0.000000	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
mobile number of user	209593.0	4.974956e+09	2.890571e+09	482738.000000	2.465991e+09	4.905684e+09	7.503370e+09	9.999895e+09	9.999412e+09
age on cellular network in days	209593.0	8.112343e+03	7.569608e+04	-48.000000	2.460000e+02	5.270000e+02	9.820000e+02	9.998608e+05	9.999088e+05
Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)	209593.0	5.381402e+03	9.220623e+03	-93.012667	4.244000e+01	1.469178e+03	7.244000e+03	2.659260e+05	2.660190e+05
Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)	209593.0	6.082515e+03	1.091881e+04	-93.012667	4.269200e+01	1.500000e+03	7.802790e+03	3.206300e+05	3.207230e+05
Average main account balance over last 30 days	209593.0	2.692582e+03	4.308587e+03	-23737.140000	2.804200e+02	1.083570e+03	3.356940e+03	1.989261e+05	2.226632e+05
Average main account balance over last 90 days	209593.0	3.483407e+03	5.770461e+03	-24720.580000	3.002600e+02	1.334000e+03	4.201790e+03	2.001481e+05	2.248687e+05
Number of days till last recharge of main account	209593.0	3.755848e+03	5.390589e+04	-29.000000	1.000000e+00	3.000000e+00	7.000000e+00	9.986504e+05	9.986794e+05
Number of days till last recharge of data account	209593.0	3.712203e+03	5.337483e+04	-29.000000	0.000000e+00	0.000000e+00	0.000000e+00	9.991718e+05	9.992008e+05
Amount of last recharge of main account (in Indonesian Rupiah)	209593.0	2.064453e+03	2.370786e+03	0.000000	7.700000e+02	1.539000e+03	2.309000e+03	5.500000e+04	5.500000e+04
Number of times main account got recharged in last 30 days	209593.0	3.978057e+00	4.256090e+00	0.000000	1.000000e+00	3.000000e+00	5.000000e+00	2.030000e+02	2.030000e+02
Frequency of main account recharged in last 30 days	209593.0	3.737355e+03	5.364363e+04	0.000000	0.000000e+00	2.000000e+00	6.000000e+00	9.998064e+05	9.998064e+05
Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)	209593.0	7.704501e+03	1.013962e+04	0.000000	1.540000e+03	4.628000e+03	1.001000e+04	8.100960e+05	8.100960e+05
Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)	209593.0	1.812818e+03	2.070865e+03	0.000000	7.700000e+02	1.539000e+03	1.924000e+03	5.500000e+04	5.500000e+04

Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)	209593.0	3.851928e+03	5.400637e+04	-200.000000	1.100000e+01	3.390000e+01	8.300000e+01	9.994794e+05	9.996794e+05
Number of times main account got recharged in last 90 days	209593.0	6.315430e+00	7.193470e+00	0.000000	2.000000e+00	4.000000e+00	8.000000e+00	3.360000e+02	3.360000e+02
Frequency of main account recharged in last 90 days	209593.0	7.716780e+00	1.259025e+01	0.000000	0.000000e+00	2.000000e+00	8.000000e+00	8.800000e+01	8.800000e+01
Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)	209593.0	1.239622e+04	1.685779e+04	0.000000	2.317000e+03	7.226000e+03	1.600000e+04	9.530360e+05	9.530360e+05
Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)	209593.0	1.864596e+03	2.081681e+03	0.000000	7.730000e+02	1.539000e+03	1.924000e+03	5.500000e+04	5.500000e+04
Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)	209593.0	9.202554e+01	3.692157e+02	-200.000000	1.460000e+01	3.600000e+01	7.931000e+01	4.145650e+04	4.165650e+04
Number of times data account got recharged in last 30 days	209593.0	2.625781e+02	4.183898e+03	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	9.991444e+04	9.991444e+04
Frequency of data account recharged in last 30 days	209593.0	3.749494e+03	5.388541e+04	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	9.998092e+05	9.998092e+05
Number of times data account got recharged in last 90 days	209593.0	4.149471e-02	3.975557e-01	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	3.800000e+01	3.800000e+01
Frequency of data account recharged in last 90 days	209593.0	4.571240e-02	9.513858e-01	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	6.400000e+01	6.400000e+01
Number of loans taken by user in last 30 days	209593.0	2.758981e+00	2.554502e+00	0.000000	1.000000e+00	2.000000e+00	4.000000e+00	5.000000e+01	5.000000e+01
Total amount of loans taken by user in last 30 days	209593.0	1.795202e+01	1.737974e+01	0.000000	6.000000e+00	1.200000e+01	2.400000e+01	3.060000e+02	3.060000e+02
maximum amount of loan taken by the user in last 30 days	209593.0	2.746587e+02	4.245265e+03	0.000000	6.000000e+00	6.000000e+00	6.000000e+00	9.986456e+04	9.986456e+04
Median of amounts of loan taken by the user in last 30 days	209593.0	5.402852e-02	2.180386e-01	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	3.000000e+00	3.000000e+00
Number of loans taken by user in last 90 days	209593.0	1.852092e+01	2.247974e+02	0.000000	1.000000e+00	2.000000e+00	5.000000e+00	4.997518e+03	4.997518e+03
Total amount of loans taken by user in last 90 days	209593.0	2.364540e+01	2.646986e+01	0.000000	6.000000e+00	1.200000e+01	3.000000e+01	4.380000e+02	4.380000e+02
maximum amount of loan taken by the user in last 90 days	209593.0	6.703134e+00	2.103864e+00	0.000000	6.000000e+00	6.000000e+00	6.000000e+00	1.200000e+01	1.200000e+01
Median of amounts of loan taken by the user in last 90 days	209593.0	4.607740e-02	2.006915e-01	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	3.000000e+00	3.000000e+00
Average payback time in days over last 30 days	209593.0	3.398826e+00	8.813729e+00	0.000000	0.000000e+00	0.000000e+00	3.750000e+00	1.715000e+02	1.715000e+02
Average payback time in days over last 90 days	209593.0	4.321485e+00	1.030811e+01	0.000000	0.000000e+00	1.666667e+00	4.500000e+00	1.715000e+02	1.715000e+02
Date	209593.0	1.439894e+01	8.438900e+00	1.000000	7.000000e+00	1.400000e+01	2.100000e+01	3.100000e+01	3.000000e+01
Month	209593.0	6.797321e+00	7.414352e-01	6.000000	6.000000e+00	7.000000e+00	7.000000e+00	8.000000e+00	2.000000e+00
Year	209593.0	2.016000e+03	0.000000e+00	2016.000000	2.016000e+03	2.016000e+03	2.016000e+03	2.016000e+03	0.000000e+00

```
df.describe(include='object').T
```

	count	unique	top	freq
telecom circle	209593	1	UPW	209593

6. Unique values/value counts:

```
for i in cat_columns:
    print('For column',i,'unique values are: ',df[i].unique())
    print('For column',i,'count of unique values are: ',df[i].nunique(),'\n\n')
```

```
For column telecom circle unique values are:  ['UPW']
For column telecom circle count of unique values are:  1
```

```
df['Date'].value_counts()
```

```
11    8092
10    8050
6     8030
12    8028
7     8026
5     7989
13    7969
8     7899
2     7839
1     7824
15    7820
14    7816
9     7717
17    7643
3     7607
16    7556
18    7305
4     7154
19    6857
20    6729
21    5964
23    5816
22    5753
27    5283
25    5269
26    5174
30    5129
24    5103
29    5077
28    4897
31    2178
Name: Date, dtype: int64
```

```
df['Month'].value_counts()
```

```
7    85765
6    83154
8    40674
Name: Month, dtype: int64
```

```
df['Year'].value_counts()
```

```
2016    209593
Name: Year, dtype: int64
```

```
print(len(date))
print(len(month))
print(len(year))
```

```
209593
209593
209593
```

```
df['mobile number of user'][0:5]
```

```
0    2140870789
1    7646270374
2    1794370372
3    5577370781
4     381382730
Name: mobile number of user, dtype: int64
```

```
df.dtypes
```

```
Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}      int64
mobile number of user                                                                                                    int64
age on cellular network in days                                                                                        float64
Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)                                float64
Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)                                float64
Average main account balance over last 30 days                                                                         float64
Average main account balance over last 90 days                                                                         float64
Number of days till last recharge of main account                                                                      float64
Number of days till last recharge of data account                                                                      float64
Amount of last recharge of main account (in Indonesian Rupiah)                                                           int64
Number of times main account got recharged in last 30 days                                                               int64
Frequency of main account recharged in last 30 days                                                                    float64
Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)                                      float64
Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)              float64
Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)               float64
Number of times main account got recharged in last 90 days                                                                int64
Frequency of main account recharged in last 90 days                                                                       int64
Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)                                         int64
Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)              float64
Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)               float64
Number of times data account got recharged in last 30 days                                                             float64
Frequency of data account recharged in last 30 days                                                                    float64
Number of times data account got recharged in last 90 days                                                                int64
Frequency of data account recharged in last 90 days                                                                       int64
Number of loans taken by user in last 30 days                                                                             int64
Total amount of loans taken by user in last 30 days                                                                       int64
maximum amount of loan taken by the user in last 30 days                                                               float64
Median of amounts of loan taken by the user in last 30 days                                                            float64
Number of loans taken by user in last 90 days                                                                          float64
Total amount of loans taken by user in last 90 days                                                                       int64
maximum amount of loan taken by the user in last 90 days                                                                  int64
Median of amounts of loan taken by the user in last 90 days                                                            float64
Average payback time in days over last 30 days                                                                         float64
Average payback time in days over last 90 days                                                                         float64
telecom circle                                                                                                          object
Date                                                                                                                       int64
Month                                                                                                                      int64
Year                                                                                                                       int64
dtype: object
```

7. Data analysis on the basis of label values:

```
df_success = df[df['Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:
df_failure = df[df['Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:
```

```
df_success.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}	183431.0	1.000000e+00	0.000000e+00	1.000000	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
mobile number of user	183431.0	4.977080e+09	2.889324e+09	482738.000000	2.471891e+09	4.907489e+09	7.504341e+09	9.999895e+09
age on cellular network in days	183431.0	8.004140e+03	7.510604e+04	-48.000000	2.590000e+02	5.450000e+02	1.002000e+03	9.998608e+05
Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)	183431.0	5.967455e+03	9.641761e+03	-93.012667	5.620333e+01	2.127695e+03	8.274000e+03	2.659260e+05
Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)	183431.0	6.767646e+03	1.144642e+04	-93.012667	5.634850e+01	2.190950e+03	9.037790e+03	3.206300e+05
Average main account balance over last 30 days	183431.0	2.787095e+03	4.373509e+03	-23737.140000	3.105950e+02	1.152070e+03	3.519615e+03	1.989261e+05
Average main account balance over last 90 days	183431.0	3.647985e+03	5.926903e+03	-24720.580000	3.577100e+02	1.443330e+03	4.418190e+03	2.001481e+05
Number of days till last recharge of main account	183431.0	3.831747e+03	5.451509e+04	-29.000000	1.000000e+00	3.000000e+00	7.000000e+00	9.986504e+05
Number of days till last recharge of data account	183431.0	3.746693e+03	5.361809e+04	-29.000000	0.000000e+00	0.000000e+00	0.000000e+00	9.979203e+05
Amount of last recharge of main account (in Indonesian Rupiah)	183431.0	2.182462e+03	2.386379e+03	0.000000	7.700000e+02	1.539000e+03	2.309000e+03	5.500000e+04
Number of times main account got recharged in last 30 days	183431.0	4.359530e+00	4.353103e+00	0.000000	1.000000e+00	3.000000e+00	6.000000e+00	2.030000e+02
Frequency of main account recharged in last 30 days	183431.0	3.764303e+03	5.392291e+04	0.000000	0.000000e+00	2.000000e+00	6.000000e+00	9.996064e+05
Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)	183431.0	8.481192e+03	1.047280e+04	0.000000	2.309000e+03	5.498000e+03	1.087600e+04	8.100960e+05
Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)	183431.0	1.923474e+03	2.080892e+03	0.000000	7.715000e+02	1.539000e+03	1.924000e+03	5.500000e+04
Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)	183431.0	3.753438e+03	5.324214e+04	-200.000000	1.500000e+01	3.900000e+01	9.011000e+01	9.994794e+05
Number of times main account got recharged in last 90 days	183431.0	6.957630e+00	7.397362e+00	0.000000	2.000000e+00	5.000000e+00	9.000000e+00	3.360000e+02
Frequency of main account recharged in last 90 days	183431.0	8.118012e+00	1.264578e+01	0.000000	0.000000e+00	3.000000e+00	9.000000e+00	8.800000e+01
Total amount of recharge in main account over last 90 days (in Indonesian Rupiah)	183431.0	1.370640e+04	1.747384e+04	0.000000	3.252000e+03	8.473000e+03	1.759350e+04	9.530360e+05
Median of amount of recharges done in main account over last 90 days at user level (in Indonesian Rupiah)	183431.0	1.959608e+03	2.079656e+03	0.000000	7.730000e+02	1.539000e+03	1.928000e+03	5.500000e+04

days	183431.0	8.118012e+00	1.264578e+01	0.000000	0.000000e+00	3.000000e+00	9.000000e+00	8.800000e+01
Total amount of recharge in main account over last 50 days (in Indonesian Rupiah)	183431.0	1.370640e+04	1.747384e+04	0.000000	3.252000e+03	8.473000e+03	1.759350e+04	9.530360e+05
Median of amount of recharges done in main account over last 50 days at user level (in Indonesian Rupiah)	183431.0	1.959608e+03	2.079656e+03	0.000000	7.730000e+02	1.539000e+03	1.928000e+03	5.500000e+04
Median of main account balance just before recharge in last 50 days at user level (in Indonesian Rupiah)	183431.0	9.750539e+01	3.441674e+02	-200.000000	1.800000e+01	4.000000e+01	8.500000e+01	4.145650e+04
Number of times data account got recharged in last 30 days	183431.0	2.686255e+02	4.243575e+03	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	9.991444e+04
Frequency of data account recharged in last 30 days	183431.0	3.748955e+03	5.382133e+04	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	9.998092e+05
Number of times data account got recharged in last 90 days	183431.0	4.194493e-02	3.887649e-01	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	3.800000e+01
Frequency of data account recharged in last 90 days	183431.0	4.376578e-02	9.441316e-01	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	6.400000e+01
Number of loans taken by user in last 30 days	183431.0	2.948340e+00	2.645055e+00	0.000000	1.000000e+00	2.000000e+00	4.000000e+00	5.000000e+01
Total amount of loans taken by user in last 30 days	183431.0	1.924683e+01	1.800788e+01	0.000000	6.000000e+00	1.200000e+01	2.400000e+01	3.060000e+02
maximum amount of loan taken by the user in last 30 days	183431.0	2.750560e+02	4.243452e+03	0.000000	6.000000e+00	6.000000e+00	6.000000e+00	9.986456e+04
Median of amounts of loan taken by the user in last 30 days	183431.0	5.770017e-02	2.255948e-01	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	3.000000e+00
Number of loans taken by user in last 90 days	183431.0	1.892275e+01	2.256075e+02	0.000000	1.000000e+00	3.000000e+00	5.000000e+00	4.997518e+03
Total amount of loans taken by user in last 90 days	183431.0	2.564259e+01	2.751614e+01	0.000000	6.000000e+00	1.800000e+01	3.000000e+01	4.380000e+02
maximum amount of loan taken by the user in last 90 days	183431.0	6.769990e+00	2.197498e+00	0.000000	6.000000e+00	6.000000e+00	6.000000e+00	1.200000e+01
Median of amounts of loan taken by the user in last 90 days	183431.0	4.878674e-02	2.069159e-01	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	3.000000e+00
Average payback time in days over last 30 days	183431.0	3.559715e+00	8.633455e+00	0.000000	0.000000e+00	1.500000e+00	4.000000e+00	1.715000e+02
Average payback time in days over last 90 days	183431.0	4.512952e+00	1.011878e+01	0.000000	0.000000e+00	2.040000e+00	4.800000e+00	1.715000e+02
Date	183431.0	1.442069e+01	8.538290e+00	1.000000	7.000000e+00	1.400000e+01	2.100000e+01	3.100000e+01
Month	183431.0	6.840708e+00	7.598691e-01	6.000000	6.000000e+00	7.000000e+00	7.000000e+00	8.000000e+00
Year	183431.0	2.016000e+03	0.000000e+00	2016.000000	2.016000e+03	2.016000e+03	2.016000e+03	2.016000e+03

```
df_failure.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}	26162.0	0.000000e+00	0.000000e+00	0.000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
mobile number of user	26162.0	4.960068e+09	2.899314e+09	1782735.000	2.426210e+09	4.892384e+09	7.497071e+09	9.999485e+09
age on cellular network in days	26162.0	8.870999e+03	7.970789e+04	-48.000	1.710000e+02	3.990000e+02	8.230000e+02	9.951483e+05
Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)	26162.0	1.272377e+03	3.167434e+03	-18.378	5.333333e-01	3.344750e+01	1.117274e+03	1.037649e+05
Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)	26162.0	1.278818e+03	3.181302e+03	-18.378	5.666667e-01	3.519500e+01	1.125888e+03	1.038846e+05
Average main account balance over last 30 days	26162.0	2.029915e+03	3.756423e+03	-5686.030	6.742250e+01	5.916200e+02	2.307932e+03	7.421444e+04
Average main account balance over last 90 days	26162.0	2.329486e+03	4.352825e+03	-6422.260	7.800000e+01	6.599400e+02	2.619698e+03	9.159572e+04
Number of days till last recharge of main account	26162.0	3.223694e+03	4.942187e+04	-28.000	0.000000e+00	3.000000e+00	1.100000e+01	9.976768e+05
Number of days till last recharge of data account	26162.0	3.470381e+03	5.163747e+04	0.000	0.000000e+00	0.000000e+00	0.000000e+00	9.991718e+05
Amount of last recharge of main account (in Indonesian Rupiah)	26162.0	1.237046e+03	2.078090e+03	0.000	0.000000e+00	7.700000e+02	1.539000e+03	5.500000e+04
Number of times main account got recharged in last 30 days	26162.0	1.303417e+00	2.021023e+00	0.000	0.000000e+00	1.000000e+00	2.000000e+00	3.900000e+01
Frequency of main account recharged in last 30 days	26162.0	3.548411e+03	5.164367e+04	0.000	0.000000e+00	0.000000e+00	1.000000e+00	9.942058e+05
Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)	26162.0	2.258850e+03	4.558161e+03	0.000	0.000000e+00	7.730000e+02	2.320000e+03	1.266260e+05
Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)	26162.0	1.036987e+03	1.819041e+03	0.000	0.000000e+00	7.700000e+02	1.539000e+03	5.500000e+04
Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)	26162.0	4.542477e+03	5.908422e+04	-200.000	0.000000e+00	3.635000e+00	3.900000e+01	9.910998e+05
Number of times main account got recharged in last 90 days	26162.0	1.812744e+00	2.778879e+00	0.000	0.000000e+00	1.000000e+00	2.000000e+00	5.100000e+01
Frequency of main account recharged in last 90 days	26162.0	4.903601e+00	1.181741e+01	0.000	0.000000e+00	0.000000e+00	2.000000e+00	8.700000e+01
Total amount of recharge in main account over last 90 days (in Indonesian Rupiah)	26162.0	3.210103e+03	6.283083e+03	0.000	0.000000e+00	1.539000e+03	3.851000e+03	1.756760e+05

Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)	26162.0	1.198435e+03	1.971179e+03	0.000	0.000000e+00	7.730000e+02	1.539000e+03	5.500000e+04
Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)	26162.0	5.360437e+01	5.098321e+02	-200.000	0.000000e+00	1.000000e+01	4.400000e+01	2.761170e+04
Number of times data account got recharged in last 30 days	26162.0	2.201777e+02	3.738622e+03	0.000	0.000000e+00	0.000000e+00	0.000000e+00	9.966585e+04
Frequency of data account recharged in last 30 days	26162.0	3.753279e+03	5.433367e+04	0.000	0.000000e+00	0.000000e+00	0.000000e+00	9.949359e+05
Number of times data account got recharged in last 90 days	26162.0	3.833805e-02	4.544353e-01	0.000	0.000000e+00	0.000000e+00	0.000000e+00	3.000000e+01
Frequency of data account recharged in last 90 days	26162.0	5.936091e-02	1.000684e+00	0.000	0.000000e+00	0.000000e+00	0.000000e+00	3.500000e+01
Number of loans taken by user in last 30 days	26162.0	1.431313e+00	1.100109e+00	1.000	1.000000e+00	1.000000e+00	1.000000e+00	2.600000e+01
Total amount of loans taken by user in last 30 days	26162.0	8.873834e+00	7.213658e+00	6.000	6.000000e+00	6.000000e+00	6.000000e+00	1.560000e+02
maximum amount of loan taken by the user in last 30 days	26162.0	2.718738e+02	4.258033e+03	6.000	6.000000e+00	6.000000e+00	6.000000e+00	9.957111e+04
Median of amounts of loan taken by the user in last 30 days	26162.0	2.828530e-02	1.525830e-01	0.000	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
Number of loans taken by user in last 90 days	26162.0	1.570355e+01	2.190174e+02	1.000	1.000000e+00	1.000000e+00	2.000000e+00	4.894102e+03
Total amount of loans taken by user in last 90 days	26162.0	9.642382e+00	8.976451e+00	6.000	6.000000e+00	6.000000e+00	1.200000e+01	1.560000e+02
maximum amount of loan taken by the user in last 90 days	26162.0	6.234386e+00	1.162510e+00	6.000	6.000000e+00	6.000000e+00	6.000000e+00	1.200000e+01
Median of amounts of loan taken by the user in last 90 days	26162.0	2.708126e-02	1.485841e-01	0.000	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
Average payback time in days over last 30 days	26162.0	2.270777e+00	9.913831e+00	0.000	0.000000e+00	0.000000e+00	0.000000e+00	1.580000e+02
Average payback time in days over last 90 days	26162.0	2.979047e+00	1.145945e+01	0.000	0.000000e+00	0.000000e+00	0.000000e+00	1.580000e+02
Date	26162.0	1.424643e+01	7.704552e+00	1.000	8.000000e+00	1.400000e+01	2.000000e+01	3.000000e+01
Month	26162.0	6.493120e+00	4.999622e-01	6.000	6.000000e+00	6.000000e+00	7.000000e+00	7.000000e+00
Year	26162.0	2.016000e+03	0.000000e+00	2016.000	2.016000e+03	2.016000e+03	2.016000e+03	2.016000e+03

,

8. Grouped Data analysis:

	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}	0	1
mobile number of user	mean	4.960068e+09	4.977080e+09
	median	4.892384e+09	4.907489e+09
age on cellular network in days	mean	8.870999e+03	8.004140e+03
	median	3.990000e+02	5.450000e+02
Number of days till last recharge of main account	mean	3.223694e+03	3.831747e+03
	median	3.000000e+00	3.000000e+00
Number of days till	mean	3.470381e+03	3.746693e+03

	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}	0	1
last recharge of data account	median	0.000000e+00	0.000000e+00
Amount of last recharge of main account (in Indonesian Rupiah)	mean	1.237046e+03	2.182462e+03
	median	7.700000e+02	1.539000e+03
Frequency of main account recharged in last 30 days	mean	3.548411e+03	3.764303e+03
	median	0.000000e+00	2.000000e+00
Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)	mean	1.036967e+03	1.923474e+03
	median	7.700000e+02	1.539000e+03
Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)	mean	4.542477e+03	3.753438e+03
	median	3.635000e+00	3.900000e+01
Frequency of main account recharged in last 90 days	mean	4.903601e+00	8.118012e+00
	median	0.000000e+00	3.000000e+00
Median of amount of	mean	1.198435e+03	1.959608e+03

	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}	0	1
recharges done in main account over last 90 days at user level (in Indonasian Rupiah)	median	7.730000e+02	1.539000e+03
Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)	mean	5.360437e+01	9.750539e+01
	median	1.000000e+01	4.000000e+01
Number of times data account got recharged in last 30 days	mean	2.201777e+02	2.686255e+02
	median	0.000000e+00	0.000000e+00
Frequency of data account recharged in last 30 days	mean	3.753279e+03	3.748955e+03
	median	0.000000e+00	0.000000e+00
Number of times data account got recharged in last 90 days	mean	3.833805e-02	4.194493e-02
	median	0.000000e+00	0.000000e+00
Frequency of data account recharged in last 90 days	mean	5.936091e-02	4.376578e-02
	median	0.000000e+00	0.000000e+00
maximum amount of	mean	2.718738e+02	2.750560e+02

	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}	0	1
loan taken by the user in last 30 days	median	6.000000e+00	6.000000e+00
Median of amounts of loan taken by the user in last 30 days	mean	2.828530e-02	5.770017e-02
	median	0.000000e+00	0.000000e+00
Number of loans taken by user in last 90 days	mean	1.570355e+01	1.892275e+01
	median	1.000000e+00	3.000000e+00
maximum amount of loan taken by the user in last 90 days	mean	6.234386e+00	6.769990e+00
	median	6.000000e+00	6.000000e+00
Median of amounts of loan taken by the user in last 90 days	mean	2.708126e-02	4.878674e-02
	median	0.000000e+00	0.000000e+00
Average payback time in days over last 30 days	mean	2.270777e+00	3.559715e+00
	median	0.000000e+00	1.500000e+00
Average payback time in days over last 90 days	mean	2.979047e+00	4.512952e+00
	median	0.000000e+00	2.040000e+00
Date	mean	1.424643e+01	1.442069e+01

	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}	0	1
	median	1.400000e+01	1.400000e+01
Month	**mean**	6.493120e+00	6.840708e+00
	median	6.000000e+00	7.000000e+00
Year	**mean**	2.016000e+03	2.016000e+03
	median	2.016000e+03	2.016000e+03

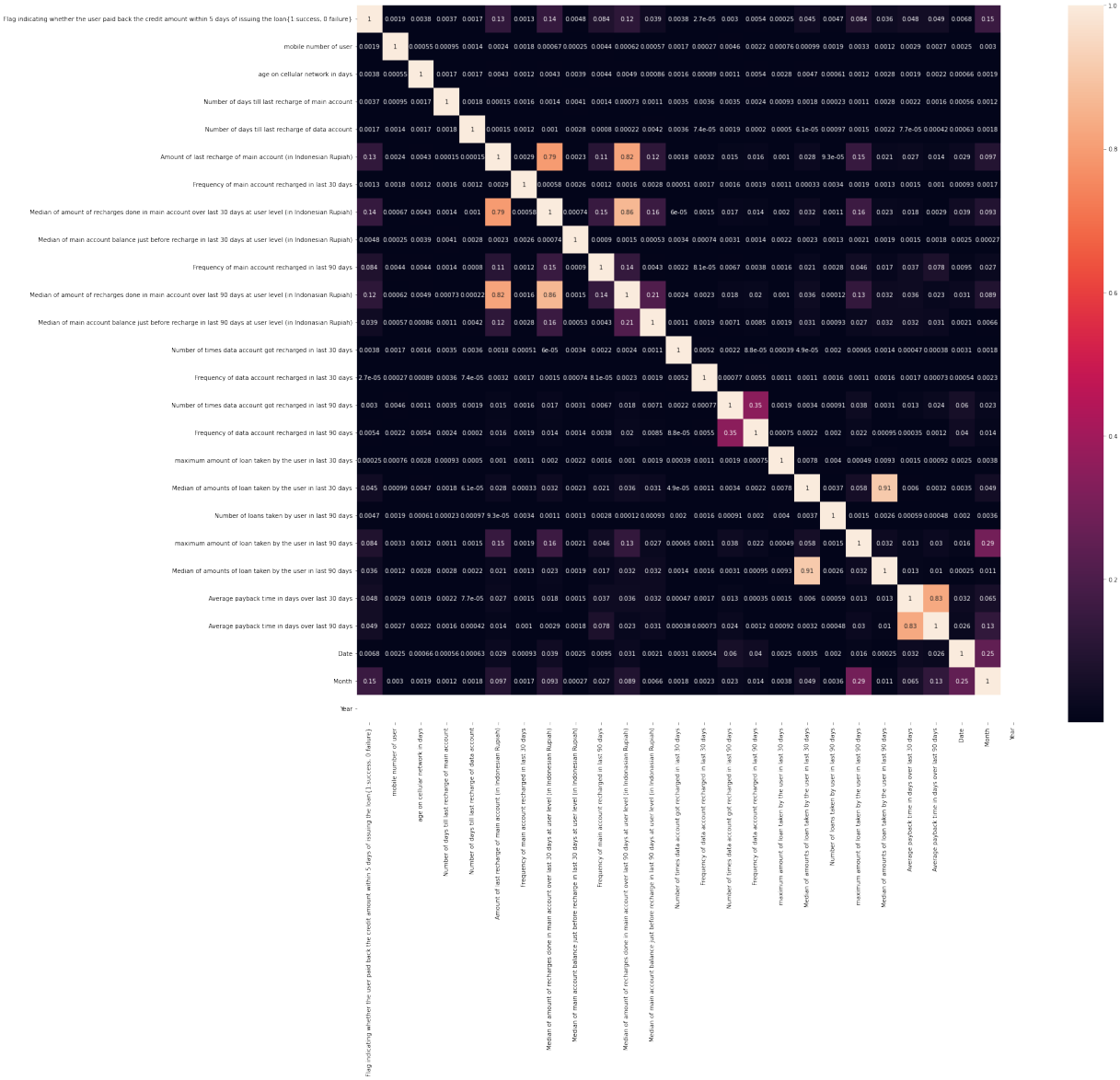
9. **Checking skewness:**

Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure} -2.270254
Year 0.000000
mobile number of user
0.018124
Date 0.199845
Month 0.343242
maximum amount of loan taken by the user in last 90 days
1.678304
Frequency of main account recharged in last 90 days
2.285423
Number of loans taken by user in last 30 days
2.713421
Total amount of loans taken by user in last 30 days
2.975719
Total amount of loans taken by user in last 90 days
3.150006

Number of times main account got recharged in last 30 days
3.283842
Number of times main account got recharged in last 90 days
3.425254
Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah) 3.512324
Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah) 3.752706
Amount of last recharge of main account (in Indonesian Rupiah)
3.781149
Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah) 3.946230
Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah) 4.252565
Average main account balance over last 90 days
4.437681
Average main account balance over last 30 days
4.521929
Median of amounts of loan taken by the user in last 30 days
4.551043
Median of amounts of loan taken by the user in last 90 days
4.895720
Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)
4.897950
Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)
6.386787
Average payback time in days over last 90 days
6.899951
Average payback time in days over last 30 days
8.310695
age on cellular network in days
10.392949
Frequency of main account recharged in last 30 days
14.772833
Frequency of data account recharged in last 30 days
14.776430
Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah) 14.779875
Number of days till last recharge of main account
14.790974

Number of days till last recharge of data account
14.814857
Number of loans taken by user in last 90 days
16.594408
maximum amount of loan taken by the user in last 30 days
17.658052
Number of times data account got recharged in last 30 days
17.818364
Number of times data account got recharged in last 90 days
27.267278
Frequency of data account recharged in last 90 days
28.988083
Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah) 44.8805

10.Heatmap:



SYNTAX

```
df_corr = df.corr().abs()
plt.figure(figsize=(25,22))
sns.heatmap(df_corr,annot=True,annot_kws={'size':10})
plt.show()
```

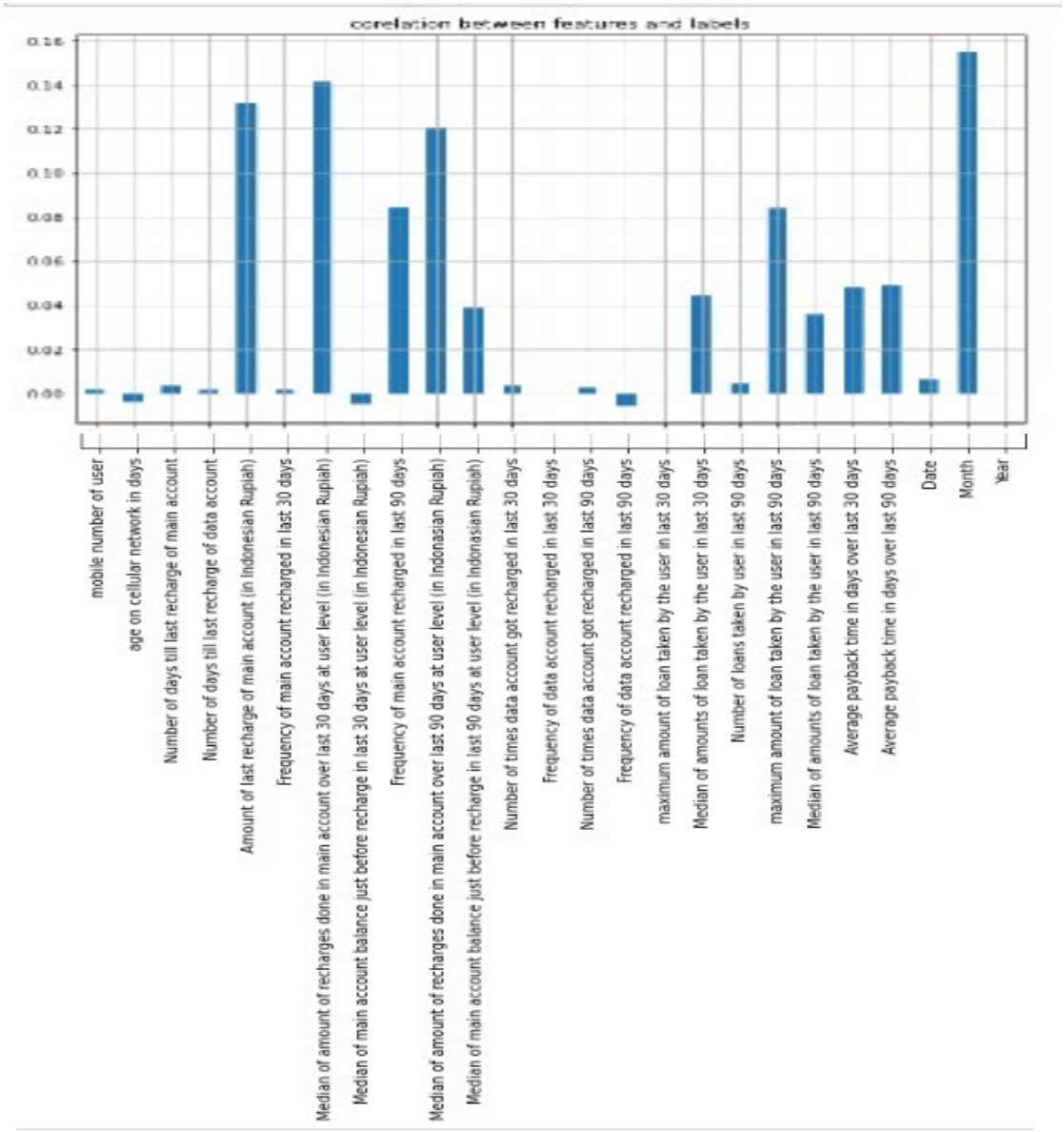
11. Variance inflation factor:

SYNTAX

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()

vif['vif'] = [variance_inflation_factor(df[cont_columns],i) for i in
range(df[cont_columns].shape[1])]

vif['features'] = df[cont_columns].columns

vif
```

	vif	features
17	5.991523	Median of amounts of loan taken by the user in...
20	5.970883	Median of amounts of loan taken by the user in...
10	5.033922	Median of amount of recharges done in main acc...
7	4.426496	Median of amount of recharges done in main acc...
5	3.367173	Amount of last recharge of main account (in In...
22	3.320329	Average payback time in days over last 90 days
21	3.256046	Average payback time in days over last 30 days
24	1.248220	Month
14	1.144044	Number of times data account got recharged in ...
15	1.139870	Frequency of data account recharged in last 90...
19	1.135111	maximum amount of loan taken by the user in la...
23	1.092159	Date
11	1.056524	Median of main account balance just before rec...
0	1.054602	Flag indicating whether the user paid back the...
9	1.046814	Frequency of main account recharged in last 90...
2	1.000174	age on cellular network in days
16	1.000156	maximum amount of loan taken by the user in la...
12	1.000154	Number of times data account got recharged in ...
1	1.000143	mobile number of user
8	1.000125	Median of main account balance just before rec...
18	1.000124	Number of loans taken by user in last 90 days
13	1.000121	Frequency of data account recharged in last 30...
6	1.000119	Frequency of main account recharged in last 30...
3	1.000107	Number of days till last recharge of main account
4	1.000102	Number of days till last recharge of data account
25	0.000000	Year

12. Feature Selection/Finding best features:




```
x = vif.sort_values(by='vif',ascending=False).T
x.columns
```

```
hc_list = df.columns[[4,  3, 25, 24, 15, 17, 10,  6,  5, 12, 29,]]
```

```
df.drop(columns = hc_list ,axis = 1, inplace=True)
```

```
# Splitting data into features and label:

y = df[df.columns[0]]
X = df.drop(df.columns[0],axis=1)
```

```
y
```

```
0         0
1         1
2         1
3         1
4         1
         ..
209588    1
209589    1
209590    1
209591    1
209592    1
Name: Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}, Len
gth: 209593, dtype: int64
```

```
X_Dums = pd.get_dummies(X)
```

13. Power Transform and PCA:

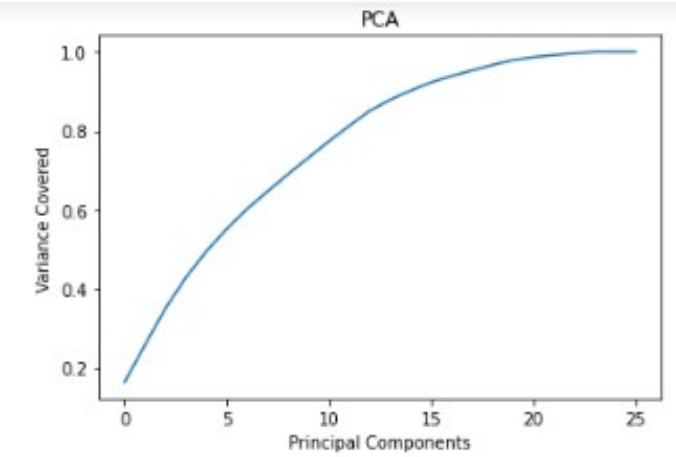
SYNTAX

from sklearn.preprocessing import PowerTransformer

pt = PowerTransformer()

X_cont_trans = pt.fit_transform(Xcont_data)

PCA: For numerical columns in features:




```
from sklearn.decomposition import PCA

pcaf = PCA(n_components=25)
new_pcompf = pcaf.fit_transform(X_trans)
princi_compf = pd.DataFrame(new_pcompf)
princi_compf
```

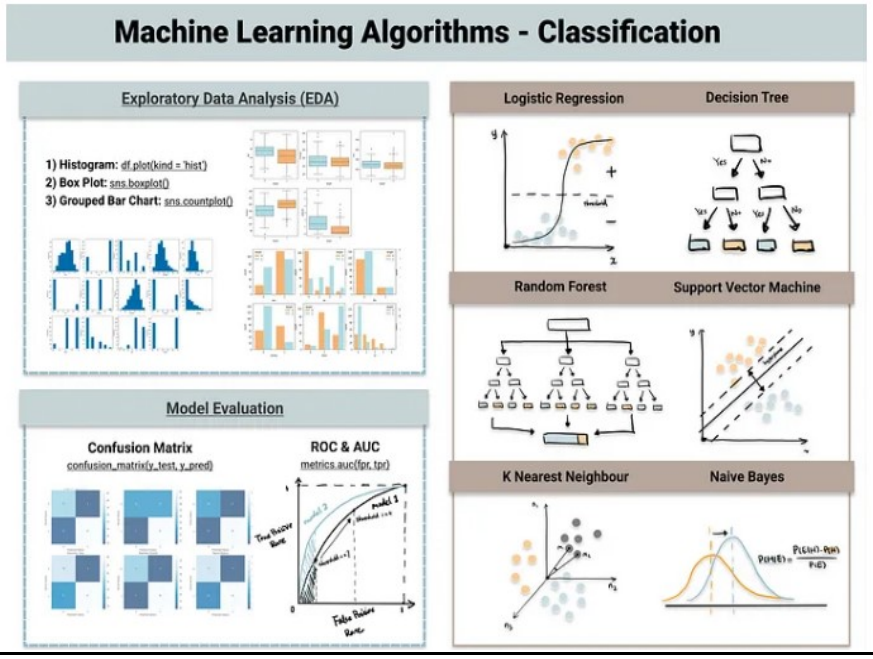
14. Data Splitting and Treating imbalanced:

```
# Splitting our data to training data and testing data
# x_train,x_test,y_train,y_test

X_train,X_test,y_train,y_test = train_test_split(princi_compf,y,test_size=0.20,random_state=1)

# Here we are keeping training data as our scalled data and testing data as our label or target.
```

```
from imblearn.over_sampling import SMOTE

print("Before OverSampling, counts of label '1': {}".format(sum(y_train==1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train==0)))

sm = SMOTE(random_state=2)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())

print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))

print("After OverSampling, counts of label '1': {}".format(sum(y_train_res==1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res==0)))
```

```
Before OverSampling, counts of label '1': 146692
Before OverSampling, counts of label '0': 20982

After OverSampling, the shape of train_X: (293384, 25)
After OverSampling, the shape of train_y: (293384,)

After OverSampling, counts of label '1': 146692
After OverSampling, counts of label '0': 146692
```

ALGORITHMS USED TO PREDICT PRICE OF HOUSES:

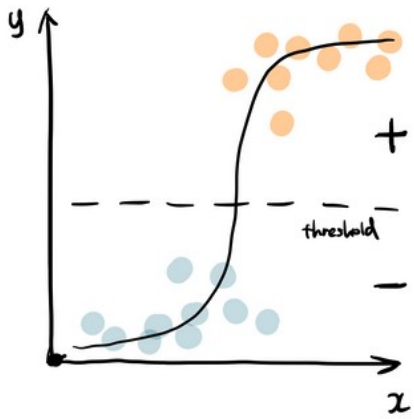


- Logistic Regression
- Decission Tree classifier

- Random Forest classifier (with Hyperparameter tuning)
- K-Neighbours classifier (with Hyperparameter tuning)
- Support Vector classifier
- AdaBoost classifier (with Hyperparameter tuning)

Run and Evaluating selected models

1. Logistic Regression:



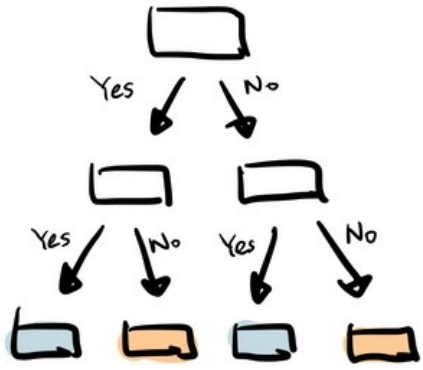
Logistics regression uses sigmoid function above to return the probability of a label. It is widely used when the classification problem is binary — true or false, win or lose, positive or negative.

The sigmoid function generates a probability output. By comparing the probability with a pre-defined threshold, the object is assigned to a label accordingly.

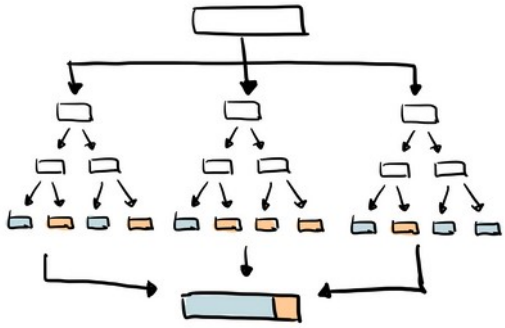
```
# Logistic Regression: in case of oversampling

log_reg = LogisticRegression(random_state=1)

log_reg.fit(X_train_res,y_train_res)

pred_train = log_reg.predict(X_train_res)

y_pred = log_reg.predict(X_test)

acc = accuracy_score(y_test,y_pred)

print('Training accuracy: ', accuracy_score(y_train_res,pred_train)*100)
print('Testing accuracy: ', acc*100)

confusion_mat = confusion_matrix(y_test,y_pred)

print('Confusion matrix: \n',confusion_mat)

clr = classification_report(y_test,y_pred)

print('classification report: ' ,clr)
```

```
Training accuracy:  73.97540424835711
Testing accuracy:  74.27180991912975
Confusion matrix:
 [[ 3853  1327]
 [ 9458 27281]]
classification report:                precision    recall  f1-score   support

           0       0.29      0.74      0.42      5180
           1       0.95      0.74      0.83     36739

    accuracy                           0.74     41919
   macro avg       0.62      0.74      0.63     41919
weighted avg       0.87      0.74      0.78     41919
```

2. Decision Tree and Random Forest Models:



Decision tree builds tree branches in a hierarchy approach and each branch can be considered as an if-else statement. The branches develop by partitioning the dataset into subsets based on most important features. Final classification happens at the leaves of the decision tree.



Random forest is a collection of decision trees. It is a common type of ensemble methods which aggregate results from multiple predictors. Random forest additionally utilizes bagging technique that allows each tree trained on a random sampling of original dataset and takes the majority vote from trees. Compared to decision tree, it has better generalization but less interpretable, because of more layers added to the model.

```
: # Decision Tree Classifier:

dtc = DecisionTreeClassifier(random_state=1)
dtc.fit(X_train_res,y_train_res)
pred_train_dtc = dtc.predict(X_train_res)
y_pred = dtc.predict(X_test)
acc = accuracy_score(y_test,y_pred)
print('Training accuracy: ', accuracy_score(y_train_res,pred_train_dtc)*100)
print('Testing accuracy: ', acc*100)
confusion_mat = confusion_matrix(y_test,y_pred)
print('Confusion matrix: \n',confusion_mat)
clr = classification_report(y_test,y_pred)
print('classification report: ' ,clr)
```

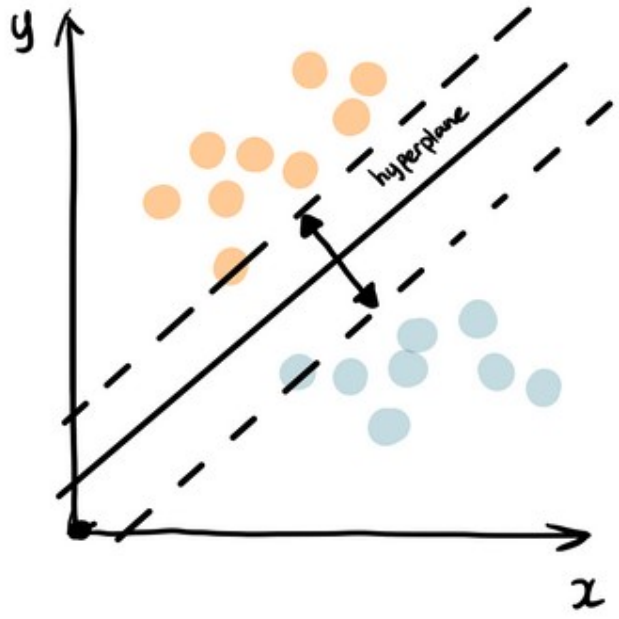
```
Training accuracy:  99.99931829956644
Testing accuracy:  80.82730981177986
Confusion matrix:
 [[ 2742  2438]
 [ 5599 31140]]
classification report:               precision    recall  f1-score   support

           0       0.33      0.53      0.41      5180
           1       0.93      0.85      0.89     36739

    accuracy                           0.81     41919
   macro avg       0.63      0.69      0.65     41919
weighted avg       0.85      0.81      0.83     41919
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
pred_train_rfc_f = rfc_f.predict(X_train_res)
y_pred = rfc_f.predict(X_test)
acc = accuracy_score(y_test,y_pred)
print('Training accuracy: ', accuracy_score(y_train_res,pred_train_rfc_f)*100)
print('Testing accuracy: ', acc*100)
confusion_mat = confusion_matrix(y_test,y_pred)
print('Confusion matrix: \n',confusion_mat)
clr = classification_report(y_test,y_pred)
print('classification report: ' ,clr)
```

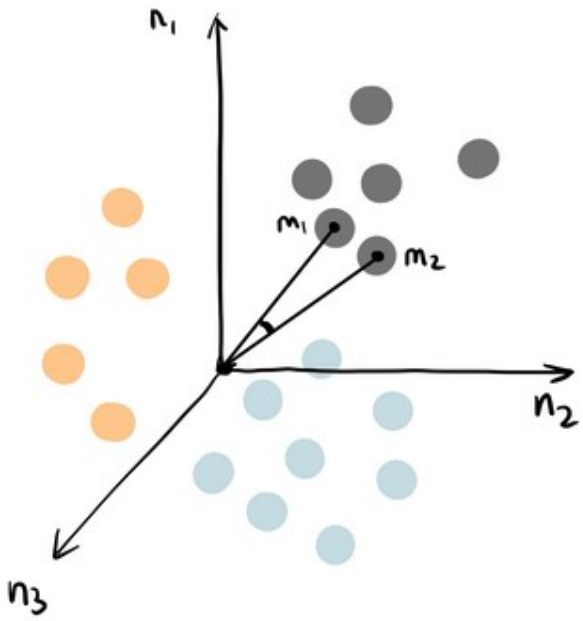
```
Training accuracy:  99.99795489869932
Testing accuracy:  86.05405663303037
Confusion matrix:
 [[ 3083  2097]
 [ 3749 32990]]
classification report:               precision    recall  f1-score   support

           0       0.45      0.60      0.51      5180
           1       0.94      0.90      0.92     36739

    accuracy                           0.86     41919
   macro avg       0.70      0.75      0.72     41919
weighted avg       0.88      0.86      0.87     41919
```

3. Support Vector machine and K-Nearest Neighbour:



Support vector machine finds the best way to classify the data based on the position in relation to a border between positive class and negative class. This border is known as the hyperplane which maximize the distance between data points from different classes. Similar to decision tree and random forest, support vector machine can be

used in both classification and regression, SVC (support vector classifier) is for classification problem.

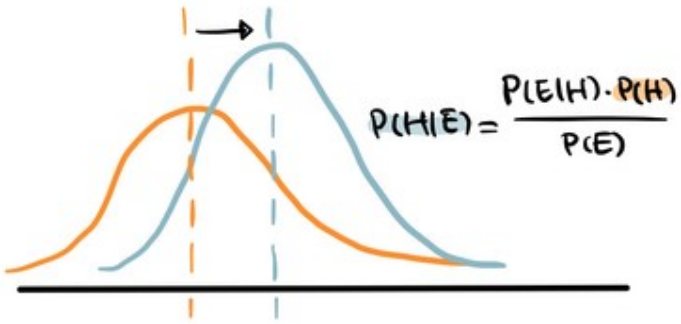


k nearest neighbour algorithm as representing each data point in a n dimensional space — which is defined by n features. And it calculates the distance between one point to another, then assign the label of unobserved data based on the labels of nearest observed data points. KNN can also be used for building recommendation system.

```
# Using best parameters for improved score:

svc = SVC()

svc.fit(X_train_res,y_train_res)

pred_train_svc = svc.predict(X_train_res)
y_pred = svc.predict(X_test)
acc = accuracy_score(y_test,y_pred)
print('Training accuracy: ', accuracy_score(y_train_res,pred_train_svc)*100)
print('Testing accuracy: ', acc*100)
confusion_mat = confusion_matrix(y_test,y_pred)
print('Confusion matrix: \n',confusion_mat)
clr = classification_report(y_test,y_pred)
print('classification report: ' ,clr)
```

```
# KNN classifier:

knn = KNeighborsClassifier()
knn.fit(X_train_res,y_train_res)
pred_train_knn = knn.predict(X_train_res)
y_pred = knn.predict(X_test)
acc = accuracy_score(y_test,y_pred)
print('Training accuracy: ', accuracy_score(y_train_res,pred_train_knn)*100)
print('Testing accuracy: ', acc*100)
```

4. Naïve Bayes:



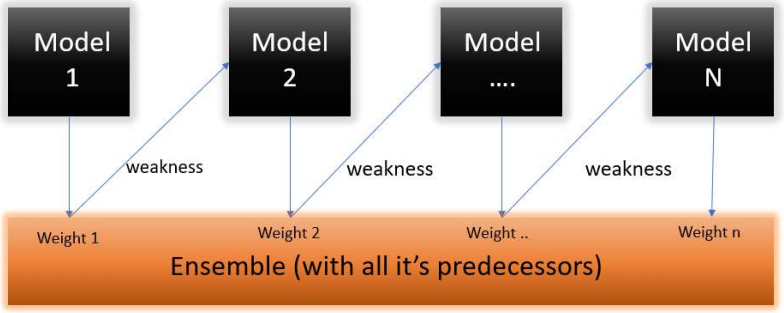
Naive Bayes is based on Bayes' Theorem — an approach to calculate conditional probability based on prior knowledge, and the naive assumption that each feature is independent to each other. The biggest advantage of Naive Bayes is that, while most machine learning algorithms rely on large amount of training data, it performs relatively well even when the training data size is small. Gaussian Naive Bayes is a type of Naive Bayes classifier that follows the normal distribution.

```
# training a Naive Bayes classifier
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB().fit(X_train_res, y_train_res)
gnb_predictions = gnb.predict(X_test)

# accuracy on X_test
accuracy = gnb.score(X_test, y_test)
print(accuracy)
```

```
0.46948528161454423
```

5. AdaBoost Regressor:

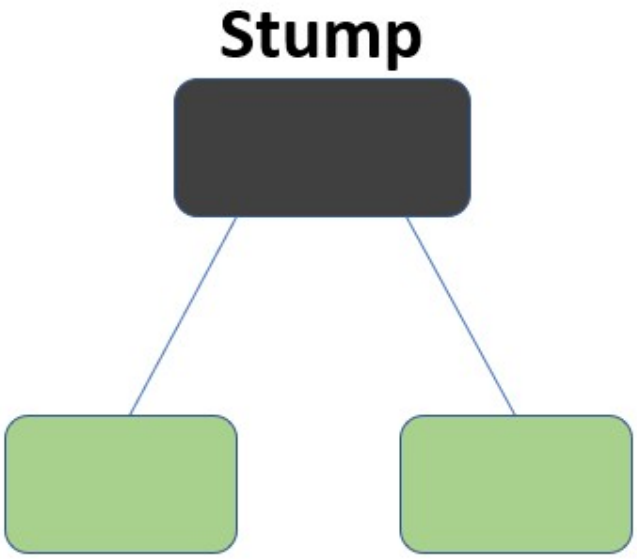
AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are also called **Decision Stumps.**



Boosting is an ensemble modelling technique that was first presented by Freund and Schapire in the year 1997, since then, Boosting has been a prevalent technique for tackling binary classification problems. These algorithms improve the prediction power by **converting a number of weak learners to strong learners.**

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision

trees with one level that means with Decision trees with only 1 split. These trees are also called **Decision Stumps.**



What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lowe error is received.

```
# ADA model:
from sklearn.ensemble import AdaBoostClassifier

ada = AdaBoostClassifier()
ada.fit(X_train_res,y_train_res)
yb_pred = ada.predict(X_test)
pred_train_ada = ada.predict(X_train_res)
y_pred = ada.predict(X_test)
acc = accuracy_score(y_test,y_pred)
print('Training accuracy: ', accuracy_score(y_train_res,pred_train_ada)*100)
print('Testing accuracy: ', acc*100)
confusion_mat = confusion_matrix(y_test,y_pred)
print('Confusion matrix: \n',confusion_mat)
clr = classification_report(y_test,y_pred)
print('classification report: ' ,clr)
```

```
Training accuracy:  78.72447031876312
Testing accuracy:  78.56580548200101
Confusion matrix:
 [[ 4026  1154]
 [ 7831 28908]]
classification report:               precision    recall  f1-score   support

           0       0.34      0.78      0.47      5180
           1       0.96      0.79      0.87     36739

    accuracy                           0.79     41919
   macro avg       0.65      0.78      0.67     41919
weighted avg       0.88      0.79      0.82     41919
```

6. XG Boost:

XGBoost is an implementation of Gradient Boosted decision trees. XGBoost models majorly dominate in many Kaggle Competitions.

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

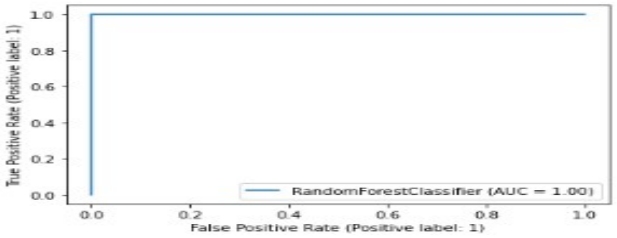
```
from sklearn.ensemble import GradientBoostingClassifier

# GBC

gbdt_clf = GradientBoostingClassifier(random_state=1)
gbdt_clf.fit(X_train_res,y_train_res)
pred_train_gbdt_clf = gbdt_clf.predict(X_train_res)
y_pred = gbdt_clf.predict(X_test)
acc = accuracy_score(y_test,y_pred)
print('Training accuracy: ', accuracy_score(y_train_res,pred_train_gbdt_clf)*100)
print('Testing accuracy: ', acc*100)
confusion_mat = confusion_matrix(y_test,y_pred)
print('Confusion matrix: \n',confusion_mat)
clr = classification_report(y_test,y_pred)
print('classification report: ' ,clr)
```

```
Training accuracy:  80.83365146020233
Testing accuracy:  80.4026813616737
Confusion matrix:
 [[ 4134  1046]
 [ 7169 29570]]
classification report:               precision    recall  f1-score   support

           0       0.37      0.80      0.50      5180
           1       0.97      0.80      0.88     36739

    accuracy                           0.80     41919
   macro avg       0.67      0.80      0.69     41919
weighted avg       0.89      0.80      0.83     41919
```

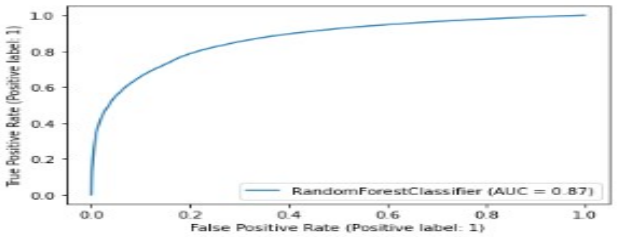
Evaluation Metrics Used

```
from sklearn.metrics import roc_curve,roc_auc_score
from sklearn.metrics import plot_roc_curve
```

```
disp = plot_roc_curve(rfc_f,X_train_res,y_train_res)
```




```
disp = plot_roc_curve(rfc_f,X_test,y_test)
```



Results/Observations

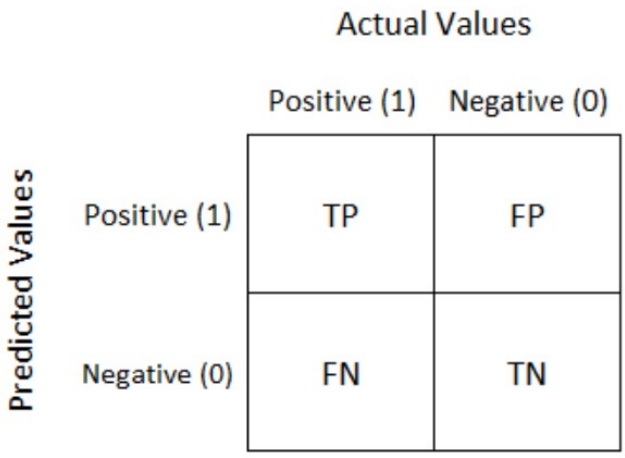
Key Metrics for success

Metrics Used:

Accuracy

Accuracy simply measures how often the classifier correctly predicts. We can define accuracy as the ratio of the number of correct predictions and the total number of predictions.

Confusion Matrix



Confusion Matrix is a performance measurement for the machine learning classification problems where the output can be two or more classes. It is a table with combinations of predicted and actual values.

A confusion matrix is defined as thetable that is often used to describe the performance of a classification model on a set of the test data for which the true values are known.

True Positive: We predicted positive and it's true. In the image, we predicted that a woman is pregnant and she actually is.

True Negative: We predicted negative and it's true. In the image, we predicted that a man is not pregnant and he actually is not.

False Positive (Type 1 Error)- We predicted positive and it's false. In the image, we predicted that a man is pregnant but he actually is not.

False Negative (Type 2 Error)- We predicted negative and it's false. In the image, we predicted that a woman is not pregnant but she actually is.

Precision — Precision explains how many of the correctly predicted cases actually turned out to be positive. Precision is useful in the cases where False Positive is a higher concern than False Negatives. The importance of *Precision is in music or video recommendation systems, e-commerce websites, etc. where wrong results could lead to customer churn and this could be harmful to the business.*

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

2. **Recall (Sensitivity) —** Recall explains how many of the actual positive cases we were able to predict correctly with our model. It is a useful metric in cases where False Negative is of higher concern than False Positive. It *is important in medical cases where it doesn't matter whether we raise a false alarm but the actual positive cases should not go undetected!*

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

F1 Score — It gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall.

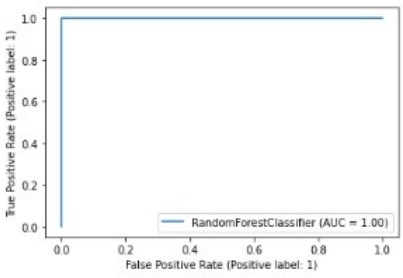
$$F1 = 2.\frac{Precision \times Recall}{Precision + Recall}$$

AUC-ROC — The Receiver Operator Characteristic (ROC) is a probability curve that plots the TPR(True Positive Rate) against the FPR(False Positive Rate) at various threshold values and separates the 'signal' from the 'noise'.

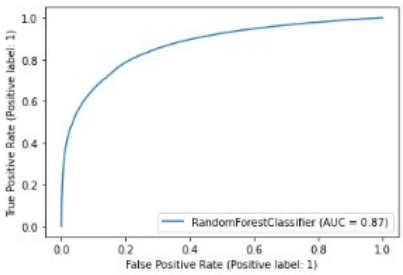
ROC AUC CURVE:

```
# Lets plot ROC AUC curve to choose best model:

from sklearn.metrics import roc_curve,roc_auc_score
from sklearn.metrics import plot_roc_curve
```

```
disp = plot_roc_curve(rfc_f,X_train_res,y_train_res)
```



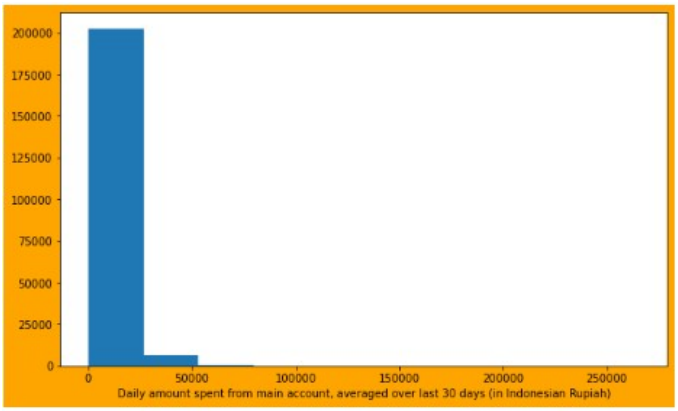

```
disp = plot_roc_curve(rfc_f,X_test,y_test)
```

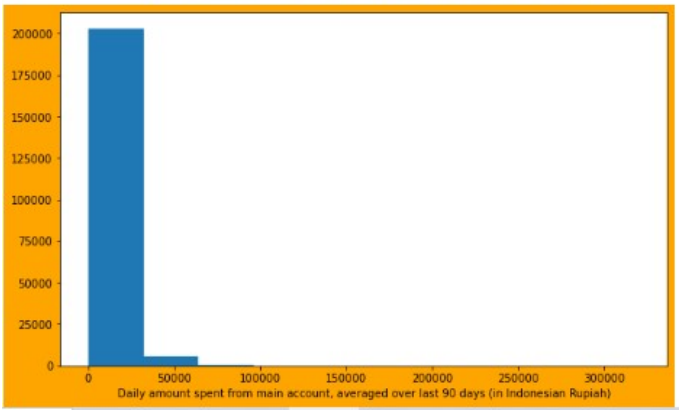


Visualizations

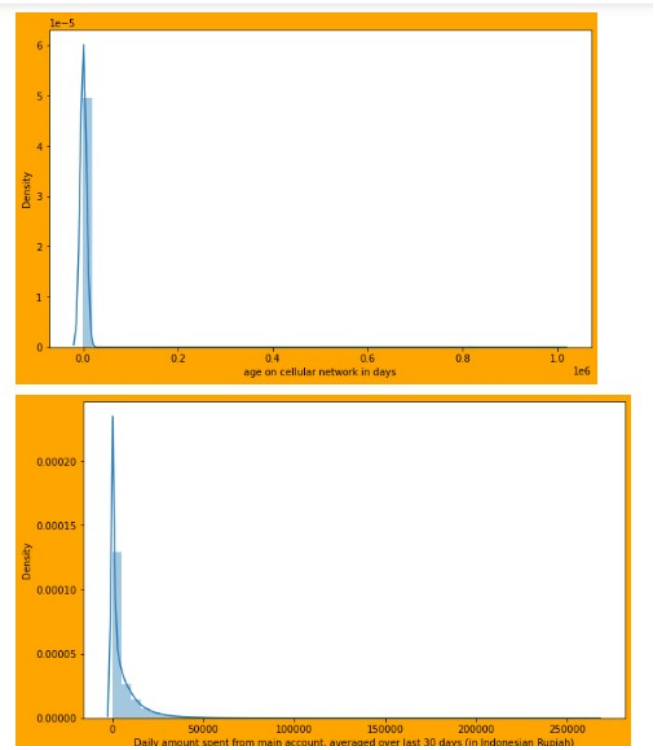
Plots used are:

- Univariate analysis: Frequency analysis plots:
  1. Histogram plot.

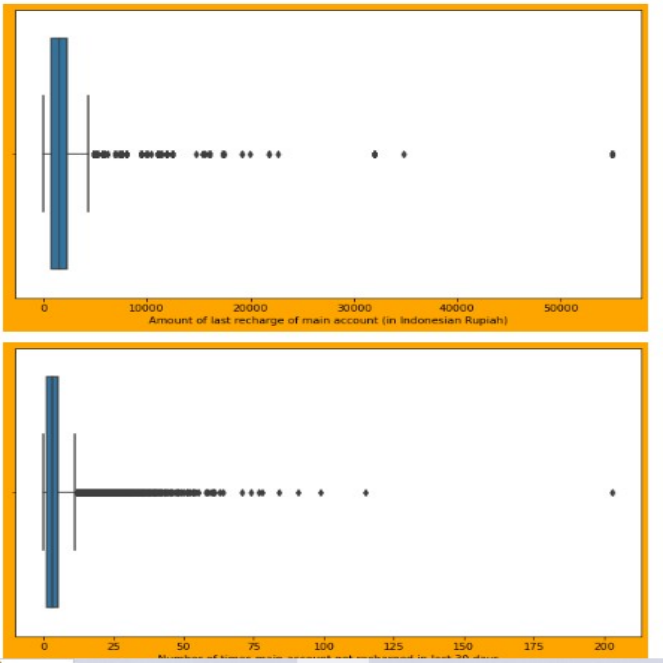




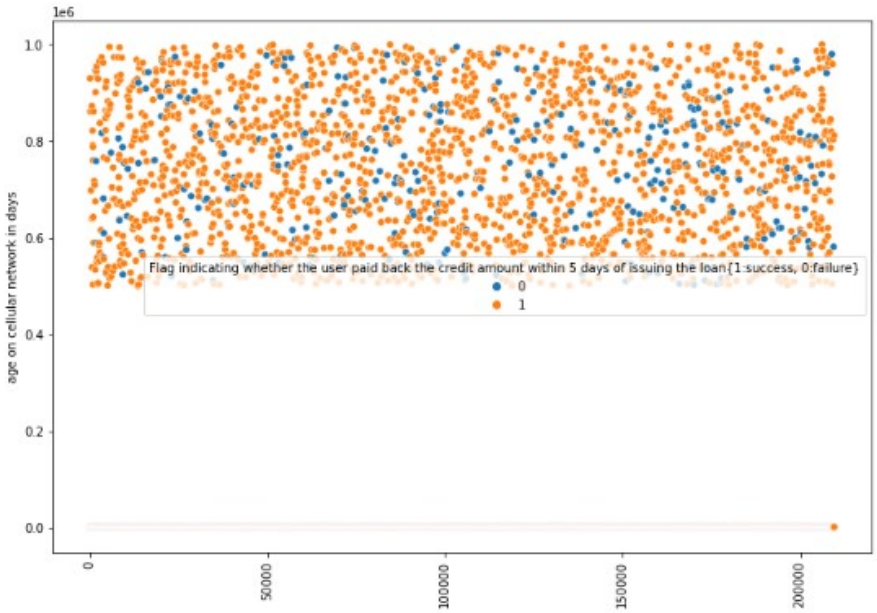
2. Distribution plot



- Outlier presence analysis:
  1. Boxplots



- Bivariate Analysis
  1. Scatterplot



  2. Catplot

```
for i in cat_columns:
    plt.figure(figsize=(12,50))
    sns.catplot(x=i, kind="count",hue = df.columns[0], data=df)
    plt.xticks(rotation=90)
    plt.show()
```

```
<Figure size 864x3600 with 0 Axes>
```

- Multivariate Analysis:
  1. Pairplot
  2. HeatMap (refer previous section)
  3. VIF analysis (refer previous section)
  4. Correlation coefficient analysis (refer previous section)

Inferences drawn from plots:

- Dataset is imbalanced.
- Outliers are present that need to be treated.
- Skewness is present in dataset.
- Multicollinearity problem exist.

Results

Grouped data can be used for better understanding mindset of loan defaulters:

Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}	0	1
mobile number of user	2140870789	7646270374
age on cellular network in days	272.0	712.0
Number of days till last recharge of main account	2.0	20.0
Number of days till last recharge of data account	0.0	0.0
Amount of last recharge of main account (in Indonesian Rupiah)	1539	5787
Frequency of main account recharged in last 30 days	21.0	0.0
Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)	1539.0	5787.0
Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)	7.5	61.04
Frequency of main account recharged in last 90 days	21	0
Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)	1539.0	5787.0
Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)	7.5	61.04
Number of times data account got recharged in last 30 days	0.0	0.0
Frequency of data account recharged in last 30 days	0.0	0.0
Number of times data account got recharged in last 90 days	0	0
Frequency of data account recharged in last 90 days	0	0
maximum amount of loan taken by the user in last 30 days	6.0	12.0
Median of amounts of loan taken by the user in last 30 days	0.0	0.0
Number of loans taken by user in last 90 days	2.0	1.0
maximum amount of loan taken by the user in last 90 days	6	12
Median of amounts of loan taken by the user in last 90 days	0.0	0.0
Average payback time in days over last 30 days	29.0	0.0
Average payback time in days over last 90 days	29.0	0.0
telecom circle	UPW	UPW
Date	20	10
Month	7	8
Year	2016	2016

Column values with ‘0 label’ indicate that defaulters have large differences in values who paid repaid loan on time.

Best algorithms are Random Forrest and XG Boost classifier.

FINAL RESULT

Training accuracy: 99.99795489869932
Testing accuracy: 86.05405663303037
Confusion matrix:
[[3083 2097]
[3749 32990]]
classification report:

	precision	recall	f1-score	support
0	0.45	0.60	0.51	5180
1	0.94	0.90	0.92	36739
accuracy			0.86	41919
macro avg	0.70	0.75	0.72	41919
weighted avg	0.88	0.86	0.87	41919

Limitations of this work and Scope for Future Work

1. Other modelling approaches could have been utilized.
2. Intense hyperparameter tuning could have been performed.

With improvement this model can be deployed to predict loan defaulters.