Micro Credit Defaulter

Submitted By

RAJASEKAR

TABLE OF CONTENTS

INTRODUCTION

1. What is Micro-Finance?

The Micro Finance is defined as ***"the provision of Financial assistance and Insurance Services to an individual or eligible client either directly or through a group mechanism for an amount , not exceeding the minimal amount fixed by the financial sector per individual for small and tiny enterprises, agriculture allied activities or other prescribed purposes.*** Those who were eligible for the credit should fall under the criteria prescribed by the financial institution.

The Financial Intuitions which provides such services are widely known as **Microfinance Institution (MFI).** And the services they provide are often known as **Microfinance services (MFS).**

2. Micro Finance Institutions and Telecom Service

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on. Many microfinance institutions (MFI), experts and donors are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient, and cost saving, than the traditional high-touch model used since long for the purpose of delivering microfinance services. Though, the MFI industry is primarily focusing on low income families and is very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes. 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.

3. **Business Problem – Micro credit loan defaulters**

The MFS 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).

4. **Review of Literature**

The MFS are helping people stay in touch with their loved ones irrespective of their financial conditions. Thus MFI provide MFS for the people in need. 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.

5. **Future of Micro Finance**

Although the MFI are providing MFS, some critical reasons make it hard for the MFI to continue in funding the MFS. The increase in the defaulters is one of the main reasons for the MFI to face such hardships. Thus the only way to maintain the current MFS programme is to reduce the defaulter's rate. For this we have to know the probability of the customer paying the

loan back. For that we have to find the history of the customer's account in recharging, repaying the loan etc.

Here the main objective for this project is to find the defaulters by using technology services. So we will be using Machine Learning Models for doing the same.

ANALYTICS OF THE BUSINESS PROBLEM

1. **What is Analytical problem framing?**

   Analytic problem framing involves translating the business problem into terms that can be addressed analytically via data and modelling. It's at this stage that you work backwards from the results / outputs you want to the data/inputs you're going to need, where you identify potential drivers and hypotheses to test, and where you nail down your assumptions. Analytic problem framing is the antithesis of merely working with the ready-to-hand data and seeing what comes of it, hoping for something insightful. Typically, the process moves on from here to data collection, cleansing and transformation, Methodology selection and model building, never to return. But if you're willing to borrow and use a concept from complex adaptive systems – maps and models – you can make repeat use of this stage to improve your overall outcome.

2. **Analytics of the business problem**

   As discussed above we need to build a ML Model to predict the repayment of the loan by the customer. For that we need some analytical data's. So we have to find the customer history with the telecom industry. The attributes should contain the recharge intervals , loan intervals, repayment intervals, date, area of the customer for forming a zone, time period taken for the customer to take the next loan etc.,

3. Hardware Requirements

A mid level computer that runs on Intel i3/i5/i7 or A10/A11/M1 or ryzen 3/5 or any other equivalent chipset and a suitable processor.

4. Software Requirements

Windows / Linux /Mac OS

5. Tools, Libraries and Packages used

Tool: 1.Anaconda Navigator
2. Jupyter Notebook

Libraries and Packages:

1. Numpy
2. Pandas
3. Matplotlib
4. Seaborn

6. Data sources

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.
The given data-set can be used to build a ML Model which can be used to predict in terms of a probability for each loan transaction, whether the

customer will be paying back the loaned amount within 5 days of insurance of loan. In this case, Label “1” indicates that the loan has been payed i.e. Non-defaulters, while, Label “0” indicates that the loan has not been payed i.e. defaulters.

The data description is stated below:

1. **Label** - indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}.
2. **msisdn** - mobile number of user.
3. **aon** - age on cellular network in days.
4. **daily_decr30** - Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah).
5. **daily_decr90** - Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah).
6. **rental30** - Average main account balance over last 30 days.
7. **rental90** - Average main account balance over last 90 days.
8. **last_rech_date_ma** - Number of days till last recharge of main account.
9. **Last rech date da** -Number of days till last recharge of data account.
10. **last_rech_amt_ma** - Amount of last recharge of main account (in Indonesian Rupiah)
11. **cnt_ma_rech30** - Number of times main account got recharged in last 30 days
12. **fr_ma_rech30** - Frequency of main account recharged in last 30 days
13. **sumamnt_ma_rech30** - Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)
14. **medianamnt_ma_rech30** - Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
15. **medianmarechprebal30** - Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
16. **cnt_ma_rech90** - Number of times main account got recharged in last 90 days
17. **fr_ma_rech90** - Frequency of main account recharged in last 90 days
18. **sumamnt_ma_rech90** - Total amount of recharge in main account over last 90 days (in Indonesian Rupiah)
19. **medianamnt_ma_rech90** - Median of amount of recharges done in main account over last 90 days at user level (in Indonesian Rupiah)
20. **medianmarechprebal90** - Median of main account balance just before recharge in last 90 days at user level (in Indonesian Rupiah)
21. **cnt_da_rech30** - Number of times data account got recharged in last 30 days
22. **fr_da_rech30** - Frequency of data account recharged in last 30 days.
23. **cnt_da_rech90** - Number of times data account got recharged in last 90 days

24. **fr_da_rech90** - Frequency of data account recharged in last 90 days
25. **cnt_loans30** - Number of loans taken by user in last 30 days
26. **amnt_loans30** - Total amount of loans taken by user in last 30 days
27. **maxamnt_loans30** - maximum amount of loan taken by the user in last 30 days There are only two options: 5 & 10Rs.,for which the user needs to pay back 6 & 12 Rs. respectively
28. **medianamnt_loans30** - Median of amounts of loan taken by the user in last 30 days
29. **cnt_loans90** - Number of loans taken by user in last 90 days
30. **amnt_loans90** - Total amount of loans taken by user in last 90 days
31. **maxamnt_loans90** - maximum amount of loan taken by the user in last 90 days
32. **medianamnt_loans90** - Median of amounts of loan taken by the user in last 90 days
33. **payback30** - Average payback time in days over last 30 days
34. **payback90** - Average payback time in days over last 90 days
35. **pcircle** - telecom circle
36. **pdate** – date

The shape of the data set is **209593 x 37** , that is data set consist of **209593 rows** and **37 columns** where the rows consist of the customer information and columns consist of the categories of data we needed to build the machine learning model.

7. Pre-Assumptions

As we look into the data –set, primarily we could infer that the data-set is pretty much well informed. It has all the required data's such **as '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'.**

Using these data's we could easily find the defaulters by the help of simple visualizations and ML models.

8. Data Pre- Processing

Here we will try to make the given data-set efficient for the ML model. For that we will carry out a series of process which will make the data-set perfect for building a ML model. This series of process includes,

1. **Collection of basic statistical data.**
2. **Exploratory data analysis.**
3. **Checking for correlation with target.**
4. **Checking for skewness in the data-set.**
5. **Checking for outliers.**

By following these processes we can achieve a more efficient data-set. We will use **Python** through **Jupyter notebook** for data processing. Also we will use Libraries such as **Pandas, Numpy for Analysis** and **Matplotlib, seaborn for visualization**.

8.1 Collection of basic statistical data

Now we will import the required libraries in to the Jupyter Notebook using Python codes.

```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
import warnings
warnings.filterwarnings("ignore")
```

We now have to load the data set into the Notebook.

```
#Loading the data-set
df=pd.read_csv("Data_file_micro.csv")
df
```

This will load the data-set in place and will display the same.

Since the data-set have been loaded we will now check for the presence of any null values and the data types.

```
#checking the null values
for col in df.columns:
    print("\n\nTitle :",col,"\nCol dtype :",df[col].dtype," \nNaN val: ",df[col].isnull().sum())
```

Title : Unnamed: 0	Title : rental90	Title : medianamnt_ma_rech30
Col dtype : int64	Col dtype : float64	Col dtype : float64
NaN val: 0	NaN val: 0	NaN val: 0
Title : label	Title : last_rech_date_ma	Title : medianmarechprebal30
Col dtype : int64	Col dtype : float64	Col dtype : float64
NaN val: 0	NaN val: 0	NaN val: 0
Title : msisdn	Title : last_rech_date_da	Title : cnt_ma_rech90
Col dtype : object	Col dtype : float64	Col dtype : int64
NaN val: 0	NaN val: 0	NaN val: 0
Title : aon	Title : last_rech_amt_ma	Title : fr_ma_rech90
Col dtype : float64	Col dtype : int64	Col dtype : int64
NaN val: 0	NaN val: 0	NaN val: 0
Title : daily_decr30	Title : cnt_ma_rech30	Title : sumamnt_ma_rech90
Col dtype : float64	Col dtype : int64	Col dtype : int64
NaN val: 0	NaN val: 0	NaN val: 0
Title : daily_decr90	Title : fr_ma_rech30	Title : medianamnt_ma_rech90
Col dtype : float64	Col dtype : float64	Col dtype : float64
NaN val: 0	NaN val: 0	NaN val: 0
Title : rental30	Title : sumamnt_ma_rech	Title : medianmarechprebal90
Col dtype : float64	Col dtype : float64	Col dtype : float64
NaN val: 0	NaN val: 0	NaN val: 0

Title : cnt_da_rech30	Title : medianamnt_loans30	Title : pcircle
Col dtype : float64	Col dtype : float64	Col dtype : object
NaN val: 0	NaN val: 0	NaN val: 0
Title : fr_da_rech30	Title : cnt_loans90	Title : pdate
Col dtype : float64	Col dtype : float64	Col dtype : object
NaN val: 0	NaN val: 0	NaN val: 0
Title : cnt_da_rech90	Title : amnt_loans90	
Col dtype : int64	Col dtype : int64	
NaN val: 0	NaN val: 0	
Title : fr_da_rech90	Title : maxamnt_loans90	
Col dtype : int64	Col dtype : int64	
NaN val: 0	NaN val: 0	
Title : cnt_loans30	Title : medianamnt_loans90	
Col dtype : int64	Col dtype : float64	
NaN val: 0	NaN val: 0	
Title : amnt_loans30	Title : payback30	
Col dtype : int64	Col dtype : float64	
NaN val: 0	NaN val: 0	
Title : maxamnt_loans	Title : payback90	
Col dtype : float64	Col dtype : float64	
NaN val: 0	NaN val: 0	

We could visibly see that there are no null values present in the data types. Also all the data types are either float or int, thus no data imputation will be needed for this data-set.

Now we select some of the columns to analyse deeper into the data-set.

```
df.shape
```

```
(209593, 37)
```

```
df["label"].unique()
```

```
array([0, 1], dtype=int64)
```

```
df["pcircle"].unique()
```

```
array(['UPW'], dtype=object)
```

```
df["pdate"].unique()
```

```
array(['2016-07-20', '2016-08-10', '2016-08-19', '2016-06-06',
       '2016-06-22', '2016-07-02', '2016-07-05', '2016-08-05',
       '2016-06-15', '2016-06-08', '2016-06-12', '2016-06-20',
       '2016-06-29', '2016-06-16', '2016-08-03', '2016-06-24',
       '2016-07-04', '2016-07-03', '2016-07-01', '2016-08-08',
       '2016-06-26', '2016-06-23', '2016-07-06', '2016-07-09',
       '2016-06-10', '2016-06-07', '2016-06-27', '2016-08-11',
       '2016-06-30', '2016-06-19', '2016-07-26', '2016-08-14',
       '2016-06-14', '2016-06-21', '2016-06-25', '2016-06-28',
       '2016-06-11', '2016-07-27', '2016-07-23', '2016-08-16',
       '2016-08-15', '2016-06-02', '2016-06-05', '2016-08-02',
       '2016-07-28', '2016-07-18', '2016-08-18', '2016-07-16',
       '2016-07-29', '2016-07-21', '2016-06-03', '2016-06-13',
       '2016-08-01', '2016-07-13', '2016-07-10', '2016-06-09',
       '2016-07-15', '2016-07-11', '2016-08-09', '2016-08-12',
       '2016-07-22', '2016-06-04', '2016-07-24', '2016-06-18',
       '2016-08-13', '2016-06-17', '2016-08-07', '2016-07-12',
       '2016-08-06', '2016-07-19', '2016-08-21', '2016-08-04',
       '2016-07-25', '2016-07-30', '2016-08-17', '2016-07-08',
       '2016-07-14', '2016-06-01', '2016-07-07', '2016-07-17',
       '2016-07-31', '2016-08-20'], dtype=object)
```

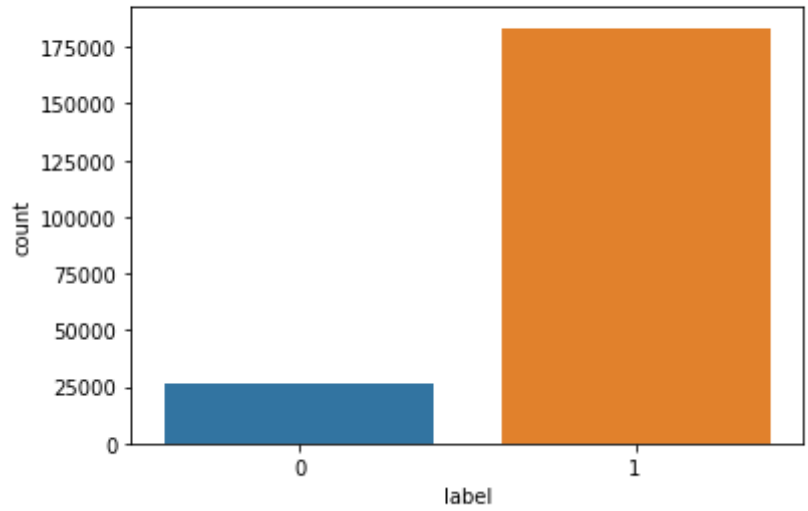
```
df["msisdn"].unique()
```

```
array(['21408I70789', '76462I70374', '17943I70372', ..., '22758I85348',
       '59712I82733', '65061I85339'], dtype=object)
```

Form the above analysis we can find that the attributes "Unnamed: 0","pcircle"(Telecom circle),"pdate (date)" AND "msisdn"(Mobile number of user) are not providing any viable information for our current objective. So we will drop them form the Data-set for further processing.

8.2 Exploratory Data Analysis

```
sn.countplot(x="label",data=df)
plt.show()
```



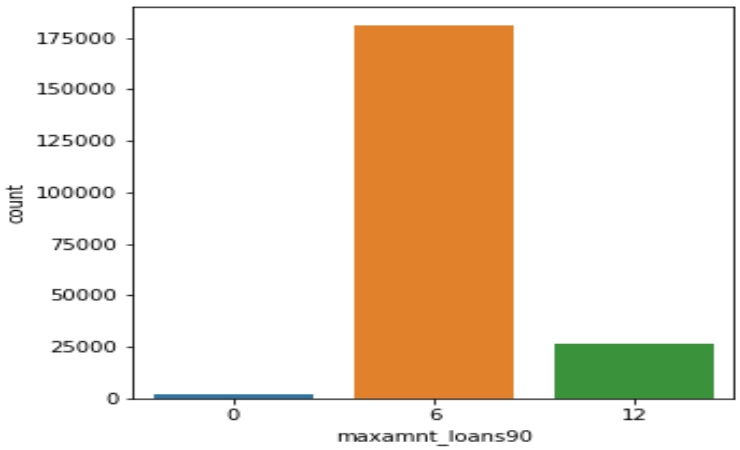
Observation:

1."0”: user failed to paid back the credit amount within 5 days & "1”: user paid back the credit amount within 5 days.

2. It seems the defaulters are lesser than the people repaying the loan amount.

```
df.groupby('maxamnt_loans90').size()
plt.figure(figsize=(5,5))
sn.countplot(df["maxamnt_loans90"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x15dd01faa00>
```



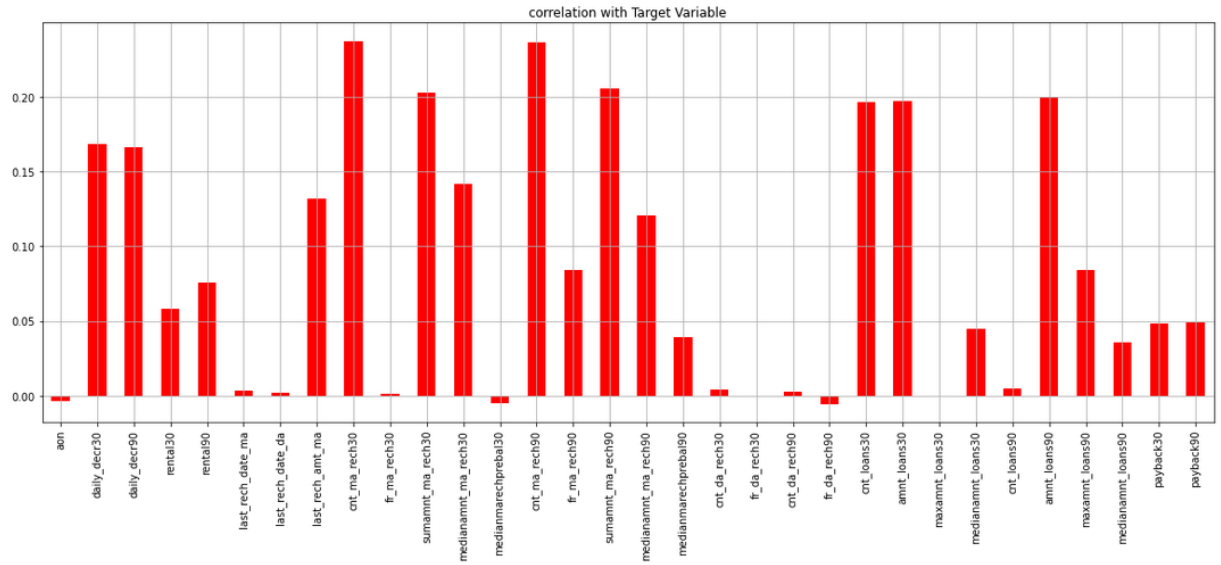
Observation:

1. People who took 5 Indo Rupiah loans are more than people who took 10 Indo Rupiah loans, on the other hand more than 98% of people who are using this telecom seems to have took this Micro finance loan.

8.3 Checking for correlation with target

```
#correlation with Target
plt.figure(figsize=(20,7))
df.drop("label",axis=1).corrwith(df["label"]).plot(kind="bar",grid=True,color="red")
plt.xticks(rotation="vertical")
plt.title("correlation with Target Variable")
```

```
Text(0.5, 1.0, 'correlation with Target Variable')
```



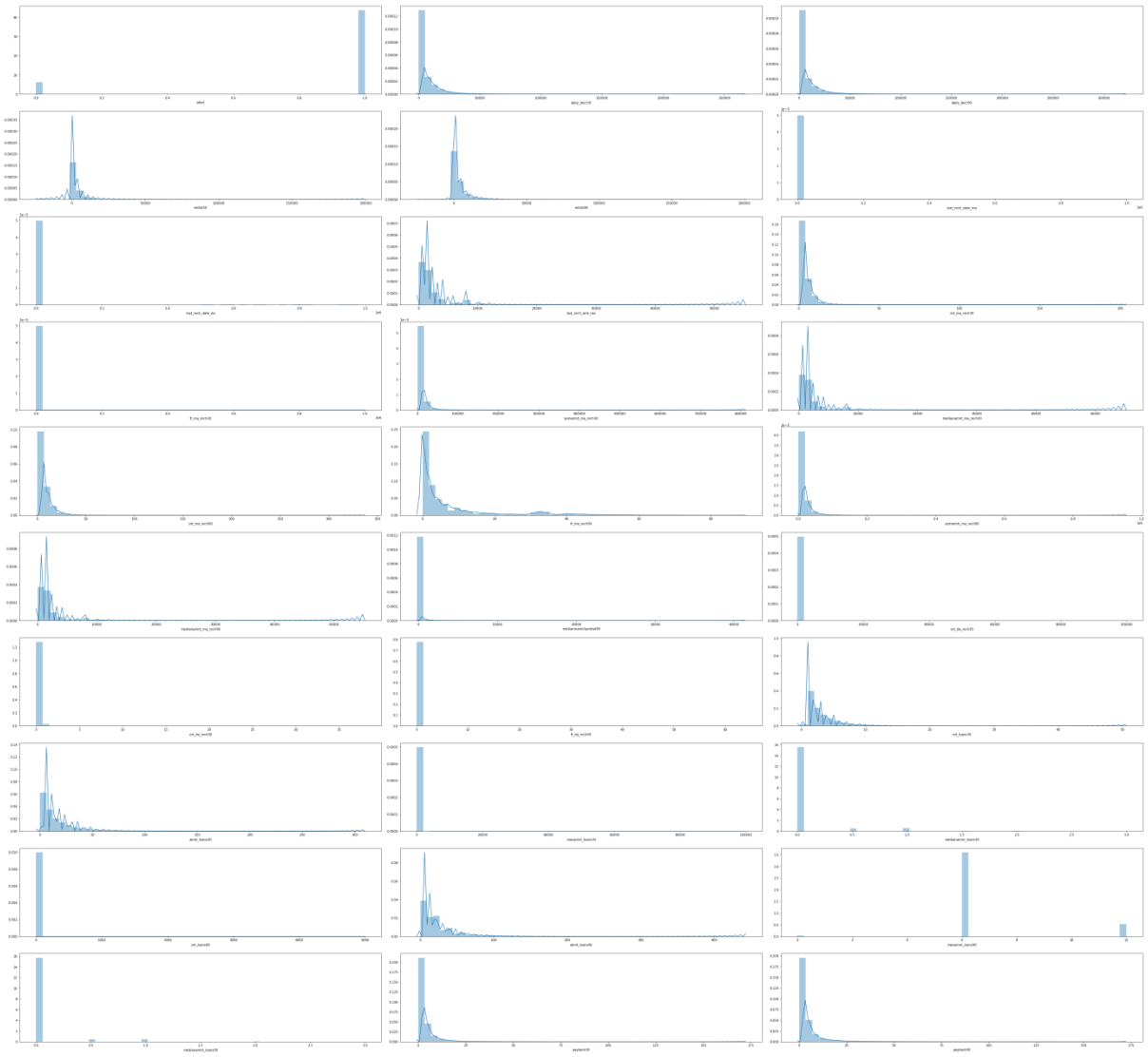
Observation:

Here, in this above chart we could see that the columns "aon","medianmarechprebal30", & "fr_da_rech30" seems to have a negative correlation with our Target variable. Thus dropping this column would give us much better Data-set to build a model.

```
#Dropping columns not correlated to our target
df.drop(columns=["aon","medianmarechprebal30","fr_da_rech30"],inplace=True)
```

8.4 Checking for Skewness

```
#Checking for skewness using "DISPLACEMENT ANALYSIS"
collist=df.columns.values
ncol=3
nrow=11
plt.figure(figsize=(50,50))
for i in range(0,len(collist)):
    plt.subplot(nrow,ncol,i+1)
    sn.distplot(df[collist[i]])
    plt.tight_layout()
```

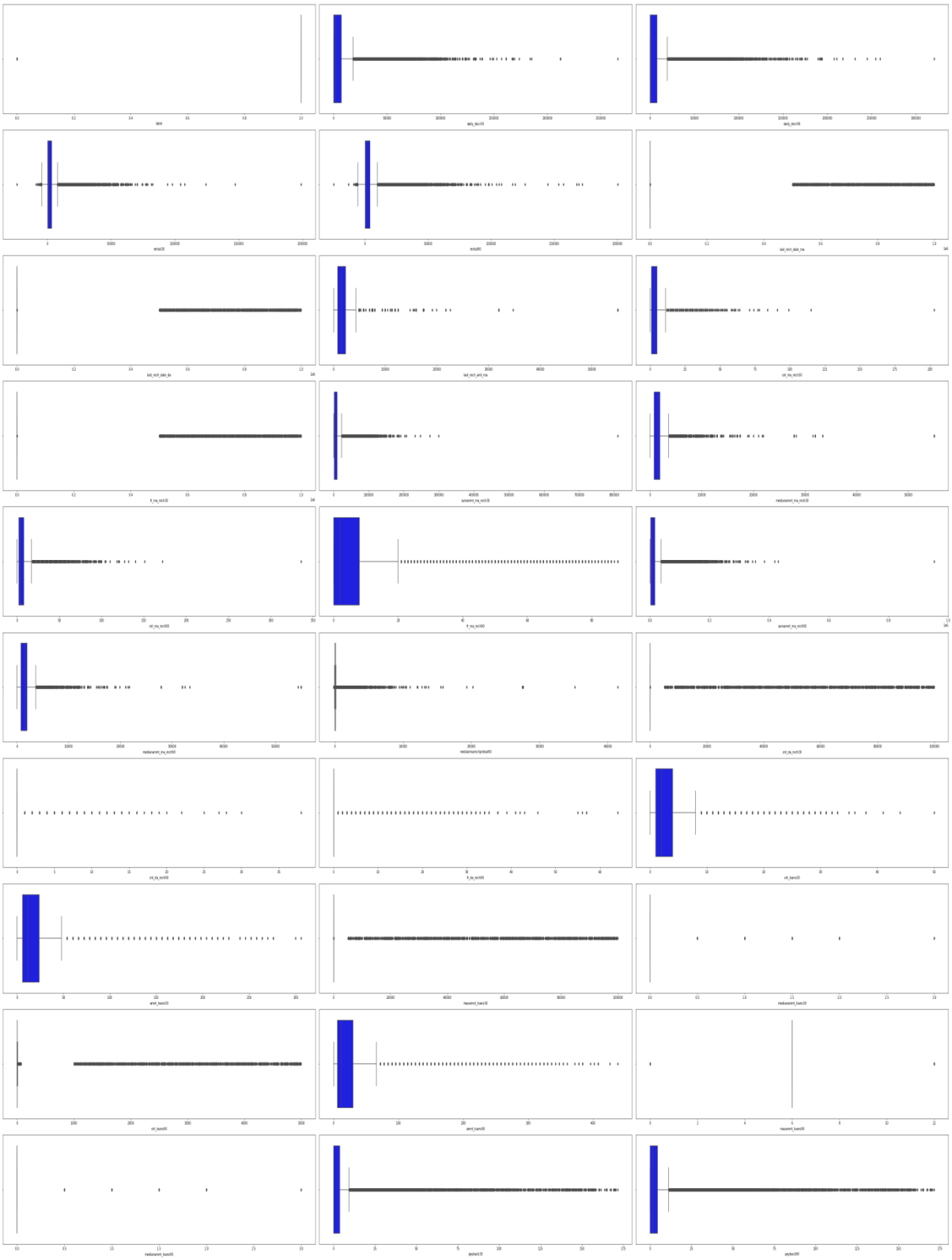


Observation:

From the above visualization we could deduce that all the attribute values are positively skewed. Thus the loss percent in the Data-set seems to low.

8.5 Checking for Outliers

```
#Plotting Outliers
collist=df.columns.values
ncol=3
nrow=11
plt.figure(figsize=(50,50))
for i in range (0,len(collist)):
    plt.subplot(nrow,ncol,i+1)
    sn.boxplot(df[collist[i]],color="blue",orient="h")
    plt.tight_layout()
```


label
daily_dev30
daily_dev90
errs30
errs90
last_rech_date_ma
last_rech_date_da
last_rech_amt_ma
cnt_ma_rech30
f_ma_rech30
sumamnt_ma_rech30
medianamnt_ma_rech30
cnt_ma_rech90
f_ma_rech90
sumamnt_ma_rech90
medianamnt_ma_rech90
medianamnt_loans90
cnt_da_rech30
cnt_da_rech90
f_da_rech90
cnt_loans30
amnt_loans30
maxamnt_loans30
medianamnt_loans30
cnt_loans90
amnt_loans90
maxamnt_loans90
medianamnt_loans90
payback30
payback90

Observation:

1. This above plot shows us that there are Outliers present in most of the Attributes.
2. Now we will try to remove the outliers using Z-score method.

8.6 Checking for removal of outliers

```
#Removing Outliers
#Z-score Techinique
from scipy.stats import zscore
z=np.abs(zscore(df))
z
```

```
threshold=3
df_new=df[(z<3).all(axis=1)]
df_new
```

```
df_new.shape
```

```
(164713, 30)
```

```
df.shape
```

```
(209593, 30)
```

```
loss_percentage =(209593-164713)/209593*100
loss_percentage
```

```
21.41292886689918
```

Observations:

We can see that the loss percentage is around 21.4%. But the Project Requirements suggest that we should not lose more than 7-8% of the data. As per the pre-processing engineering we should remove this kind of outliers from the data-set. But removing the 21.4% of loss will result in voiding the project requirements.

Thus we will continue without removing the outliers.

ML Model Development and Evaluation

1. Problem Identification

The given objective for our project is to predict the instance that whether the people will pay the loan taken by them or not. That instance is labeled in the Target variable as 1's and 0's where 1's are customers paying the loan on time and 0's are customers not paying the loan on time. This is a classic case of **"Classification"**. So we will use different type of classifying machine learning algorithms to build a prediction model.

2. Listing of ML Models

For this given Objective we will use the following Machine Learning Models.
1.Logistic Regression.
2.RandomForestClassifier.
3.NAIVE BAYES(GaussianNB).
4.KNeighborsClassifier.
5.DecisionTreeClassifier.
6.SVC.

1. Logistic Regression.

   Logistic Regression is a supervised learning algorithm that is used when the target variable is categorical. Hypothetical function h(x) of linear regression predicts unbounded values. But in the case of Logistic Regression, where the target variable is categorical we have to strict the range of predicted values. Consider a classification problem, where we need to classify whether an email is a spam or not. So, the hypothetical function of linear regression could not be used here to predict as it predicts unbound values, but we have to predict either 0 or 1 in the process.

2. RandomForestClassifier.

   The Random forest classifier creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees (DT) from a randomly selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction.

3. NAIVES BAYES(GaussianNB).

   Naive Bayes is among one of the very simple and powerful algorithms for classification based on Bayes Theorem with an assumption of independence among the predictors. The Naive Bayes classifier assumes that the presence of a feature in a class is not related to any other feature. Naive Bayes is a classification algorithm for binary and multi-class classification problems.

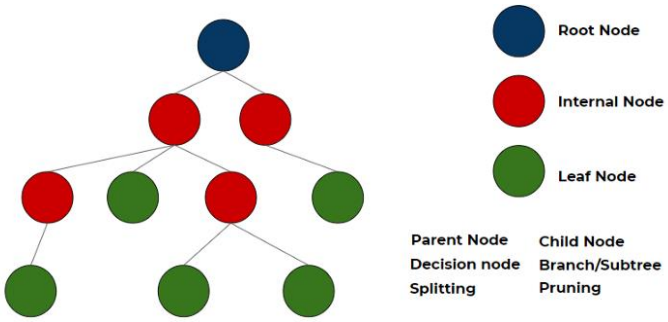
4. KNeighborsClassifier.

   K Nearest Neighbors Classification is one of the classification techniques based on instance-based learning. Models based on instance-based learning to generalize beyond the training examples. To do so, they store the training examples first. When it encounters a new instance (or test example), then they instantly build a relationship between stored training examples and this new instant to assign a target function value for this new instance. Instance-based methods are sometimes called lazy learning methods because they postponed learning until the new instance is encountered for prediction.
   Instead of estimating the hypothetical function (or target function) once for the entire space, these methods will estimate it locally and differently for each new instance to be predicted.
   K-Nearest Neighbors Classifier Learning
   Basic Assumption:

   1. All instances correspond to points in the n-dimensional space where n represents the number of features in any instance.
   2. The nearest neighbors of an instance are defined in terms of the Euclidean distance

5. DecisionTreeClassifier.

   A classification tree is used when the dependent variable is categorical. The value obtained by leaf nodes in the training data is the mode response of observation falling in that region it follows a top-down greedy approach.
   The general idea behind the Decision Tree is to find the splits that can separate the data into targeted groups.
   Terminologies associated with decision tree



6. SVC.

Support Vector Machine (SVM) is a relatively simple Supervised Machine Learning Algorithm used for classification and/or regression. It is more preferred for classification but is sometimes very useful for regression as well. Basically, SVM finds a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line.
In SVM, we plot each data item in the dataset in an N-dimensional space, where N is the number of features/attributes in the data. Next, find the optimal hyperplane to separate the data. So by this, you must have understood that inherently, SVM can only perform binary classification (i.e., choose between two classes). However, there are various techniques to use for multi-class problems.
Support Vector Machine for Multi-Class Problems
To perform SVM on multi-class problems, we can create a binary classifier for each class of the data. The two results of each classifier will be:

- The data point belongs to that class OR
- The data point does not belong to that class.

For example, in a class of fruits, to perform multi-class classification, we can create a binary classifier for each fruit. For say, the 'mango' class, there will be a binary classifier to predict if it IS a mango OR it is NOT a mango. The classifier with the highest score is chosen as the output of the SVM.

3. Processing the Data-set for Training and Testing.

Now we will use the train_test split to split and train the data-set for testing the data-set. This is a crucial phase of building the ML model.

1. Training set is a subset of the dataset used to build predictive models.
2. 2. Test set, or unseen data, is a subset of the dataset used to assess the likely future performance of a model. If a model fits to the training set much better than it fits the test set, over fitting is probably the cause.

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.3,random_state=42)
```

```
x_train.shape
```

```
(146715, 29)
```

```
y_train.shape
```

```
(146715,)
```

```
x_test.shape
```

```
(62878, 29)
```

```
y_test.shape
```

```
(62878,)
```

4. Evaluation of ML Models.

Here will now build the ML models with the help of the given Data-set. We will also use accuracy score, confusion matrix, F1 score, and Roc_Auc score for finding the efficiency of the built Machine Learning Model.
Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data.

Classification Accuracy

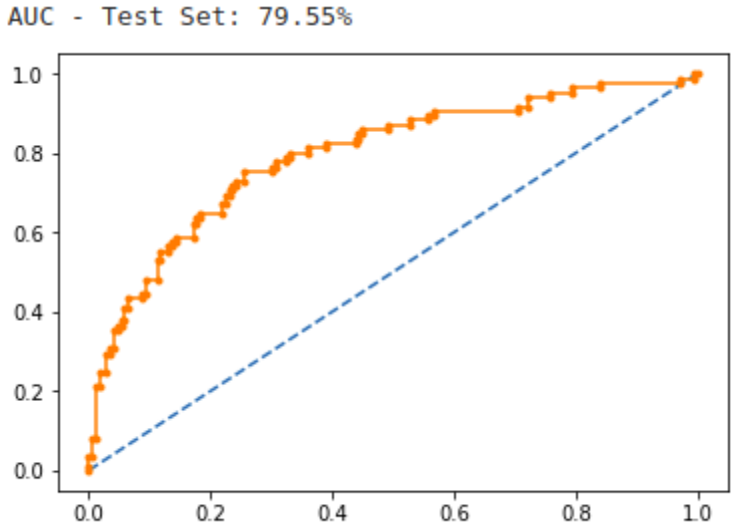
Accuracy is a common evaluation metric for classification problems. It's the number of correct predictions made as a ratio of all predictions made. We use sklearn module to compute the accuracy of a classification task.

Confusion Matrix

A confusion matrix provides a more detailed breakdown of correct and incorrect classifications for each class.

Area under Curve (AUC)

Area under ROC Curve is a performance metric for measuring the ability of a binary classifier to discriminate between positive and negative classes.



F-Measure

F-measure (also F-score) is a measure of a test's accuracy that considers both the precision and the recall of the test to compute the score. Precision is the number of correct positive results divided by the total predicted positive observations. Recall, on the other hand, is the number of correct positive results divided by the number of all relevant samples (total actual positives).

1. Logistic Regression.

```
#Logictic Regression
from sklearn.linear_model import LogisticRegression
lg=LogisticRegression()
lg.fit(x_train,y_train)#for Training purpose
lg_pred=lg.predict(x_test)#for Testing purpose
print(lg_pred)
print("\n\nAccuracy score : ",accuracy_score(y_test,lg_pred))
print("\n\nconfussion matrix \n",confusion_matrix(y_test,lg_pred))
print("\n\nclassification_report :  \n",classification_report(y_test,lg_pred))
print("\n\nf1_score :",f1_score(y_test,lg_pred))
print("\n\nroc_auc_score :",roc_auc_score(y_test,lg_pred))
```

```
[1 1 1 ... 1 1 1]


Accuracy score :  0.8753777155761951


confussion matrix
 [[  194  7707]
 [  129 54848]]


classification_report :
              precision    recall  f1-score   support

           0       0.60      0.02      0.05      7901
           1       0.88      1.00      0.93     54977

    accuracy                           0.88     62878
   macro avg       0.74      0.51      0.49     62878
weighted avg       0.84      0.88      0.82     62878



f1_score : 0.9333287955620596


roc_auc_score : 0.5111037090801513
```

Here in Logistic Regression we have acquired a Accuracy score of 87.5%. The F1 score is 93. This is a decent ML model. But we will try to infer from all the other ML Models too.

2. RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(x_train,y_train)#for Training purpose
rf_pred=rf.predict(x_test)#for Testing purpose
print(rf_pred)
print("\n\nAccuracy score : ",accuracy_score(y_test,rf_pred))
print("\n\nconfussion matrix \n",confusion_matrix(y_test,rf_pred))
print("\n\nclassification_report :  \n",classification_report(y_test,rf_pred))
print("\n\nf1_score :",f1_score(y_test,rf_pred))
print("\n\nroc_auc_score :",roc_auc_score(y_test,rf_pred))
```

```
[1 1 1 ... 1 1 1]


Accuracy score :  0.9120201024205604


confussion matrix
 [[ 3539  4362]
 [ 1170 53807]]


classification_report :
               precision    recall  f1-score   support

           0       0.75      0.45      0.56      7901
           1       0.93      0.98      0.95     54977

    accuracy                           0.91     62878
   macro avg       0.84      0.71      0.76     62878
weighted avg       0.90      0.91      0.90     62878



f1_score : 0.951107418733318


roc_auc_score : 0.7133181791015197
```

Here from the RandomForestClassifier we got an accuracy score of 91.2%. which implies that it is performing well with the given data-set.

3. NAVIE BAYES (GaussianNB)

```
#NAIVE BAYES
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
gnb = GaussianNB()
gnb.fit(x_train,y_train)
y_pred=gnb.predict(x_test)
acc={metrics.accuracy_score(y_test, y_pred)*100}
print("Accuracy:",acc)
```

```
Accuracy: {58.02665479181908}
```

Here in Navie Bayes (Gaussian NB) we got an accuracy range of 58.02%, which is pretty much low compared to all other ML Models.

4. KNeighborsClassifier

```
#KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)#for Training purpose
knn_pred=knn.predict(x_test)#for Testing purpose
print(knn_pred)
print("\n\nAccuracy score : ",accuracy_score(y_test,knn_pred))
print("\n\nconfussion matrix \n",confusion_matrix(y_test,knn_pred))
print("\n\nclassification_report :  \n",classification_report(y_test,knn_pred))
print("\n\nf1_score :",f1_score(y_test,knn_pred))
print("\n\nroc_auc_score :",roc_auc_score(y_test,knn_pred))
```

```
[1 1 1 ... 1 1 1]


Accuracy score :  0.8912815293107287


confussion matrix
 [[ 3123  4778]
 [ 2058 52919]]


classification_report :
               precision    recall  f1-score   support

           0       0.60      0.40      0.48      7901
           1       0.92      0.96      0.94     54977

    accuracy                           0.89     62878
   macro avg       0.76      0.68      0.71     62878
weighted avg       0.88      0.89      0.88     62878


f1_score : 0.9393293927614179


roc_auc_score : 0.6789162930020669
```

Here in KNeighborsClassifier we got an accuracy score 89.12%.

5. SVC

```
from sklearn.svm import SVC
svc=SVC()
svc.fit(x_train,y_train)#for Training purpose
svc_pred=svc.predict(x_test)#for Testing purpose
print(svc_pred)
print("\n\nAccuracy score : ",accuracy_score(y_test,svc_pred))
print("\n\nconfussion matrix \n",confusion_matrix(y_test,svc_pred))
print("\n\nclassification_report :  \n",classification_report(y_test,svc_pred))
print("\n\nf1_score :",f1_score(y_test,svc_pred))
print("\n\nroc_auc_score :",roc_auc_score(y_test,svc_pred))
```

```
[1 1 1 ... 1 1 1]


Accuracy score :  0.8765068863513471


confussion matrix
 [[  263  7638]
 [  127 54850]]


classification_report :
               precision    recall  f1-score   support

           0       0.67      0.03      0.06      7901
           1       0.88      1.00      0.93     54977

    accuracy                           0.88     62878
   macro avg       0.78      0.52      0.50     62878
weighted avg       0.85      0.88      0.82     62878



f1_score : 0.9338952028263738


roc_auc_score : 0.5154884343863538
```

Here we have got 87.65% using SVC. Along with an F1 score of 93.

OBSERVATION:

From the above results we can visibly see that the “RandomForestClassifier" is having a better accuracy score of 91% along with a promising f1 score and classification report.

Now we will try to tune the "RandomForestClassifier" model using Hyper parameters for a better accuracy score.

6. DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)#for Training purpose
dtc_pred=dtc.predict(x_test)#for Testing purpose
print(dtc_pred)
print("\n\nAccuracy score : ",accuracy_score(y_test,dtc_pred))
print("\n\nconfussion matrix \n",confusion_matrix(y_test,dtc_pred))
print("\n\nclassification_report :  \n",classification_report(y_test,dtc_pred))
print("\n\nf1_score :",f1_score(y_test,dtc_pred))
print("\n\nroc_auc_score :",roc_auc_score(y_test,dtc_pred))
```

```
[1 0 1 ... 0 1 1]


Accuracy score :  0.8677915964248226


confussion matrix
 [[ 3984  3917]
 [ 4396 50581]]


classification_report :
              precision    recall  f1-score   support

           0       0.48      0.50      0.49      7901
           1       0.93      0.92      0.92     54977

    accuracy                           0.87     62878
   macro avg       0.70      0.71      0.71     62878
weighted avg       0.87      0.87      0.87     62878



f1_score : 0.9240648549897237


roc_auc_score : 0.7121396293906911
```

Here using DecisionTreeClassifier we got an accuracy score of 86.77%. Along with a F1 score of 92.

Hyper Tuning of the ML Models.

Hyper parameter Tuning RandomForestClassifier using GridSearchCV

```
#Hyper parameter Tuning RandomForestClassifier using GridSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
parameters={'n_estimators':[10,20,50],'criterion':['gini','entropy'], 'max_depth':[4,6,8,10]}
rf=RandomForestClassifier()
clf=GridSearchCV(rf,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

```
{'criterion': 'gini', 'max_depth': 10, 'n_estimators': 50}
```

```
GCV=GridSearchCV(rf,parameters,cv=5,scoring="accuracy")
GCV.fit(x_train,y_train)
GCV.best_estimator_#finding best estimator
GCV_pred=GCV.best_estimator_.predict(x_test)
print("final accuracy = ",accuracy_score(y_test,rf_pred)*100)
```

```
final accuracy =  91.20201024205605
```

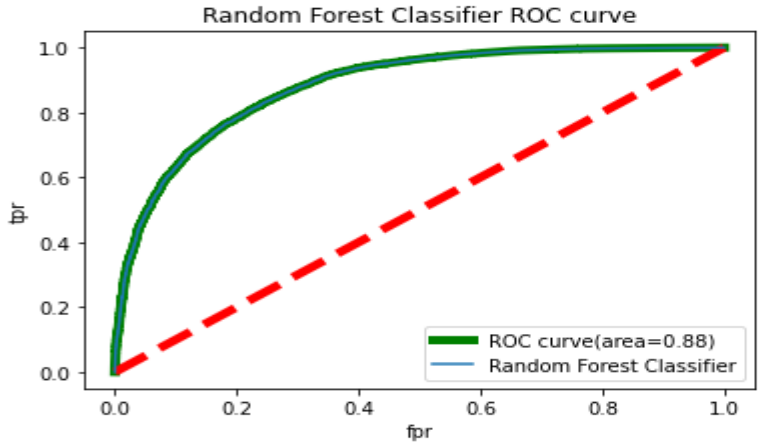
Our FInal Accuracy is 91.20%

ROC_AUC CURVE

```
from sklearn.metrics import roc_curve,auc
y_pred_roc = GCV.predict_proba(x_test)[:,1]
```

```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_roc)
roc_auc=auc(fpr,tpr)
```

```
# Plotting the ROC Curve
plt.plot(fpr,tpr,color="green",lw=5,label="ROC curve(area=%0.2f)"%roc_auc)
plt.plot([0,1],[0,1],color="red",lw=5,linestyle="--")
plt.plot(fpr,tpr, label='Random Forest Classifier')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('Random Forest Classifier ROC curve')
plt.legend(loc="lower right")
plt.show()
```



7. Final Results

SAVING THE MODEL

```
import pickle
filename="Micro_credit_project.pkl"
pickle.dump(GCV,open(filename,"wb"))
```

TESTING THE PREDICTION

```
import numpy as np
a=np.array(y_test)
predicted=np.array(GCV.predict(x_test))
df_con=pd.DataFrame({"original":a,"Predicted":predicted})
df_con.tail(5)
```

	original	Predicted
62873	1	1
62874	1	1
62875	1	1
62876	1	1
62877	1	1

CONCLUSION

KEY FINDINGS AND LEARNING OUTCOMES

Microfinance program of this telecom company has a positive impact on the lives of the poor. It is not a question of whether or not the poor need information technology, but rather what the appropriate technology is that will enable them to escape poverty. Whatever the technology, it needs to be seen as a device with real and relevant benefits, not as an extravagance. Microfinance has been found an operational tool for lifting the poor by providing them financial services to maintain communication and to keep in touch with their dear ones.

But the raise in defaulters not paying the loan back is the biggest nightmare for the Telecom industry. Because increase in defaulters will result in lesser funding for the MFS. Thus in order to save the MFS the telecommunication industry turns towards the technology asking for help.

So we technology services build certain ML Models to predict the defaulters and to reduce the MFS to regular defaulters which reduces the financial risk for the MFI.

Here during our ML Model building exercise we took the sample data provided by the client and followed a series of analysis in order to make the data more efficient. For that we used exploratory data analysis, correlation chart, various types of visualizations in order to help us find the gist of the core problem. On this process we have built an efficient ML Model with the help of Random forest classifier which has an accuracy rate of 91.2%.

Thus in future the telecom industry can use this ML Model to predict the defaulters and may control the financial stability of the MFI. Making this the actual people who may need the MFS will get them on time without any delays.

LIMITATIONS AND FUTURE WORK

When we were handling the outliers there is a whopping 21% of outliers were present in the data-set. But data is a precious thing. We can't afford to lose data's. Also the client requirement was not to lose more than 7-8% of the data. Thus we continued building a ML model without removing the outliers.

This in turn affects the accuracy of the ML model directly. The more the outliers the less efficient the Machine learning model will be. So the data-set should be engineered in a better way so that there won't be any loss of the data.

Also most of the attributes doesn't give a plausible relation with the target value during the EDA. Visualization of such instance seems to be a hardship. Thus more relatable attributes would help us to build a more efficient MLM.

THANK YOU