NAME OF THE PROJECT

Micro-Credit Defaulter using Machine Learning

Submitted by:

Mrs. Swati Amit Motugade

FLIPROBO SME:

Ms. Gulshana Chaudhari

ACKNOWLEDGMENT

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analysis skills. Also, I want to express my huge gratitude to Ms. Gulshana Chaudhari Mam (SME, Flip Robo), she is the person who has helped me to get out of all the difficulties I faced while doing this project.

A huge thanks to “Data trained” who are the reason behind Internship at Fliprobo. Last but not least my parents who are there to support me at every step of my life.

References used in this project:

- SCIKIT Learn Library Documentation.
- Blogs from towardsdatascience, Analytics Vidya, Medium.
- Andrew Ng Notes on Machine Learning (GitHub).
- Data Science Projects with Python Second Edition by Packt
- Hands on Machine learning with scikit learn and tensor flow by Aurelien Geron.
- Sifei Lu, Zengxiang Li, Zheng Qin, Xulei Yang, Rick Siow Mong Goh,” A Hybrid Regression Technique for House Prices Prediction”, @2017 IEEE, 2017 IEEE International Conference on Industrial Engineering & Engineering Management, Singapore DOI:10.1109/IEEM.2017.8289904
- CH. Raga Madhuri, Anuradha G, M. Vani Pujitha “House Price Prediction Using Regression Techniques: A Comparative Study”, IEEE 6th International Conference on smart structures and systems ICSSS 2019
- J.-G. Liu, X.-L. Zhang, and W.-P. Wu, “Application of fuzzy neural network for real estate prediction,” Advances in Neural Networks -ISNN 2006, vol. 3973, pp. 1187–1191, 2006.

- H. Kusan, O. Aytekin, and I. Ozdemir, "*e use of fuzzy logic ¨ in predicting house selling price," Expert Systems with Applications, vol. 37, no. 3, pp. 1808–1813, 2010.
- Ayush Varma, Abhijit Sarma, Rohini Nair and Sagar Doshi," House Price Prediction Using Machine Learning and Neural Networks", @2018 IEEE, 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, DOI:10.1109/ICICCT.2018.8473231.

Also, there are so many people who helped me directly and indirectly to complete this project.

Mrs. Swati Amit Motugade

Chapter 1

Introduction

1.1 Business Problem Framing

A Microfinance Institution (MFI) is an organization that offers financial services to low-income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on.

Many microfinance institutions (MFI), experts and donors are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient, and cost saving, than the traditional high-touch model used since long for the purpose of delivering microfinance services. Though, the MFI industry is primarily focusing on low-income families and are very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes.

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.

1.2 Conceptual Background of the Domain Problem

The Telecom Industry understands 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.

We have to build a model which can be used to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan. In this case, Label '1' indicates that the loan has been repaid i.e., non-defaulter, while, Label '0' indicates that the loan has not been repaid i.e., defaulter.

1.3 Review of Literature

What is Microcredit?

Microcredit, is a type of banking service provided to unemployed or low-income individuals or groups who otherwise would have no other access to financial services.

While institutions participating in the area of microfinance most often provide lending—microloans can range from as small as $100 to as large as $25,000—many banks offer additional services such as checking and savings accounts as well as micro-insurance products, and some even provide financial and business education. The goal of microfinance is to ultimately give impoverished people an opportunity to become self-sufficient.

More about Microcredit

- Microcredit is a banking service provided to unemployed or low-income individuals or groups who otherwise would have no other access to financial services.

- Microcredit allows people to take on reasonable small business loans safely, and in a manner that is consistent with ethical lending practices.
- The majority of microfinancing operations occur in developing nations, such as Uganda, Indonesia, Serbia, and Honduras.
- Like conventional lenders, microfinanciers charge interest on loans and institute specific repayment plans.
- The World Bank estimates that more than 500 million people have benefited from microfinance-related operations.

Microcredit in India

Microcredit is a way in which loans, credit, insurance, access to savings accounts, and money transfers are provided to small business owners and entrepreneurs in the underdeveloped parts of India.

The beneficiaries of microfinance are those who do not have access to these traditional financial resources. Interest rates on microloans are generally higher than that on traditional personal loan.

Importance of Microcredit

Almost half of the population of our country does not have a basic savings account. However, this segment requires financial services so that their aspirations such as building of assets and protection against risk can be fulfilled.

Microfinance provides access to capital for individuals who are financially underserved. If microfinance institutions were not offering loans to this segment of the society, these groups would have resorted to borrowing money from friends or family members. The probability of them opting for fast cash loans or payday advances (that bear huge interest rates) are also high.

Microfinance helps these groups invest wisely in their businesses, and hence, is in alignment with the government's vision of financial inclusion in the country.

Key features of Microcredit

- The borrowers are generally from low-income backgrounds
- Loans availed under microfinance are usually of small amount, i.e., micro loans
- The loan tenure is short
- Microfinance loans do not require any collateral
- These loans are usually repaid at higher frequencies
- The purpose of most microfinance loans is income generation

The CGAP (2010) identifies some unique features of microfinance as follows:

➢ Delivery of very small loans to unsalaried workers

➢ Little or no collateral requirements

➢ Group lending and liability

➢ Pre-loan savings requirement

➢ Gradually increasing loan sizes

Default in Microcredit

Default in microfinance is the failure of a client to repay a loan. The default could be in terms of the amount to be paid or the timing of the payment. MFIs can sustain and increase deployment of loans to stimulate the poverty reduction goal if repayment rates are high and consistent (Wongnaa 2013).

Machine Learning Techniques for microfinance & finance

Pollio and Obuobie [ ] applied logistic regression on four factors and concluded that the probability of default increases with the number of

dependents, whether the proceeds are used to acquire fixed assets, the frequency of monitoring, decreases with the availability of non-business income, years in business, the number of guarantors, whether the proceeds were used for working capital purposes and whether the client is a first-time borrower.

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 modelling 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, indicated that the models such as the logistic regression, random forest, and gradient boosting modelling generated more accurate results than the models based on the neural network approach incorporating various technicalities.

Machine learning-based systems are growing in popularity in research applications in most disciplines. Considerable decision-making knowledge from data has been acquired in the broad area of machine learning, in which decision-making tree-based ensemble techniques are recognized for supervised classification problems. Classification is an essential form of data analysis in data mining that formulates models while describing significant data classes (Rastogi and Shim 2000). Accordingly, such models estimate categorical class labels, which can provide users with an enhanced understanding of the data at large Hanet al. (2012) resulted in significant advancements in classification accuracy.

1.4 Motivation for the Problem Undertaken

The project was the first provided to me by Flip Robo Technologies as a part of the internship programme. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary motivation.

This project includes the real time problem for Microfinance Institution (MFI), and it is related to financial sectors, as I believe that with growing technologies and Idea can make a difference, there are so much in the financial market to explore and analyse and with Data Science the financial world becomes more interesting. The objective of the project is to prepare a model based on the sample dataset that classifies all loan defaulters and help our client in further investment and improvement in selection of customers. The model will be a good way for the management to understand whether the customer will be paying back the loaned amount within 5 days of insurance of loan.

Chapter 2

Analytical Problem Framing

2.1 Mathematical / Analytical Modelling

Whenever we employ any ML algorithm, statistical models or feature pre-processing in background lot of mathematical framework work. In this project we have done lot of data pre-processing & ML model building.

Our objective 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 this case, Label '1' indicates that the loan has been paid i.e., non-defaulter, while, Label '0' indicates that the loan has not been paid i.e., defaulter, which can be resolve by use of classification-based algorithm. In this project we are going to use different types of algorithms which uses their own mathematical equation on background. Initially data cleaning & pre-processing perform over data. Feature engineering is performed to remove unnecessary feature & for dimensionality reduction. In model building Final model is select based on evaluation benchmark among

different models with different algorithms. Further Hyperparameter tuning performed to build more accurate model out of best model.

Followings are the algorithms used in this project

1. Logistic Regression
2. Random Forest Classifier
3. Decision Tree Classifier
4. K-Neighbours Classifier
5. Extra Tree Classifier

2.2 Dataset loading and Description

Before loading dataset, it is required to import all the necessary libraries.

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

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

The dataset is provided by my internship company – Fliprobo Technologies which is in the form of excel sheet and it is loaded in jupyter notebook using panda's function

```
df= pd.read_csv(r'C:\Users\Swati\OneDrive\Desktop\DataScience\Projects\Micro Credit Project\Data file.csv')
df
```

Points to Remember about dataset:

- There are no null values in the dataset.
- There may be some customers with no loan history.

- The dataset is imbalanced. Label ‘1’ has approximately 87.5% records, while, label ‘0’ has approximately 12.5% records.
- For some features, there may be values which might not be realistic. You may have to observe them and treat them with a suitable explanation.
- You might come across outliers in some features which you need to handle as per your understanding. Keep in mind that data is expensive and we cannot lose more than 7-8% of the data.

Checking for datatypes

```
1 df.columns.to_series().groupby(df.dtypes).groups
```

```
{int64: ['Unnamed: 0', 'label', 'last_rech_amt_ma', 'cnt_ma_rech30', 'cnt_ma_rech90', 'fr_ma_rech90', 'sumamnt_ma_rech90', 'cnt
_da_rech90', 'fr_da_rech90', 'cnt_loans30', 'amnt_loans30', 'amnt_loans90', 'maxamnt_loans90'], float64: ['aon', 'daily_decr30
', 'daily_decr90', 'rental30', 'rental90', 'last_rech_date_ma', 'last_rech_date_da', 'fr_ma_rech30', 'sumamnt_ma_rech30', 'medi
anamnt_ma_rech30', 'medianmarechprebal30', 'medianamnt_ma_rech90', 'medianmarechprebal90', 'cnt_da_rech30', 'fr_da_rech30', 'ma
xamnt_loans30', 'medianamnt_loans30', 'cnt_loans90', 'medianamnt_loans90', 'payback30', 'payback90'], object: ['msisdn', 'pcirc
le', 'pdate']}
```

The different datatypes of these features are as shown in above figure. Out of all features only three features with object datatypes and rest are int64. We can note here ‘pdate’ has datatype of object instead of datetime datatype.

2.3 Data Pre-processing

The dataset is large and it may contain some data error. In order to reach clean, error free data some pre-processing is done on data. At first integrity check is perform on data for presence of missing values, whitespaces. After that statistical matrix is plotted using df.describe () command to gain more insight about data.

- Missing value check – Data contain no missing value
- Data integrity check –

The feature ‘Unnamed :0’ is of no use. So, we dropped this feature. Also, there is a datetime feature which we have converted into day, month and year data. So, drop the original date feature.

```
df.drop(['Unnamed: 0'],axis=1,inplace=True)
```

```
df['pdate']=pd.to_datetime(df['pdate'])
df['Day']=df['pdate'].apply(lambda x:x.day)
df['Month']=df['pdate'].apply(lambda x:x.month)
df['Year']=df['pdate'].apply(lambda x:x.year)
df.head()
```

```
df.drop(['pdate'],axis=1,inplace=True)
```

We checked for the value counts of 'year' and got that there is only one-year 2016 for which the data is recorded. Hence, we dropped the feature.

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

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

Since the data is only for year 2016 there is no use of column year. Hence we will drop this column.

We checked for duplicate entries and found one duplicate which we have dropped.

```
df.duplicated().sum()
```

```
1
```

there is one duplicate entry found

```
df.duplicated('msisdn').sum()
```

```
23350
```

lets drop duplicate entries

```
df.drop_duplicates(keep='last',inplace=True)
```

Check for shape of dataset after removing duplicate entries

Shape of data before removing duplicates

```
print('No. of Rows :',df.shape[0])
print('No. of Columns :',df.shape[1])
pd.set_option('display.max_columns',None)
df.head()
```

```
No. of Rows : 209593
No. of Columns : 37
```

Shape of data after removing duplicates

```
df.shape
```

```
(209592, 37)
```

Data integrity check

We checked for [‘ ‘, ‘?’, ‘-‘, ‘NA’, ‘N/A’] and found that there are no data with the above mentioned integrity.

```
df.isin(['NA','N/A','-',' ','?',' ?']).sum().any()
```

```
False
```

There are no ['NA','N/A','-','?',',',' ',' ?'] present in our dataset

Descriptive Statistics

```
df.describe().T
```

```
df.describe(include=object)
```

From df.describe () command we got some key observation about data. One of it was that some features contain negative values and another observation few features contain extreme maximum value indicating possible outliers or invalid data.

Strategy to handle data error in min and max column –

Assumption- All negative values are typing error happen accidentally by type - in front of original value (except feature depicting median).

Corrective approach - Negative values are converted into absolute value to correct negative typing error whenever applicable except feature depicting median.

```
#Let's convert the negative values into positive ones
df['aon']=abs(df['aon'])
df['daily_decr30']=abs(df['daily_decr30'])
df['daily_decr90']=abs(df['daily_decr90'])
df['rental30']=abs(df['rental30'])
df['rental90']=abs(df['rental90'])
df['last_rech_date_ma']=abs(df['last_rech_date_ma'])
df['last_rech_date_da']=abs(df['last_rech_date_da'])
```

Data error and correction in maxamnt_loans30 column

```
# Let's chck for the feature 'maxamnt_loans30'
df['maxamnt_loans30'].describe()
```

```
count    209592.000000
mean        274.660029
std        4245.274734
min           0.000000
25%           6.000000
50%           6.000000
75%           6.000000
max       99864.560864
Name: maxamnt_loans30, dtype: float64
```

The maximum value in maxamnt_loans30 is not reliable. We already know maximum loan amount taken by customers can be 0,5,10 and which can be repay with amount of 0,6,12.

Assumption - The maximum value in maxamnt_loans30 is 12.

We gone replace values greater than 12 into category of zero.

```
df.loc[(df['maxamnt_loans30'] != 6.0) & (df['maxamnt_loans30'] != 12.0)
       & (df['maxamnt_loans30']!=0.0), 'maxamnt_loans30']=0.0
```

```
df['maxamnt_loans30'].value_counts()
```

```
6.0     179192
12.0     26109
0.0       4291
Name: maxamnt_loans30, dtype: int64
```

After performing this data pre-processing we saved the processed data in a new csv file named as ‘Clean_data_file’ and loaded this data file.

```
df.to_csv('Clean_data_file.csv',sep=',',index=False)
```

```
df=pd.read_csv('Clean_data_file.csv')
```

The features ‘msisdn’ and ‘pcircle’ are irrelevant features for prediction so we dropped these features.

```
df.drop(['msisdn', 'pcircle'],axis=1,inplace=True)
```

Outliers Detection

We plotted the boxplots to detect outliers.

```
plt.figure(figsize=(25,35),facecolor='white')
plotnumber=1

for column in df:
    if plotnumber<=36:
        ax=plt.subplot(9,4,plotnumber)
        sns.boxplot(df[column],color='purple')
        plt.xlabel(column,fontsize=20)

    plotnumber+=1

plt.show()
```

Outliers Removal

We got observation from boxplot that outliers do not exist in lower bound but outliers exist in upper bound of features. Based on this observation we decided to employ quantile-based flooring- capping method. Flooring is performed at 0th percentile for lower bound and capping perform at 99th percentile for upper bound.

```
Q1 = df.quantile(0)
Q3= df.quantile(0.99)
IQR = Q3 - Q1
print(IQR)
```

```
df_new = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
print(df_new.shape)
```

```
(198174, 35)
```

```
print("Percentage Data Loss : ", ((209592-198174)/209592)*100, "%")
```

```
Percentage Data Loss :  5.447727012481392 %
```

Here we lost 5.44% data on removing outliers which we can afford.

Skewness

Skewness is checked for all the features by plotting distribution plots for the features.

```
plt.figure(figsize=(25,35),facecolor='white')
plotnumber=1

for column in df_new:
    if plotnumber<=36:
        ax=plt.subplot(9,4,plotnumber)
        sns.distplot(df_new[column])
        plt.xlabel(column,fontsize=20)

    plotnumber+=1

plt.tight_layout()
```

```
df_new.skew()
```

Considerable amount of skewness found in most features by skew () function. Power transformer from sklearn.preprocessing library used to transform skewness in features.

Since the features 'fr_da_rech30' and 'fr_da_rech90' are having only zero values we dropped these two features.

```
df_new.drop(['fr_da_rech30','fr_da_rech90'],axis=1,inplace=True)
```

The features with skewness are

```
# skewed features are
skew_fea=['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','cnt_da_rech90', 'cnt_loans30', 'amnt_loans30',
          'maxamnt_loans30', 'medianamnt_loans30', 'cnt_loans90', 'amnt_loans90',
          'maxamnt_loans90','medianamnt_loans90', 'payback30', 'payback90']
```

To remove the skewness from the above skewed features we used the Power transform method.

```
from sklearn.preprocessing import PowerTransformer
```

```
pt = PowerTransformer(method='yeo-johnson')
```

```
df_new[skew_fea] = pt.fit_transform(df_new[skew_fea].values)
```

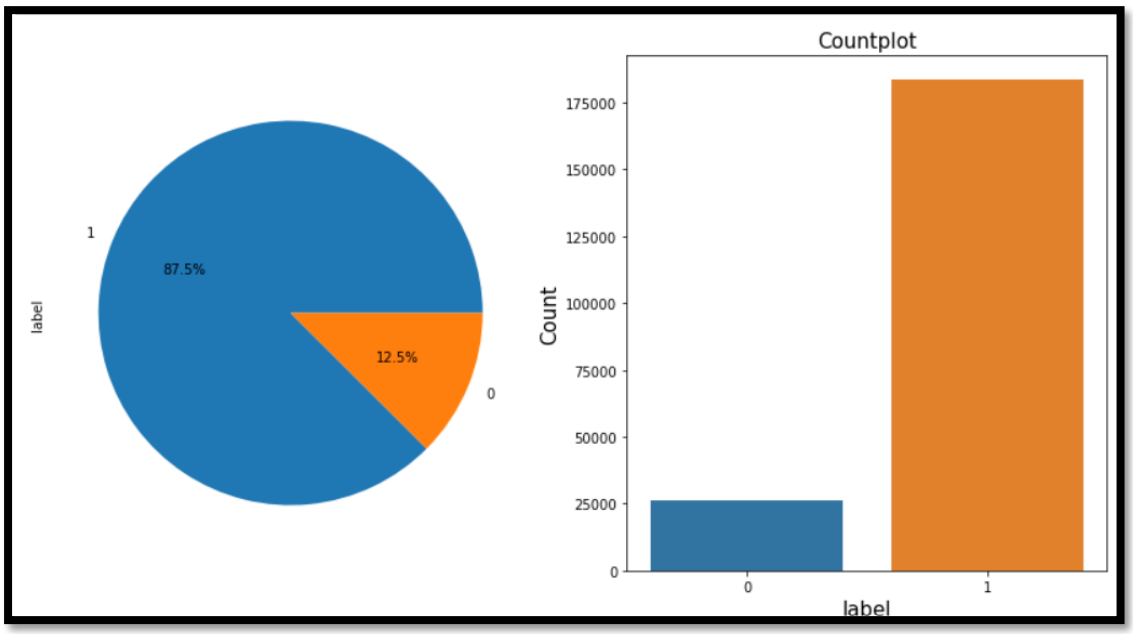
```
df_new.skew()
```

For most of feature's skewness is reduced within permissible limit except few ones.

Chapter 3
Data Visualization

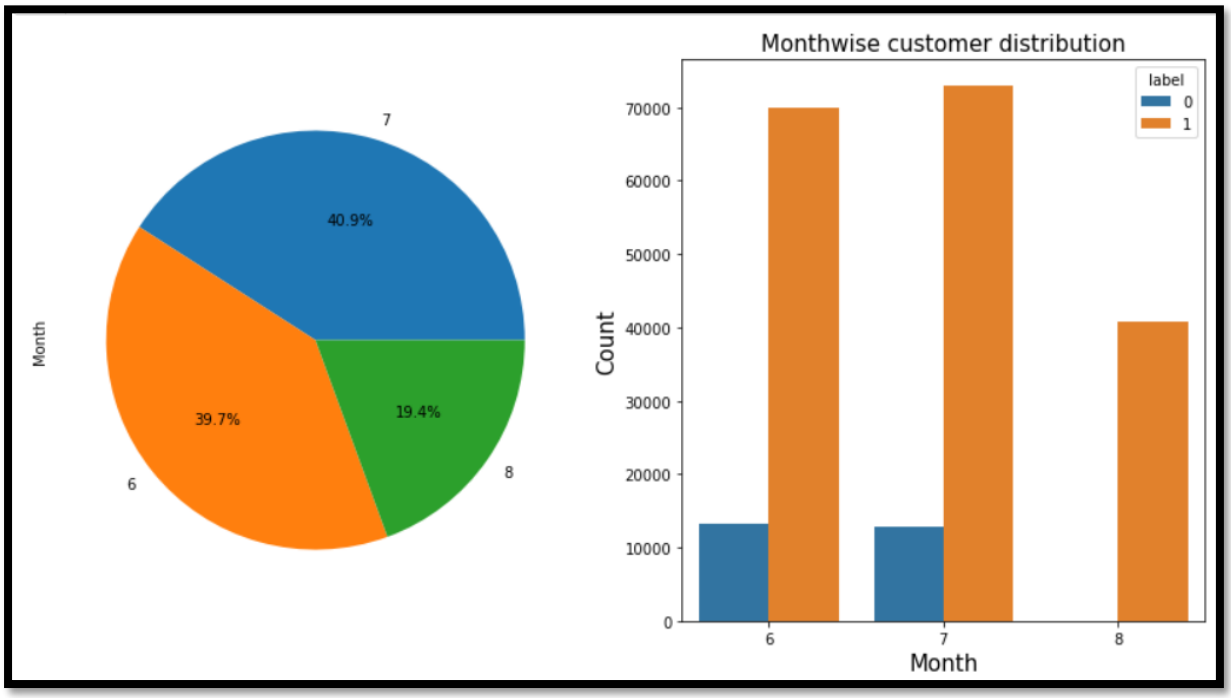
1. label:

Label is our target variable which defines the user is defaulter or not. The label value 1 represents that the user is not a defaulter i.e., user repaid the loan and label value 0 represents that the user is defaulter i.e., user didn't repay the loan.



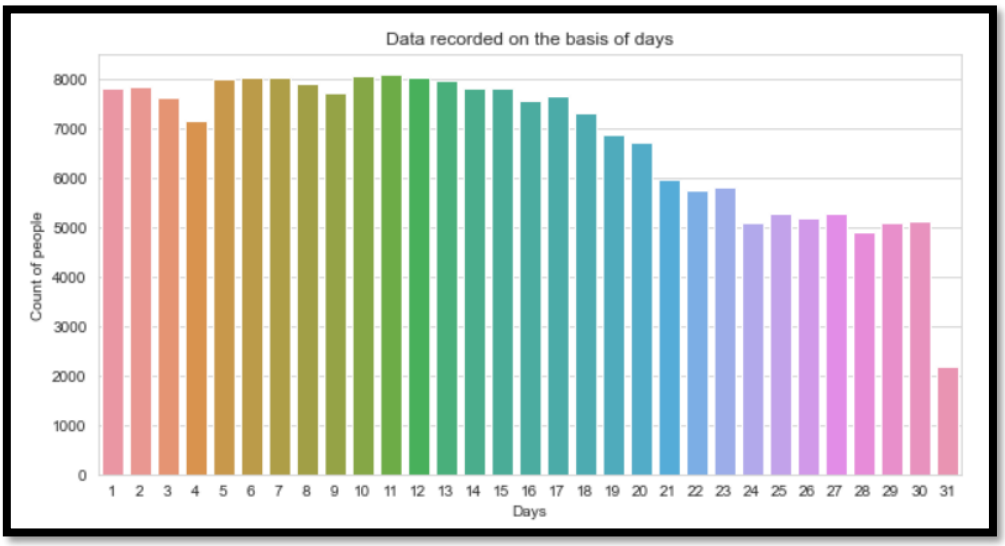
From the above plots we observed that more than 87% users are non-defaulter means they repaid the loan in mentioned time frame whereas only 12.5% users are defaulter.
Also, we noted that our target variable is highly imbalanced in order to build a model and we need to balance it before building a model.

2. Monthwise customer distribution



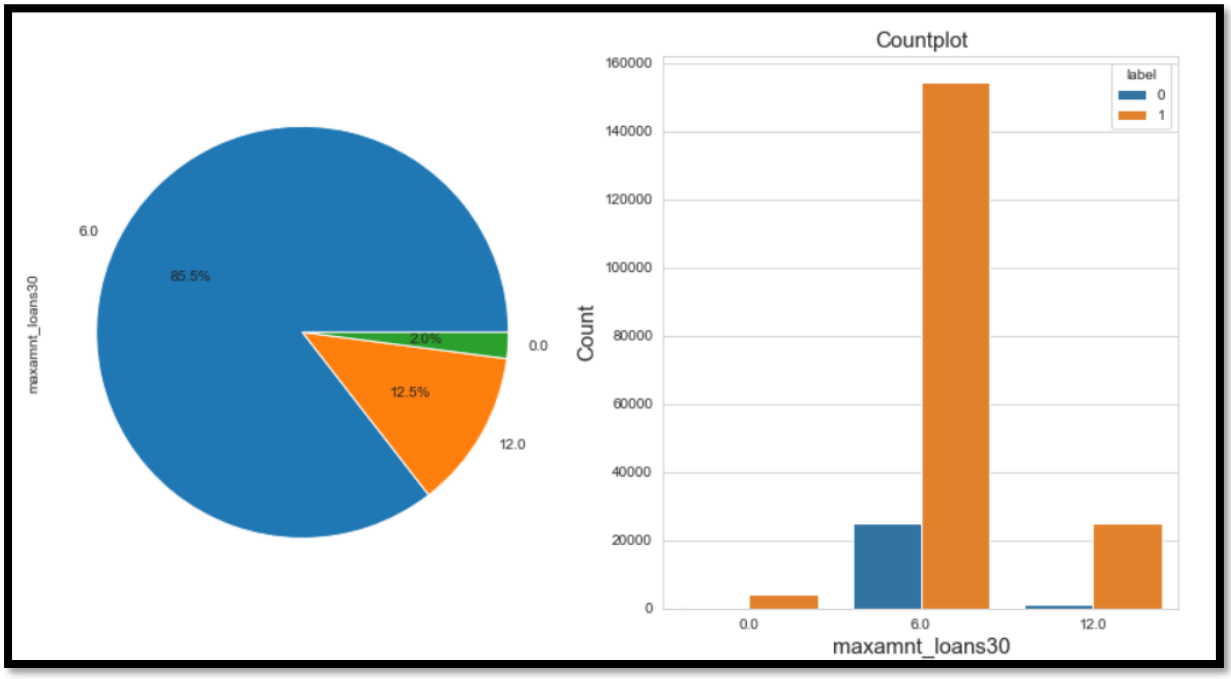
We can see that the data is only for three months i.e., June, July and August. Near about 41% loans are given in month of July, about 40% are given in June and 19.5% loans are given in August.
There are nearly equal numbers of defaulters for the loans paid in June & July. There are no defaulters for the loans paid in the month of August.

3. Day



4. maxamnt_loans30

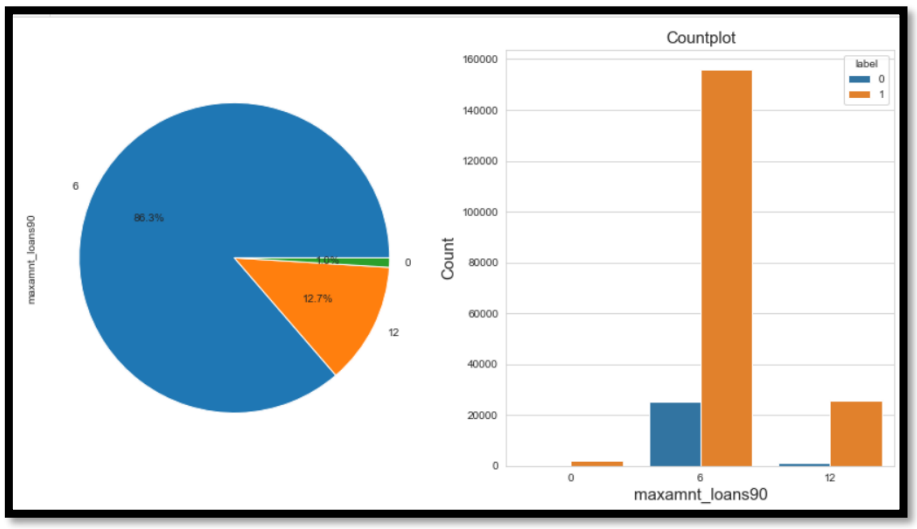
It is the maximum amount of loan taken by user in last 30 days



The pie chart shows that 85.5% users maximum amount of loan is 6, 15.5% users maximum loan amount is 12 where 2% users maximum loan amount is 0.

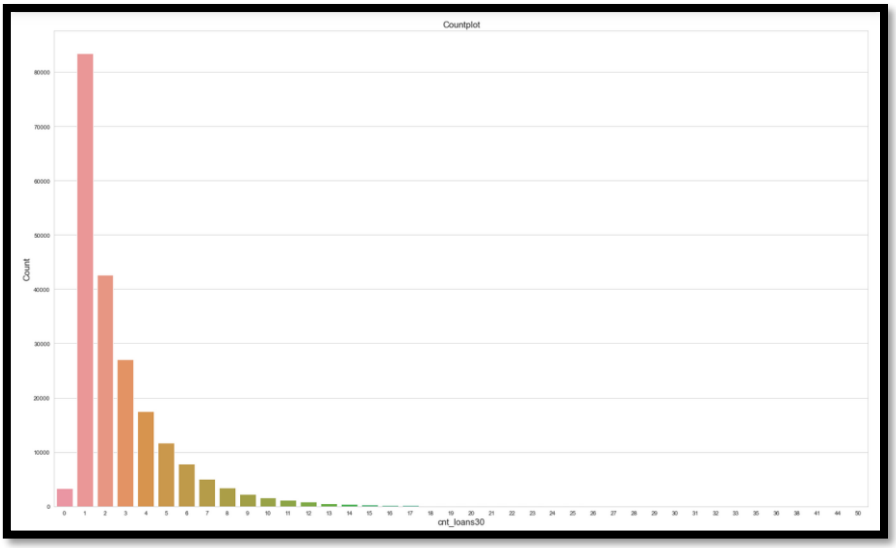
The count plot shows that the number of defaulters is minimum for 12 Rs loan as compared to 6 Rs. Loan.

5. maxamnt_loans90



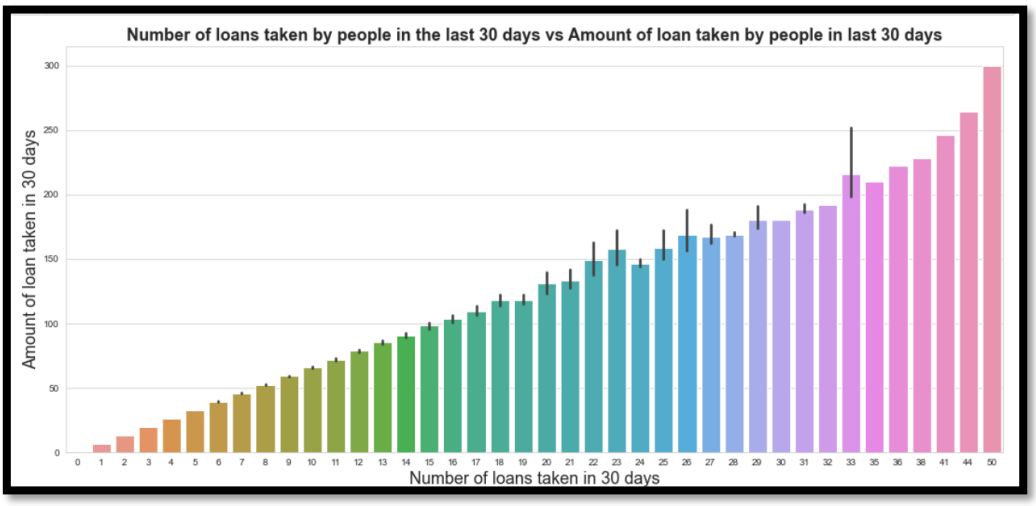
For last 90 days 86.3% users took loan of Rs 6, 12.7% users took loan of Rs 12 and 1% users didn't take any loan.
The count plot shows that the number of defaulters is minimum for 12 Rs loan as compared to 6 Rs. Loan.

6. cnt_loans30



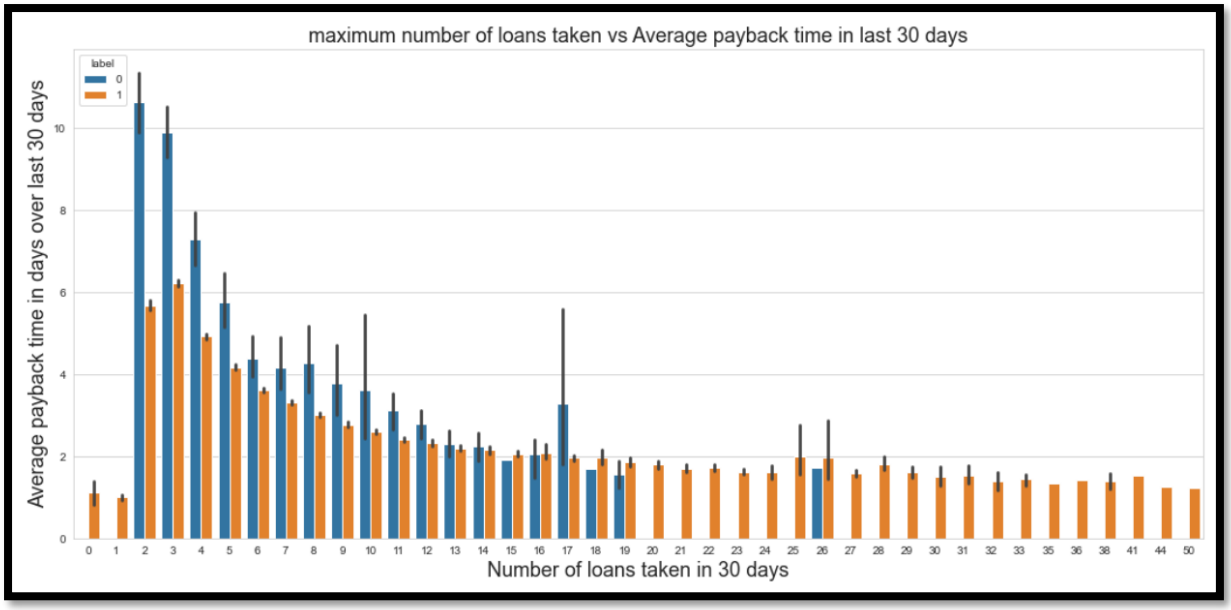
The maximum count of loan taken by user in last 30 days is 1 and the minimum count is 17.

7. cnt_loans30 vs amnt_loans30

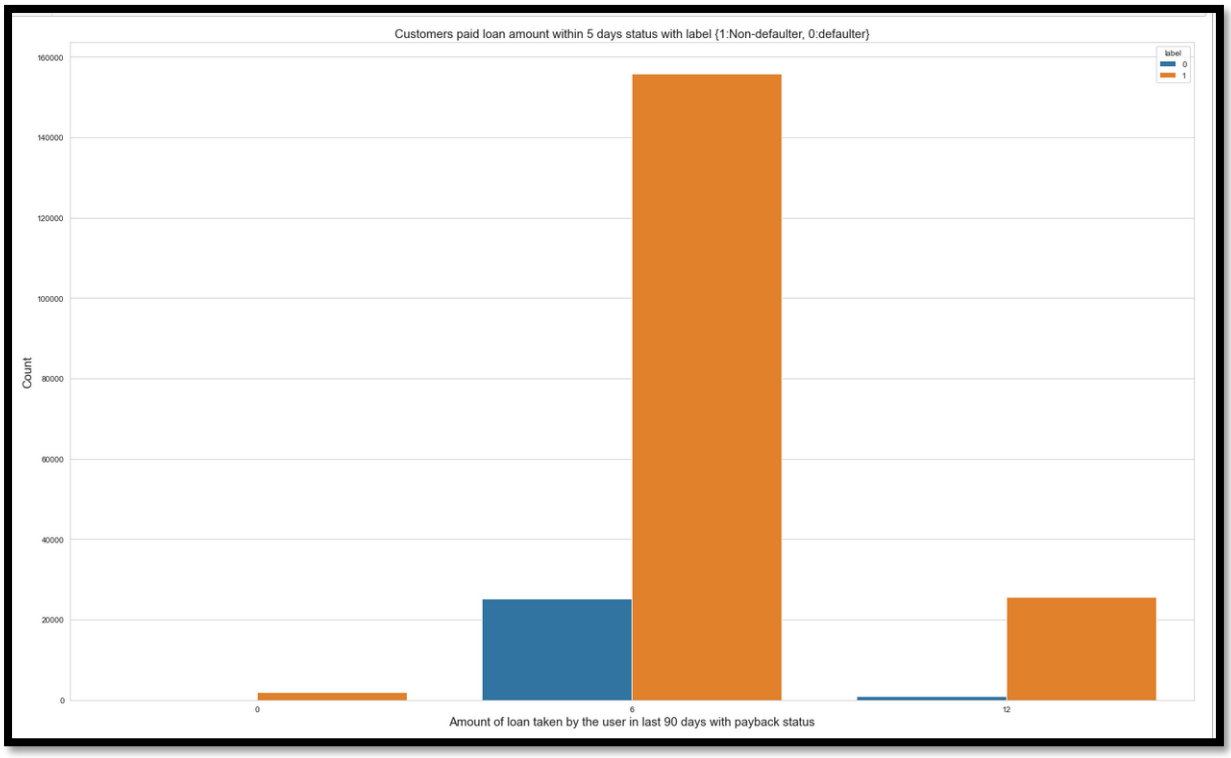


Maximum number of loans taken by the people is 50 and the Average loan amount is equivalent to 300. Minimum number of loans taken by the people is 0.

8. cnt_loans30 vs payback30



We can observe that the Average payback time over last 30 days is higher for people who had taken loan 2 times.



Very few defaulters in case of customers who have taken loan in amount of 12

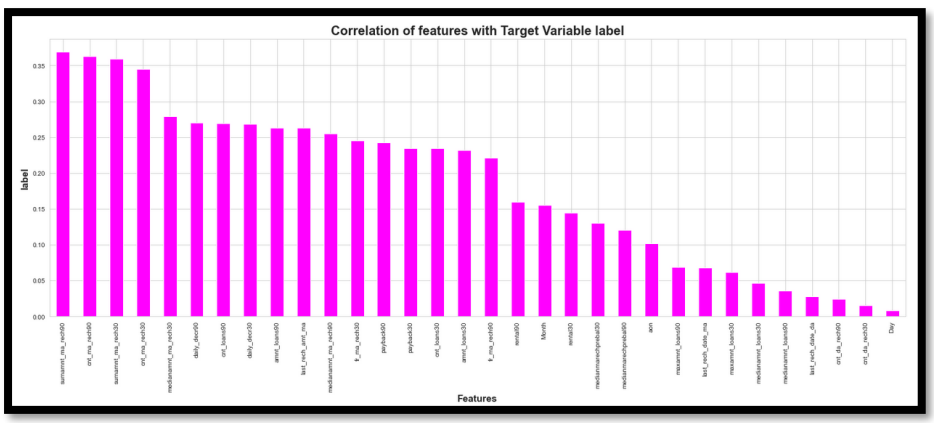
Interpretations:

- This dataset belongs from the year 2016 and the records are from the months June, July and August.
- From the visualization, I observed that the loan of amount 6Rs is taken for most of the times and most of the users repaid the loan within 5 days from issuing loan. But still there are few users failed to repay the loan.
- Most of the time the defaulters are the users who take the loans rarely whereas the users taking loans frequently are less defaulters.
- Most importantly, the people are paying the amount early or lately and sometimes they might fail to pay within the time frame, but I observed that almost 80% of users are paying the amount within 7-8 days. It is recommended that to extent loan repayment time frame from 5 days to 7 days.
- Customer who takes a greater number of loans are non-defaulters (i.e., 98% of the category) as they repay the loan within the given time i.e., 5 days

Chapter 4

Feature Scaling

4.1 Correlation



From the above correlation graph it is clear that there is no negative correlation between any feature and target.

The highly correlated features are sumamnt_ma_rech90, cnt_ma_rech90, sumamnt_ma_rech30 and cnt_ma_rech30.

The lowest correlated feature is Day.

4.2 Splitting Data into target and features

```
Y = df_new['label']
X = df_new.drop(['label'], axis =1)
```

4.3 Feature Scaling

```
from sklearn.preprocessing import StandardScaler
```

```
sc= StandardScaler()
X_scale = sc.fit_transform(X)
```

4.4 Handling of Imbalanced Data

Since our target variable 'label' is imbalanced, we have to balance it. Here, I used the SMOTE oversampling Technique to balance the target variable.

```
df_new['label'].value_counts()
```

```
1    173461
0     24713
Name: label, dtype: int64
```

```
from imblearn.over_sampling import SMOTE
oversample = SMOTE()
X_scale, Y = oversample.fit_resample(X_scale, Y)
```

```
Y.value_counts()
```

```
0    173461
1    173461
Name: label, dtype: int64
```

Here we have successfully balanced our target variable

4.5 multi-Collinearity

To check the multicollinearity, I used the VIF method.

```
vif = pd.DataFrame()
vif["VIF values"] = [variance_inflation_factor(X_scale,i) for i in range(len(X.columns))]
vif["Features"] = X.columns
vif
```

For most of the independent features VIF exceed permissible limit of 10.

Also observed that

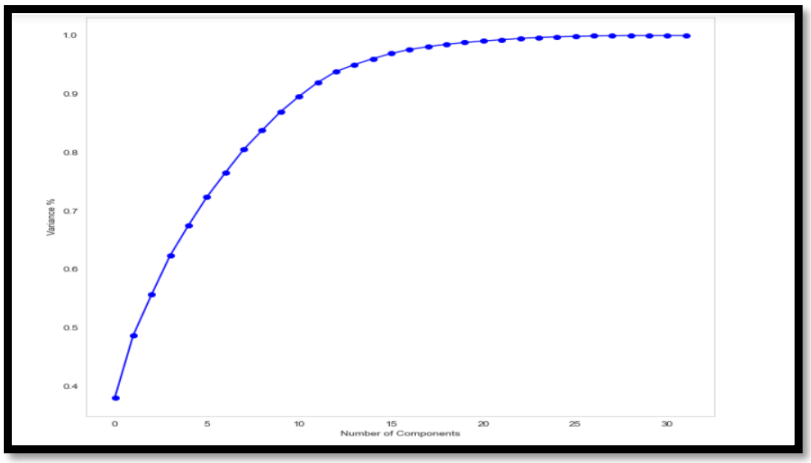
- daily_decr30 and daily_decr90 are highly correlated with each other.
- cnt_loans90 and amnt_loans90 are highly correlated with each other.
- cnt_loans30 and amnt_loans30 are highly correlated with each other.
- cnt_ma_rech30 and sumamnt_ma_rech30 are highly correlated with each other.

As most of input features are correlated with each other either moderated or poorly removing Some of highly correlated features will not work here. So, another way to deal with Multicollinearity is to Scale the Data and then apply PCA.

PCA ¶

```
from sklearn.decomposition import PCA
```

```
pca =PCA()
x_pca =pca.fit_transform(X_scale)
plt.figure(figsize=(10,10))
plt.plot(np.cumsum(pca.explained_variance_ratio_), 'bo-')
plt.xlabel('Number of Components')
plt.ylabel('Variance %')
plt.title('Explained variance Ratio')
plt.grid()
plt.show()
```



from the above graph we can observe that 11 principal components attribute for 90% of variation in the data. We shall pick the first 11 components for our prediction.

```
pca_new = PCA(n_components=11)
x_new = pca_new.fit_transform(X_scale)
```

```
principle_x=pd.DataFrame(x_new,columns=np.arange(11))
```

Chapter 5

Model Development and Evaluation

5.1. IDENTIFICATION OF POSSIBLE PROBLEM SOLVING APPROACHES (METHODS)

The target variable label has two classes i.e., label ’1’ indicates non defaulter & label ‘0’ indicates defaulter. Our objective is to predict whether customer is defaulter or not. This becomes binary classification problem which can be solved using various classification algorithms. In order to gain high accuracy of model we will train model with different classification model and select final model among them. To enhance performance of best model will employ hyper parameter tuning over it. At end we will save our final model using joblib.

5.2. Testing of Identified Approaches (Algorithms)

The different classification algorithm used in this project to build ML model are as below:

- Logistics Regression
- Random Forest Classifier
- Decision Tree Classifier
- K-Neighbours-Classifier
- Extra Tree Classifier

5.3. KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION

- Precision can be seen as a measure of quality; higher precision means that an algorithm returns more relevant results than irrelevant ones.
- Recall is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.
- Accuracy score is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.
- F1-score is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.
- Cross validation Score: To run cross-validation on multiple metrics and also to return train scores, fit times and score times. Get predictions from each split of cross-validation for diagnostic purposes. Make a scorer from a performance metric or loss function.
- AUC_ROC _score: ROC curve. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0

- We have used Accuracy Score and Cross validation score as key parameter for model evaluation in this project since balancing of data is performed.

5.4 Splitting Data into Training & test data sets

```
X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=99, test_size=.3)
print('Training feature matrix size:',X_train.shape)
print('Training target vector size:',Y_train.shape)
print('Test feature matrix size:',X_test.shape)
print('Test target vector size:',Y_test.shape)
```

```
Training feature matrix size: (242845, 11)
Training target vector size: (242845,)
Test feature matrix size: (104077, 11)
Test target vector size: (104077,)
```

5.5 Finding Best Random State

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report,f1_score
maxAccu=0
maxRS=0
for i in range(0,299):
    X_train,X_test,Y_train,Y_test = train_test_split(principle_x,Y,test_size = 0.3, random_state=i)
    lr=LogisticRegression()
    lr.fit(X_train,Y_train)
    y_pred=lr.predict(X_test)
    acc=accuracy_score(Y_test,y_pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print('Best accuracy is', maxAccu ,'on Random_state', maxRS)
```

```
Best accuracy is 0.76627881280206 on Random_state 244
```

Here, we got 244 as best random state and the best accuracy at random state 244 is 0.7662788.

5.6 Run & Evaluate the selected algorithms

1. Logistic Regression

```
X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=244, test_size=.3)
lr=LogisticRegression()
lr.fit(X_train,Y_train)
y_pred_lr=lr.predict(X_test)
print('Logistics Regression')
print('\n')
print('Accuracy Score of Logistics Regression : ', accuracy_score(Y_test, y_pred_lr))
print('\n')
print('Confusion matrix of Logistics Regression :','\n',confusion_matrix(Y_test, y_pred_lr))
print('\n')
print('classification Report of Logistics Regression','\n',classification_report(Y_test, y_pred_lr))
print('cross validation score of Logistics Regression',cross_val_score(LogisticRegression(),principle_x, Y,cv=10).mean())
```

The results of Logistic regression algorithm is

```
Accuracy Score of Logistics Regression :  0.76627881280206


Confusion matrix of Logistics Regression :
 [[39821 12394]
 [11931 39931]]


classification Report of Logistics Regression
              precision    recall  f1-score   support

           0       0.77      0.76      0.77     52215
           1       0.76      0.77      0.77     51862

    accuracy                           0.77    104077
   macro avg       0.77      0.77      0.77    104077
weighted avg       0.77      0.77      0.77    104077

cross validation score of Logistics Regression 0.762866016860267
```

2. Random Forest Classifier

```
X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=244, test_size=.3)
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train,Y_train)
y_pred_rf=rf.predict(X_test)
print('Random Forest Classifier Evaluation')
print('\n')
print('Accuracy Score of Random Forest Classifier :','\n', accuracy_score(Y_test, y_pred_rf))
print('\n')
print('Confusion matrix of Random Forest Classifier :','\n',confusion_matrix(Y_test, y_pred_rf))
print('\n')
print('classification Report of Random Forest Classifier','\n',classification_report(Y_test, y_pred_rf))
print('cross validation score of Random Forest Classifier',cross_val_score(RandomForestClassifier(),principle_x, Y,cv=10).me
```

```
Accuracy Score of Random Forest Classifier :
 0.9205107756757016


Confusion matrix of Random Forest Classifier :
 [[48758  3457]
 [ 4816 47046]]


classification Report of Random Forest Classifier
              precision    recall  f1-score   support

           0       0.91      0.93      0.92     52215
           1       0.93      0.91      0.92     51862

    accuracy                           0.92    104077
   macro avg       0.92      0.92      0.92    104077
weighted avg       0.92      0.92      0.92    104077

cross validation score of Random Forest Classifier 0.930044802661991
```

3. Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=244, test_size=.3)
dt = DecisionTreeClassifier()
dt.fit(X_train,Y_train)
y_pred_dt=dt.predict(X_test)
print('DecisionTreeClassifier Evaluation')
print('\n')
print('Accuracy Score of DecisionTreeClassifier :', accuracy_score(Y_test, y_pred_dt))
print('\n')
print('Confusion matrix of DecisionTreeClassifier :','\n',confusion_matrix(Y_test, y_pred_dt))
print('\n')
print('classification Report of DecisionTreeClassifier','\n',classification_report(Y_test, y_pred_dt))
print('cross validation score of Decision Tree Classifier',cross_val_score(DecisionTreeClassifier(),principle_x, Y,cv=10).me
```

```
Accuracy Score of DecisionTreeClassifier : 0.8559912372570309


Confusion matrix of DecisionTreeClassifier :
 [[45596  6619]
 [ 8369 43493]]


classification Report of DecisionTreeClassifier
              precision    recall  f1-score   support

           0       0.84      0.87      0.86     52215
           1       0.87      0.84      0.85     51862

    accuracy                           0.86    104077
   macro avg       0.86      0.86      0.86    104077
weighted avg       0.86      0.86      0.86    104077

cross validation score of Decision Tree Classifier 0.8690224229965484
```

4. K-Neighbours Classifier

```
X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=153, test_size=.3)
knn = KNeighborsClassifier()
knn.fit(X_train,Y_train)
y_pred_knn=knn.predict(X_test)
print('KNeighborsClassifier Evaluation')
print('\n')
print('Accuracy Score of KNeighborsClassifier :', accuracy_score(Y_test, y_pred_knn))
print('\n')
print('Confusion matrix of KNeighborsClassifier :','\n',confusion_matrix(Y_test, y_pred_knn))
print('\n')
print('classification Report of KNeighborsClassifier','\n',classification_report(Y_test, y_pred_knn))
print('cross validation score of KNeighborsClassifier',cross_val_score(KNeighborsClassifier(),principle_x, Y,cv=10).mean())
```

```
Accuracy Score of KNeighborsClassifier : 0.8900717737828723


Confusion matrix of KNeighborsClassifier :
 [[51017   989]
 [10452 41619]]


classification Report of KNeighborsClassifier
              precision    recall  f1-score   support

           0       0.83      0.98      0.90     52006
           1       0.98      0.80      0.88     52071

    accuracy                           0.89    104077
   macro avg       0.90      0.89      0.89    104077
weighted avg       0.90      0.89      0.89    104077

cross validation score of KNeighborsClassifier 0.9011247604205768
```

5. Extra Tree Classifier

```
X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=153, test_size=.3)
xt = ExtraTreesClassifier()
xt.fit(X_train,Y_train)
y_pred_xt=xt.predict(X_test)
print('ExtraTreesClassifier Evaluation')
print('\n')
print('Accuracy Score of ExtraTreesClassifier :','\n', accuracy_score(Y_test, y_pred_xt))
print('\n')
print('Confusion matrix of ExtraTreesClassifier :','\n',confusion_matrix(Y_test, y_pred_xt))
print('\n')
print('classification Report of ExtraTreesClassifier','\n',classification_report(Y_test, y_pred_xt))
print('cross validation score of ExtraTreesClassifier',cross_val_score(ExtraTreesClassifier(),principle_x, Y,cv=10).mean())
```

```
Accuracy Score of ExtraTreesClassifier :
 0.9345676758553763


Confusion matrix of ExtraTreesClassifier :
 [[49421  2585]
 [ 4225 47846]]


classification Report of ExtraTreesClassifier
              precision    recall  f1-score   support

           0       0.92      0.95      0.94     52006
           1       0.95      0.92      0.93     52071

    accuracy                           0.93    104077
   macro avg       0.93      0.93      0.93    104077
weighted avg       0.94      0.93      0.93    104077

cross validation score of ExtraTreesClassifier 0.9440796377205805
```

Accuracy Scores of all used models

Algorithm	Accuracy Score	CV Score
Logistic Regression	0.7659040	0.762442
Random Forest Classifier	0.919530	0.929915
Decision Tree Classifier	0.857470	0.868546
K-Neighbours Classifier	0.891013	0.900608
Extra Tree Classifier	0.935749	0.943825

Since the Random Forest Classifier model gives best accuracy score with minimum difference between accuracy score & cross validation score.

Hence, we will select this one as final model and will do Hyperparameter tuning for the same.

Hyperparameter Optimization

```
from sklearn.model_selection import GridSearchCV
```

```
n_estimators = [2,12,22,32,42]
criterion_list = ['gini','entropy']
max_features = ["auto",'log']
max_depth = [2,5]
min_samples_split = [3,4]
min_samples_leaf = [1,6]
bootstrap = ['true','false']
```

```
param_grid ={'n_estimators':n_estimators,
    'max_features': max_features,
    'max_depth': max_depth,
    'min_samples_split': min_samples_split,
     'min_samples_leaf': min_samples_leaf,
     'bootstrap': bootstrap}
```

```
clf = GridSearchCV(rf,param_grid = param_grid,cv = 10,verbose = True,n_jobs = -1)
```

```
best_clf = clf.fit(principle_x, Y)
```

```
Fitting 10 folds for each of 160 candidates, totalling 1600 fits
```

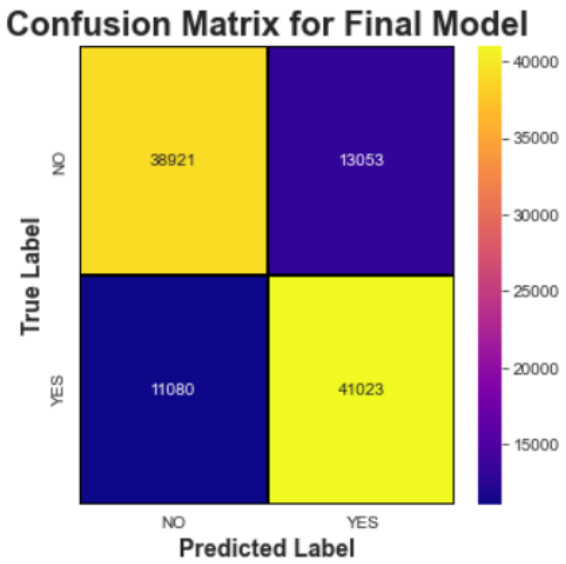
```
best_clf.best_estimator_
```

```
RandomForestClassifier(bootstrap='false', max_depth=5, max_features='auto',
                       min_samples_split=3, n_estimators=12)
```

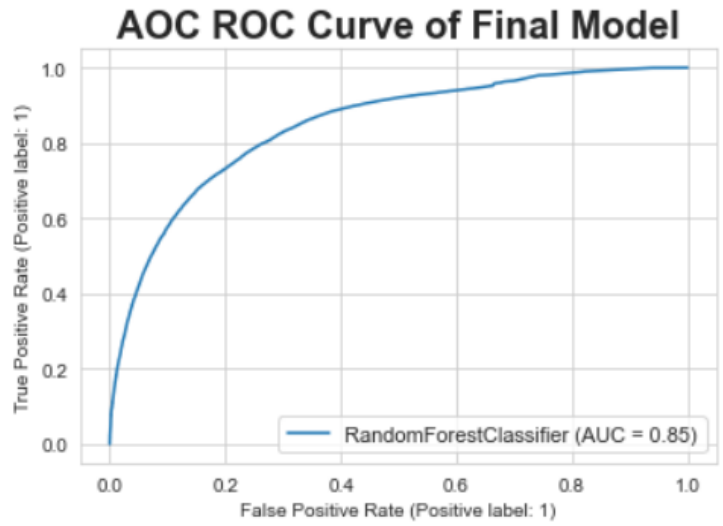
```
Final_mod = RandomForestClassifier(bootstrap='true',n_estimators= 32, max_depth=5 ,
                                   min_samples_split= 3, max_features= 'auto')
Final_mod.fit(X_train,Y_train)
y_pred=Final_mod.predict(X_test)
print('\033[1m'+'Accuracy Score :'+'\033[0m\n', accuracy_score(Y_test, y_pred))
```

```
Accuracy Score :
 0.7681236007955649
```

```
# Lets plot confusion matrix for  FinalModel
Matrix = confusion_matrix(Y_test, y_pred)

x_labels = ["NO","YES"]
y_labels = ["NO","YES"]

fig , ax = plt.subplots(figsize=(5,5))
sns.heatmap(Matrix, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax,
            cmap="plasma", xticklabels = x_labels, yticklabels = y_labels)

plt.xlabel("Predicted Label",fontsize=14,fontweight='bold')
plt.ylabel("True Label",fontsize=14,fontweight='bold')
plt.title('Confusion Matrix for Final Model',fontsize=20,fontweight='bold')
plt.show()
```



```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import plot_roc_curve
disp = plot_roc_curve(Final_mod,X_test,Y_test)
plt.legend(prop={'size':11}, loc='lower right')
plt.title('AOC ROC Curve of Final Model',fontsize=20,fontweight='bold')
plt.show()
auc_score = roc_auc_score(Y_test, Final_mod.predict(X_test))
print('\033[1m'+'Auc Score :'+'\033[0m\n',auc_score)
```



Auc Score :
0.7680997478109799

5.7 Final Model Saving

```
import joblib
joblib.dump(Final_mod,'Insurance_claims_Final.pkl')
```

```
# Prediction
prediction = Final_mod.predict(X_test)
```

```
Actual = np.array(Y_test)
df_Pred = pd.DataFrame()
df_Pred["Predicted Values"] = prediction
df_Pred["Actual Values"] = Actual
df_Pred.head(10)
```

Predictions on Test data

	Predicted Values	Actual Values
0	1	1
1	0	0
2	0	0
3	1	1
4	1	1
5	1	1
6	0	1
7	1	1
8	0	0
9	0	0

Chapter 6

Conclusion

- Random Forest Classifier Hyper parameter tuned gives maximum accuracy score of 0.919530 with cross validation score of 0.929915. It also gives us maximum AUC score.
- 12.5 % customers are defaulters out of whole dataset.
- Tendency to pay loan within 5 days is high among customer take loan many times within month compare to those who take loan 1-2 times.
- It is recommended that to extent loan repayment time frame from 5 days to 7 days.