MICRO CREDIT DEFAULTER PROJECT

Submitted by:

MONIKA SINGH

1. Introduction

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.

Conceptual Background of the Domain Problem

Telecom Industries 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.

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 payed i.e. Non- defaulter, while, Label '0' indicates that the loan has not been payed i.e. defaulter.

Review of Literature

Microfinance, also called 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. Microfinance is usually understood to entail the provision of financial services to micro entrepreneurs and small businesses, which lack access to banking and related services due to the high transaction costs associated with serving these client categories.

No doubt, microfinance is very powerful tool to alleviate the poverty and empowerment to women and poor populations. The first organization to receive attention was the Grameen Bank, which was started in 1983 by Muhammad Yunus in Bangladesh. In 2006, the Nobel Peace Prize was awarded to both Yunus and the Grameen Bank for their efforts in developing the microfinance system.

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

- Delivery of very small loans to unsalaried workers
- The loan tenure is short
- Microfinance loans do not require any collateral
- These loans are usually repaid at higher frequencies

Default in Microfinance:

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.

Machine learning-based systems are growing in popularity in research applications in most disciplines. Classification is an essential form of data analysis in data mining that formulates models while describing significant data classes. 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.

2. Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

The goal of this project is to predict the defaulter of Micro Finance Loan with the help of classification- based algorithm. In this project different types of algorithms are used. The algorithms are used with their own mathematical equation on background.

This project have one big set of data for training & testing. Primarily, the dataset is imbalanced. We need to balance the dataset by over sampling method (SMOTE). Then different steps of data pre-processing is performed over the training and testing data like data cleaning, data visualization, relationship between features with label. After this, unnecessary feature are removed. In model building. Final model is select based on Accuracy score, AUC score and Cross Validation score of different algorithms.

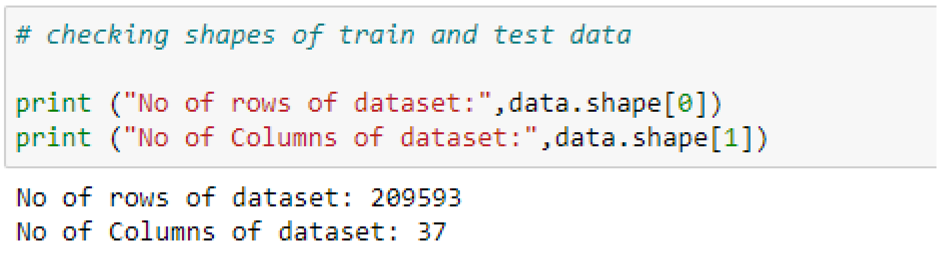
Here 6 different algorithm are used.

Data Sources and their formats

The given dataset is in csv form.

The data is used to train the model. It has 209593 rows and 37 columns. The model will train with the help of this dataset. It has 36 independent features and one dependent or target variable (label)

Later the dataset is divided into two parts, training and testing. After determine the proper model, the model is applied to predict the target variable for the test data.



```
# checking shapes of train and test data

print ("No of rows of dataset:",data.shape[0])
print ("No of Columns of dataset:",data.shape[1])
```

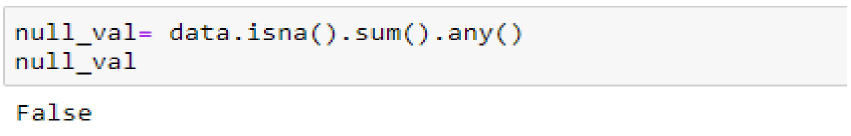
```
No of rows of dataset: 209593
No of Columns of dataset: 37
```

To determine the data format, info() method is used. There are total 3 categorical columns among 37 columns.

Data Pre-processing Done:

Checking Null:

Null values are not present in the total dataset.

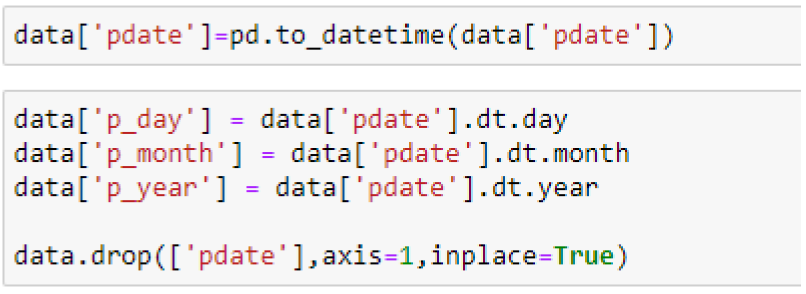


```
null_val= data.isna().sum().any()
null_val
```

```
False
```

Date Datatype (pdate):

pdate is an object datatype. Need to convert in in date format.



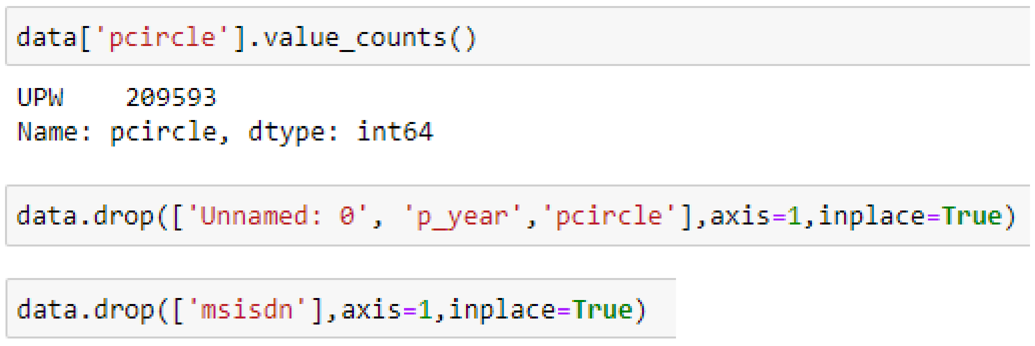
```
data['pdate']=pd.to_datetime(data['pdate'])
```

```
data['p_day'] = data['pdate'].dt.day
data['p_month'] = data['pdate'].dt.month
data['p_year'] = data['pdate'].dt.year

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

Drop unnecessary data:

Here pcircle has one unique value in the whole dataset. All p_year is single unique value which is 2016. Also Unnamed: 0 is an unnecessary column because it has different value for every set.

Here one categorical feature is available which is msisdn: mobile number of user. No need to keep it as it has no importance to determine label. Let's drop it also.



```
data['pcircle'].value_counts()
```

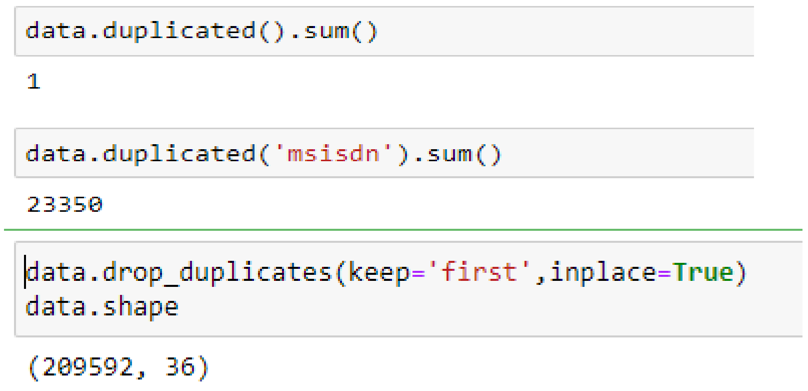
```
UPW    209593
Name: pcircle, dtype: int64
```

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

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

Checking duplicate:

Let’s check duplicate entry in the total dataset. There 1 row with completely duplicate values of all features. There 23350 rows with duplicate mobile number. Seems same mobile number is switch to other customers.



```
data.duplicated().sum()
```

```
1
```

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

```
23350
```

```
data.drop_duplicates(keep='first',inplace=True)
data.shape
```

```
(209592, 36)
```

Removal of Negative values:

Some features has negative values in minimum column. They are aon, daily_decr30, daily_decr90, rental30, rental90, last_rech_date_ma, last_rech_date_da, medianmarechprebal90, medianmarechprebal30. Let's explore those columns further.

```
data['aon'].describe()
```

```
count    209592.000000
mean       8112.380399
std       75696.261220
min         -48.000000
25%         246.000000
50%         527.000000
75%         982.000000
max      999860.755168
Name: aon, dtype: float64
```

Here maximum is 999860.755168 and the minimum is -48. Let's convert maximum value in years. The minimum value is wrongly entered.

```
print( "Maximum value of 'aon' in years :", 999860/365)
```

```
Maximum value of 'aon' in years : 2739.3424657534247
```

Maximum value of 'aon' in years is 2739.34 year which is not possible at all. There are 1539 entries with value less than 0. In this particular column, a huge problem is raised due to wrong entry of data. Let's minimize the chance of error with some assumptions:

- All negative values are typing by mistake "-" (minus) in front of original value. Negative values are converted into absolute value (positive) to correct error.
- Upper limit of these features handle by outlier removal.

Let's apply this technique to all negative minimum values column.

```
print(" No of entries with negetive aon value: ")
data[data['aon']<0].value_counts().sum()
```

```
 No of entries with negetive aon value: 

1539
```

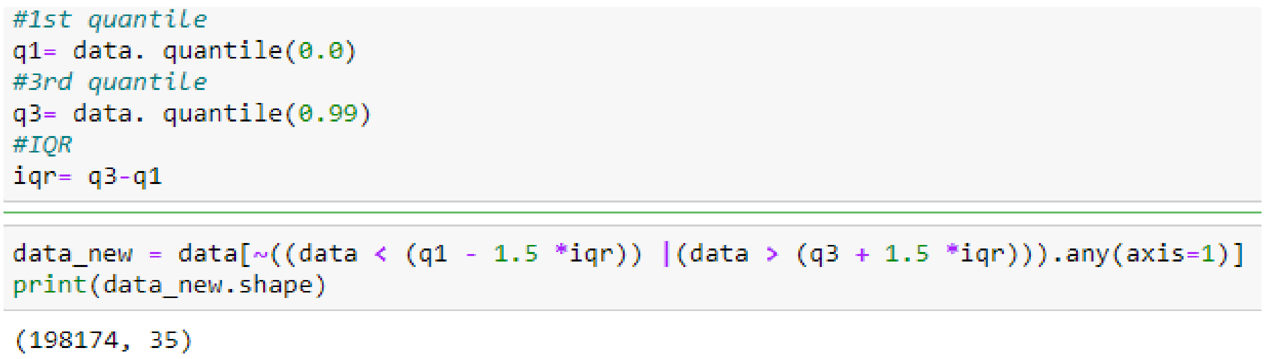
```
data['daily_decr30']=abs(data['daily_decr30'])
data['daily_decr90']=abs(data['daily_decr90'])
data['rental30']=abs(data['rental30'])
data['rental90']=abs(data['rental90'])
data['last_rech_date_ma']=abs(data['last_rech_date_ma'])
data['last_rech_date_da']=abs(data['last_rech_date_da'])
```

All negative min value except medianmarechprebal90, medianmarechprebal30 are deleted by replacing with abs of corresponding values. But medianmarechprebal90, medianmarechprebal30 are still negative. The median of main account balance can be negative. So let's keep this.

Now as maxamnt_loans30 and maxamnt_loans90 are same type of datatype. The data structure of this two column are same. So it is clear that, the repay amount be 0,6,12. Let's replace values greater than 12 into category of zero. The assumption taken is: loan amount greater than 12 will be replaced by zero (0).

Removal of outliers:

When we apply Z score method, around 23.5% data are deleted. But As per problem statement, **data is expensive and we cannot lose more than 7-8% of** the data. Now we use Quantile-based technique. But as the dataset only outliers at upper side. It has no lower capping level. Let's take 1st quantile (lower capping level) as 0% and 3rd quantile (upper capping level) as 99%.



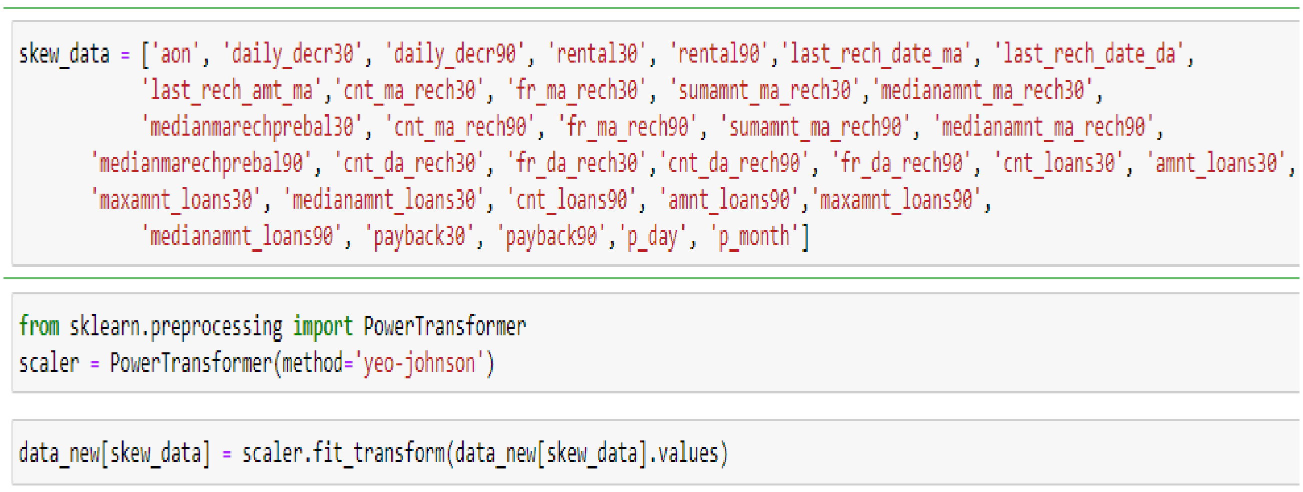
```
#1st quantile
q1= data. quantile(0.0)
#3rd quantile
q3= data. quantile(0.99)
#IQR
iqr= q3-q1
```

```
data_new = data[~((data < (q1 - 1.5 *iqr)) |(data > (q3 + 1.5 *iqr))).any(axis=1)]
print(data_new.shape)
```

```
(198174, 35)
```

Skewness:

Skewness is present in all features including target. Let's remove the skewness except label as it is the target variable. Let's use Power Transformer to transform skewness in features.



```
skew_data = ['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','p_day', 'p_month']
```

```
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method='yeo-johnson')
```

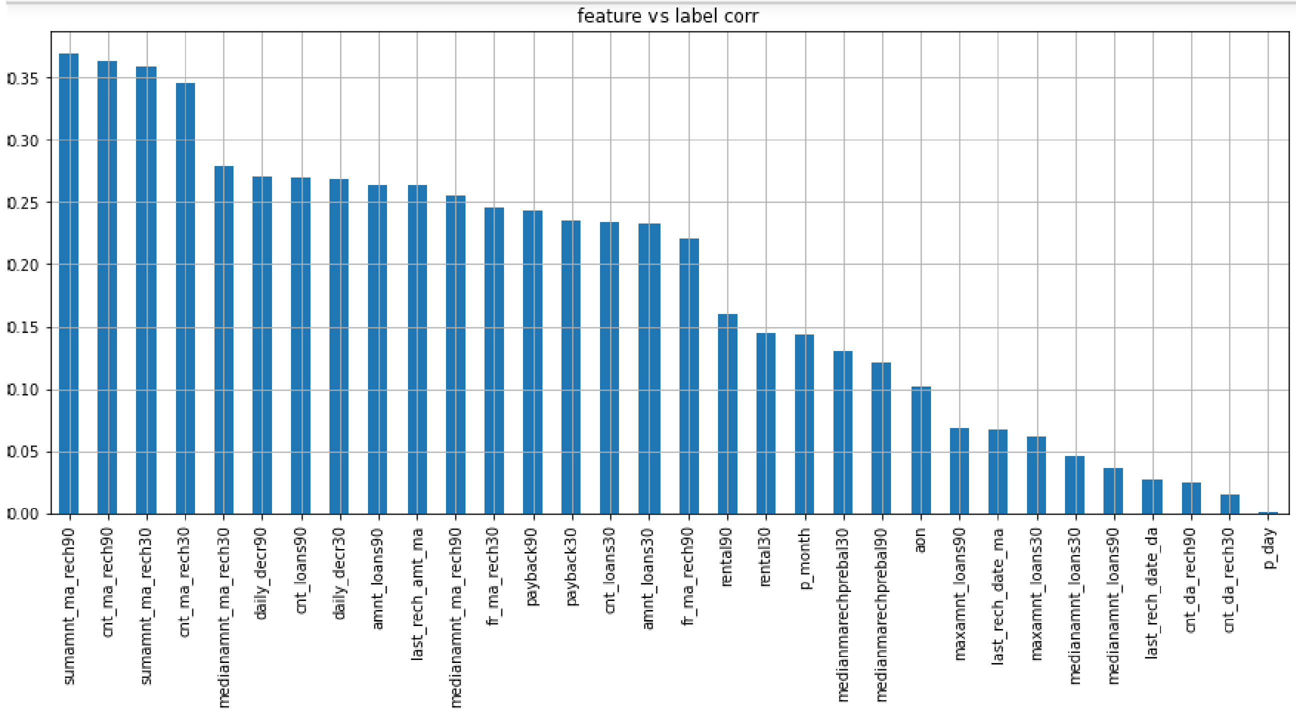
```
data_new[skew_data] = scaler.fit_transform(data_new[skew_data].values)
```

Correlation:

As are unnecessary columns, let's remove it. It has no correlation with others.

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

```
data_new.drop("label", axis=1).corrwith(data_new["label"]).sort_values(ascending=False).plot(kind='bar',
        figsize=(15,6),grid='True',title='feature vs label corr')
plt.show()
```



sumamnt_ma_rech90, cnt_ma_rech90, sumamnt_ma_rech30, cnt_ma_rech30 are highly correlated with each other

Data Balancing (SMOTE):

Let's balance the target variable label by SMOTE().

```
from imblearn.over_sampling import SMOTE
over = SMOTE()

# splitting data in target and dependent feature
x = data_new.drop(['label'], axis =1)
y = data_new['label']
x,y = over.fit_resample(x,y)
y.value_counts()
```

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

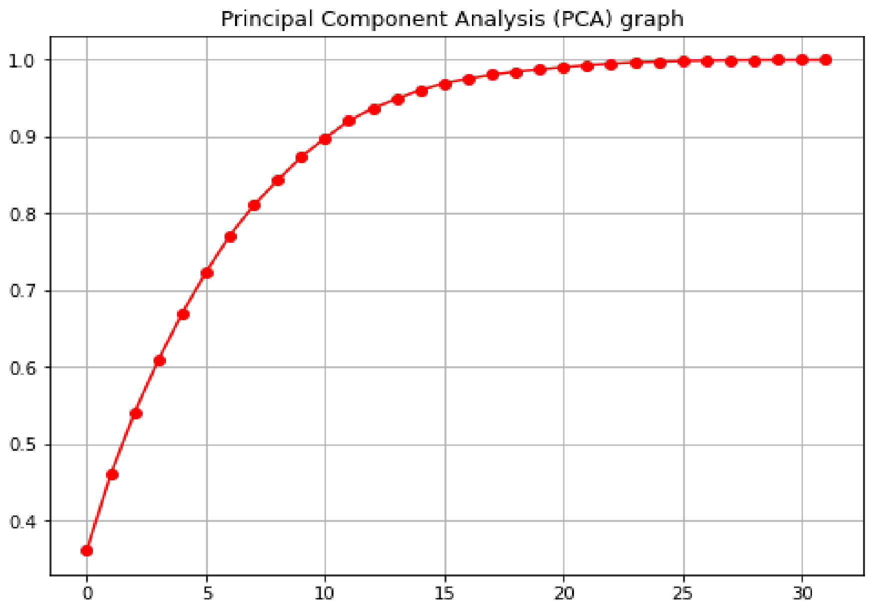
Checking Multicollinearity:

Let’s check multicollinearity. Multicollinearity is present in this dataset as for most independent feature VIF is exceed permissible limit of 10. PCA is applied to remove multicollinearity among features.

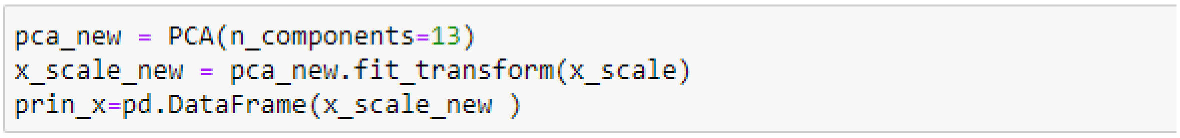
```
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif= pd.DataFrame()
vif["VIF"]= [variance_inflation_factor(x.values,i)for i in range(x.shape[1])]
vif["Features"] = x.columns
vif
```

Principal Component Analysis (PCA):

First check Principal Component. Remove the multicollinearity after checking PCA.



We can see that 13 principal components attribute for around 95% of variation in the data. We shall pick the first 13 components for our prediction.



```
pca_new = PCA(n_components=13)
x_scale_new = pca_new.fit_transform(x_scale)
prin_x=pd.DataFrame(x_scale_new )
```

Data Inputs- Logic- Output Relationships

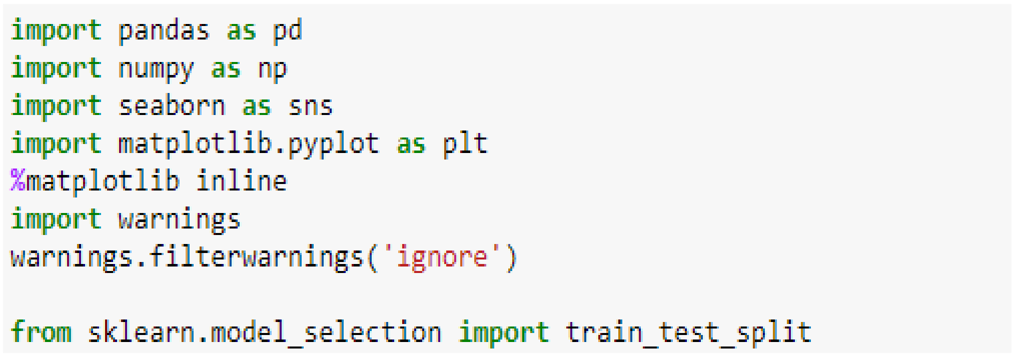
We can see in the correlation that every features are correlated with each other and also they are highly correlated with target variable label.

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

- The maximum value in maxamnt_loans30 is 12.
- All negative values are typing error happen accidentally by type - in front of original value (except feature depicting median).

Hardware and Software Requirements and Tools Used

Processor: Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz 2.00 GHz
RAM: 4.00 GB
System Type: 64-bit operating system, x64-based processor
Window: Windows 10 Pro
Anaconda – Jupyter Notebook
Libraries Used –



```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
```

Except this, different libraries are used for machine learning model building from sklear

3. Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

In this problem classification-based machine learning algorithm like logistic regression can be used. For that first data encoding and data scaling using standard scalar is done. For building an appropriate ML model before implementing classification algorithms, data is split in training & test data using train_test_split. Then different statistical parameter like Accuracy score, training score, testing score, classification report, etc. are determined for every algorithm. Hyper parameter tuning is performed to get the accuracy score much higher and accurate than earlier.

Then cross-validation is done to check best CV Score. AUC-ROC graph is plotted and the best curve is chosen from 6 different algorithm. Then final model is determined.

Testing of Identified Approaches (Algorithms)

Total 6 algorithms used for the training and testing are:

1. Logistic Regression.
2. DecisionTreeClassifier
3. GradientBoostingClassifier
4. RandomForestClassifier
5. ExtraTreesClassifier
6. AdaBoostClassifier

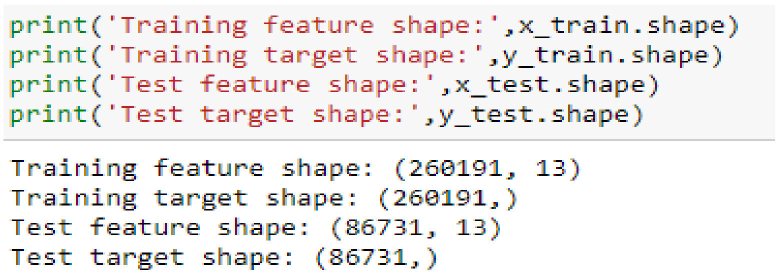
Key Metrics for success in solving problem under consideration:

From metrics module of sklearn library import classification_report, accuracy_score, confusion_matrix, classification_report and f1_score. From model_selection also, we use cross_val_score. Those are the matrices use to validate the model's quality. Let's discuss every metrics shortly.

- Classification report: It is a performance evaluation metric in machine learning which is used to show the precision, recall, F1 Score, and support score of your trained classification model
- Accuracy score: It is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.
- Confusion Matrix: It is a table that is used in classification problems to assess where errors in the model were made. The rows represent the actual classes the outcomes should have been. While the columns represent the predictions we have made. Using this table it is easy to see which predictions are wrong.
- Precision: It can be seen as a measure of quality. If the precision is high, an algorithm returns more relevant results than irrelevant ones.
- Recall: The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples.
- F1 Score: F1 = 2 * (precision * recall) / (precision + recall)

Run and Evaluate selected models

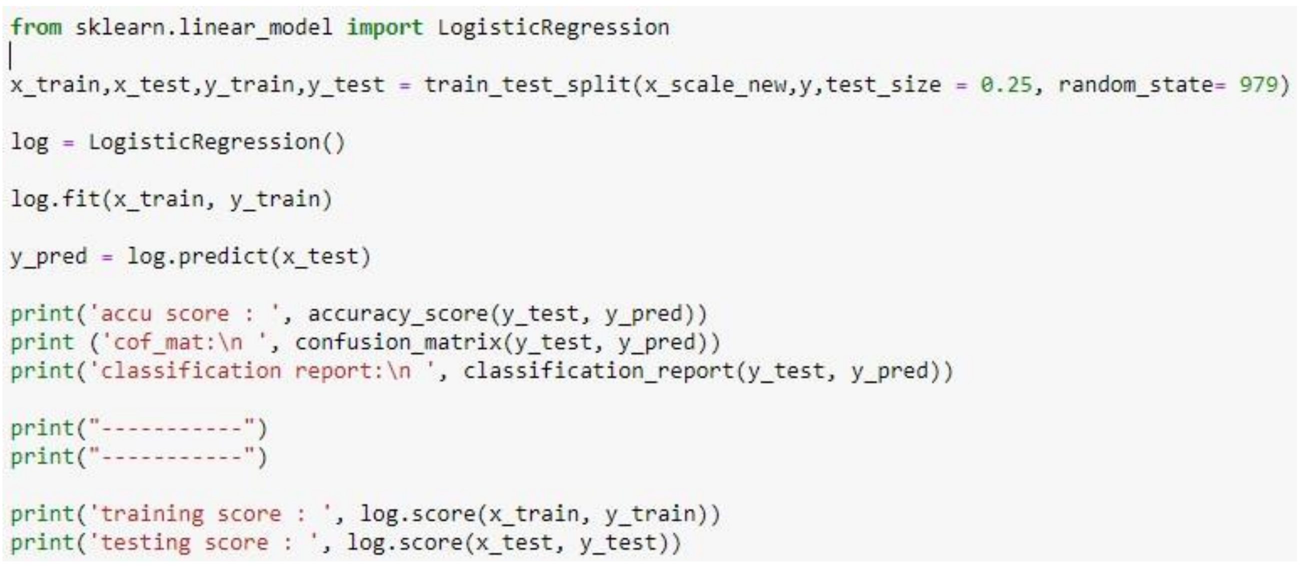
First find the best random state of train_test_split to get best accuracy. Here the random state is 979. Then after splitting the data into 4 different part and check the shape of the data.



```
print('Training feature shape:',x_train.shape)
print('Training target shape:',y_train.shape)
print('Test feature shape:',x_test.shape)
print('Test target shape:',y_test.shape)
```

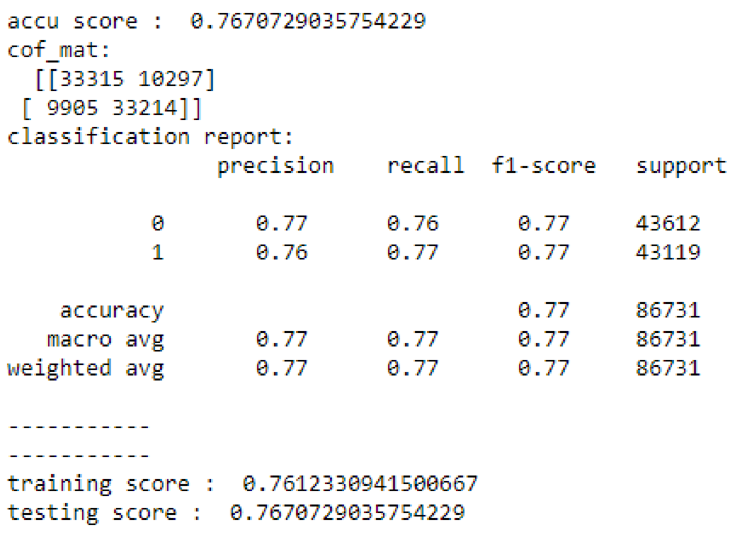
```
Training feature shape: (260191, 13)
Training target shape: (260191,)
Test feature shape: (86731, 13)
Test target shape: (86731,)
```

A Logistic Regression:



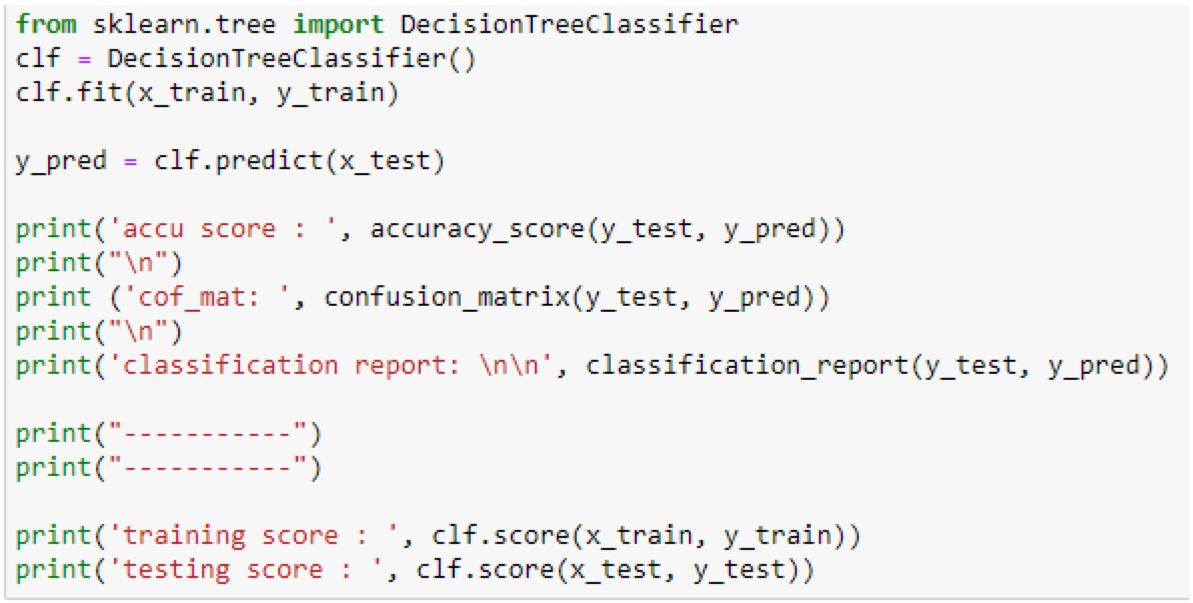
```
from sklearn.linear_model import LogisticRegression
|
x_train,x_test,y_train,y_test = train_test_split(x_scale_new,y,test_size = 0.25, random_state= 979)

log = LogisticRegression()

log.fit(x_train, y_train)

y_pred = log.predict(x_test)

print('accu score : ', accuracy_score(y_test, y_pred))
print ('cof_mat:\n ', confusion_matrix(y_test, y_pred))
print('classification report:\n ', classification_report(y_test, y_pred))

print("-----------")
print("-----------")

print('training score : ', log.score(x_train, y_train))
print('testing score : ', log.score(x_test, y_test))
```

The matric score is as follows.



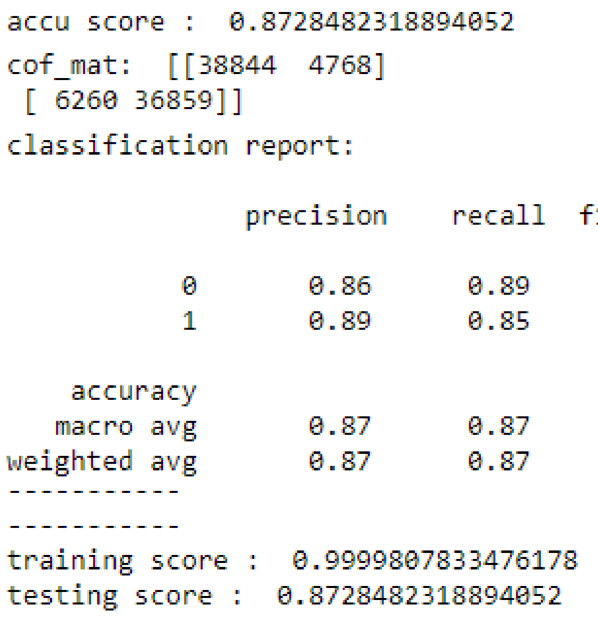
```
accu score :  0.7670729035754229
cof_mat:
  [[33315 10297]
 [ 9905 33214]]
classification report:
               precision    recall  f1-score   support

           0       0.77      0.76      0.77     43612
           1       0.76      0.77      0.77     43119

    accuracy                           0.77     86731
   macro avg       0.77      0.77      0.77     86731
weighted avg       0.77      0.77      0.77     86731

-----------
-----------
training score :  0.7612330941500667
testing score :  0.7670729035754229
```

B DecisionTreeClassifier:



```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(x_train, y_train)

y_pred = clf.predict(x_test)

print('accu score : ', accuracy_score(y_test, y_pred))
print("\n")
print ('cof_mat: ', confusion_matrix(y_test, y_pred))
print("\n")
print('classification report: \n\n', classification_report(y_test, y_pred))

print("-----------")
print("-----------")

print('training score : ', clf.score(x_train, y_train))
print('testing score : ', clf.score(x_test, y_test))
```

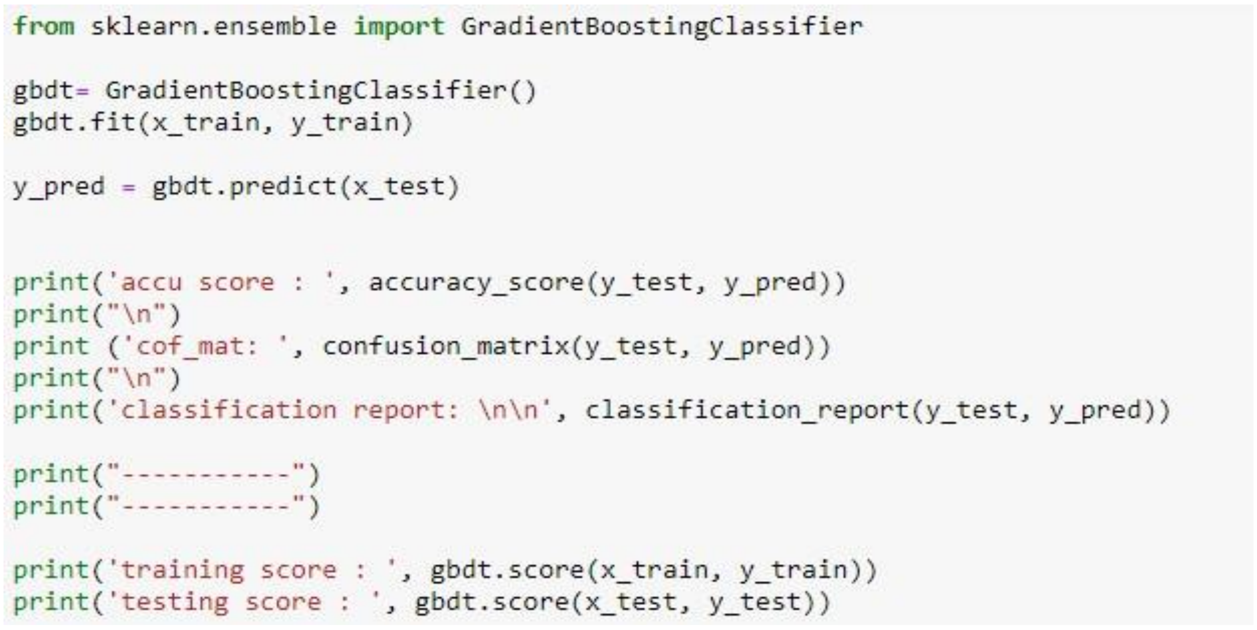
The matric score is as follows.



```
accu score :  0.8728482318894052
cof_mat:  [[38844  4768]
 [ 6260 36859]]
classification report:

              precision    recall  f1-score   support

           0       0.86      0.89      0.88     43612
           1       0.89      0.85      0.87     43119

    accuracy                           0.87     86731
   macro avg       0.87      0.87      0.87     86731
weighted avg       0.87      0.87      0.87     86731
-----------
-----------
training score :  0.9999807833476178
testing score :  0.8728482318894052
```

c. GradientBoostingClassifier:

The matric score is as follows.

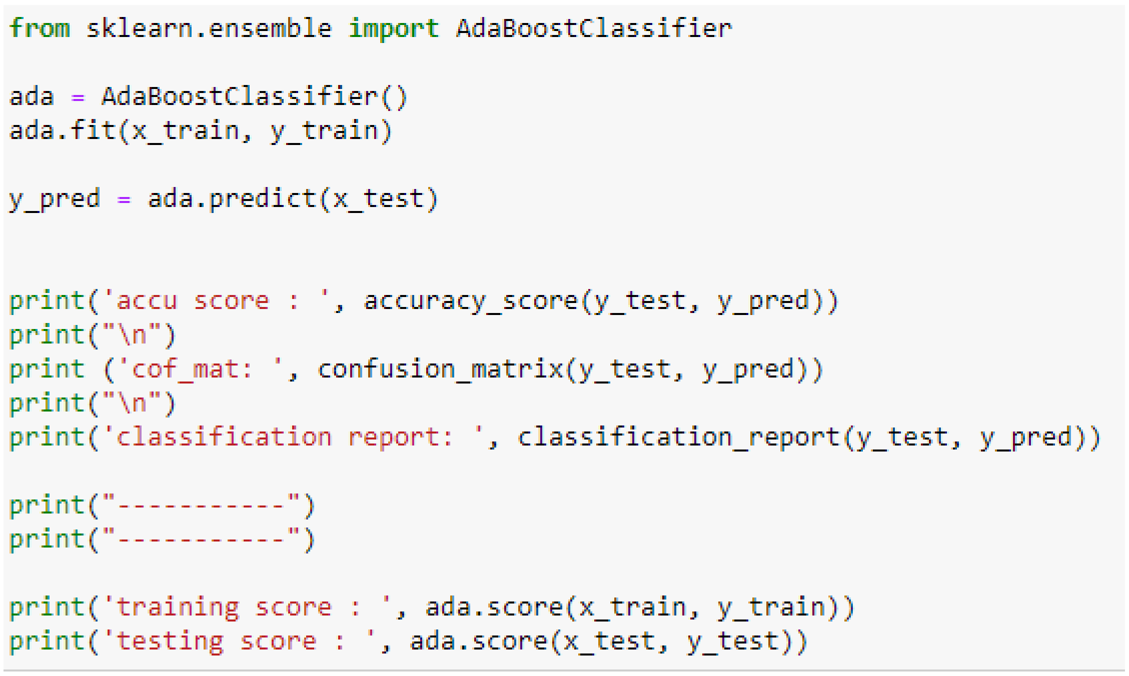


```
from sklearn.ensemble import GradientBoostingClassifier

gbdt= GradientBoostingClassifier()
gbdt.fit(x_train, y_train)

y_pred = gbdt.predict(x_test)


print('accu score : ', accuracy_score(y_test, y_pred))
print("\n")
print ('cof_mat: ', confusion_matrix(y_test, y_pred))
print("\n")
print('classification report: \n\n', classification_report(y_test, y_pred))

print("-----------")
print("-----------")

print('training score : ', gbdt.score(x_train, y_train))
print('testing score : ', gbdt.score(x_test, y_test))
```

The matric score is as follows.

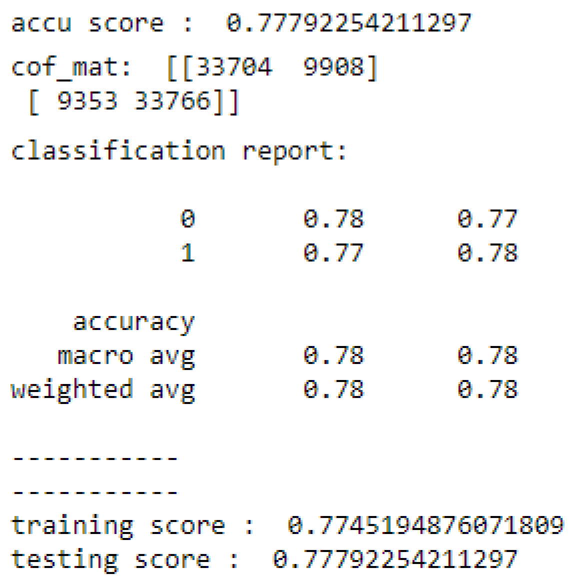
C AdaBoostClassifier:

D AdaBoostClassifier:



```
from sklearn.ensemble import AdaBoostClassifier

ada = AdaBoostClassifier()
ada.fit(x_train, y_train)

y_pred = ada.predict(x_test)


print('accu score : ', accuracy_score(y_test, y_pred))
print("\n")
print ('cof_mat: ', confusion_matrix(y_test, y_pred))
print("\n")
print('classification report: ', classification_report(y_test, y_pred))

print("-----------")
print("-----------")

print('training score : ', ada.score(x_train, y_train))
print('testing score : ', ada.score(x_test, y_test))
```

The matric score is as follows.

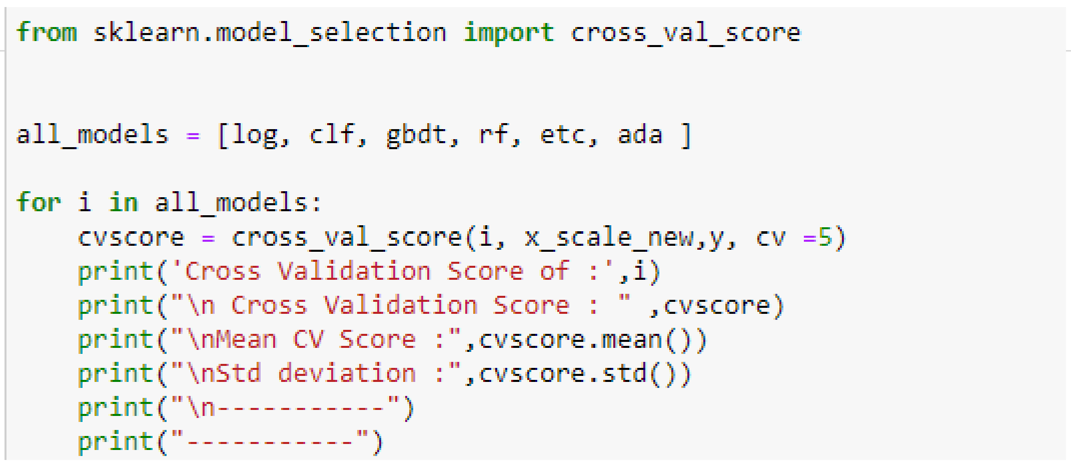


```
accu score :  0.77792254211297
cof_mat:  [[33704  9908]
 [ 9353 33766]]
classification report:               precision    recall  f1-score   support

           0       0.78      0.77      0.78     43612
           1       0.77      0.78      0.78     43119

    accuracy                           0.78     86731
   macro avg       0.78      0.78      0.78     86731
weighted avg       0.78      0.78      0.78     86731

-----------
-----------
training score :  0.7745194876071809
testing score :  0.77792254211297
```

As per 6 different regression model we can see that the model with maximum accuracy score and is ExtraTreesClassifier().

Cross validation:

Let's check the cross validation score taking cross fold =5, before final prediction. Here also ExtraTreesClassifier() is the best model with max cv score and min standard deviation.



```
from sklearn.model_selection import cross_val_score


all_models = [log, clf, gbdt, rf, etc, ada ]

for i in all_models:
    cvscore = cross_val_score(i, x_scale_new,y, cv =5)
    print('Cross Validation Score of :',i)
    print("\n Cross Validation Score : " ,cvscore)
    print("\nMean CV Score :",cvscore.mean())
    print("\nStd deviation :",cvscore.std())
    print("\n-----------")
    print("-----------")
```

AUC-ROC Curve:

Cross validation:

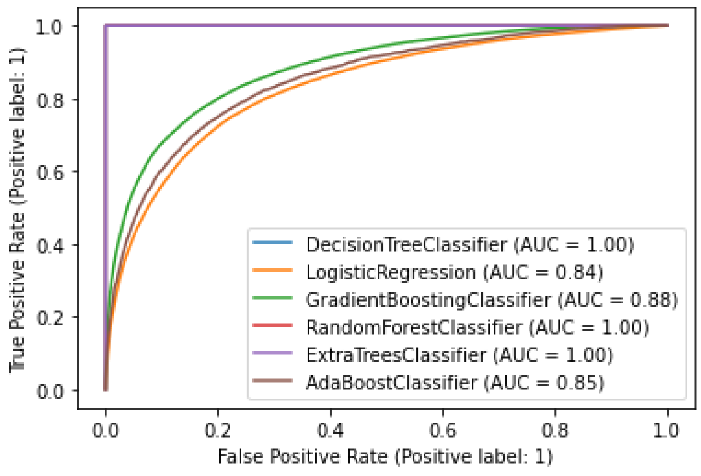
Let's check the cross validation score taking cross fold =5, before final prediction. Here also ExtraTreesClassifier() is the best model with max cv score and min standard deviation.

```
from sklearn.model_selection import cross_val_score


all_models = [log, clf, gbdt, rf, etc, ada ]

for i in all_models:
    cvscore = cross_val_score(i, x_scale_new,y, cv =5)
    print('Cross Validation Score of :',i)
    print("\n Cross Validation Score : " ,cvscore)
    print("\nMean CV Score :",cvscore.mean())
    print("\nStd deviation :",cvscore.std())
    print("\n-----------")
    print("-----------")
```

AUC-ROC Curve:

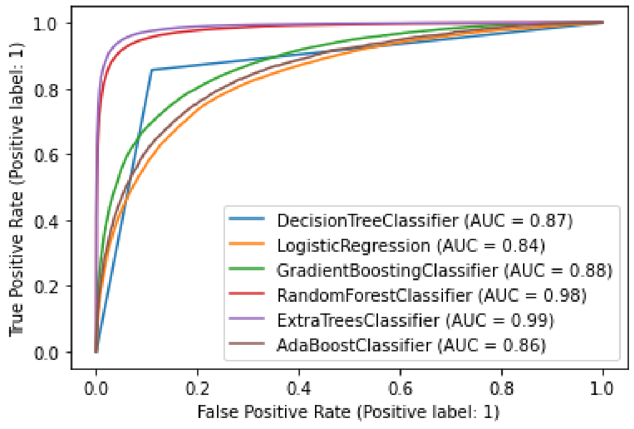
First plot the AUC curve of training dataset.

```
from sklearn.metrics import plot_roc_curve

disp = plot_roc_curve(clf, x_train, y_train)

plot_roc_curve(log, x_train, y_train, ax=disp.ax_)
plot_roc_curve(gbdt, x_train, y_train, ax=disp.ax_)
plot_roc_curve(rf, x_train, y_train, ax=disp.ax_)
plot_roc_curve(etc, x_train, y_train, ax=disp.ax_)
plot_roc_curve(ada, x_train, y_train, ax=disp.ax_)

plt.show()
```



Then plot the AUC curve of testing dataset.

```
from sklearn.metrics import plot_roc_curve

disp = plot_roc_curve(clf, x_test, y_test)

plot_roc_curve(log, x_test, y_test, ax=disp.ax_)
plot_roc_curve(gbdt, x_test, y_test, ax=disp.ax_)
plot_roc_curve(rf,x_test, y_test, ax=disp.ax_)
plot_roc_curve(etc, x_test, y_test, ax=disp.ax_)
plot_roc_curve(ada, x_test, y_test, ax=disp.ax_)

plt.show()
```



Here also ExtraTreesClassifier gives best AUC score for training and testing dataset. **So it is the final model for this dataset**.

Hyper Parameter Tuning:

Let’s do hyper parameter tuning to choose the best parameter and enhance accuracy of model. Taking CV value as 3 for hyper parameter.

```
from sklearn.model_selection import GridSearchCV
params = { 'criterion' : ["gini", "entropy","log_loss"] }

etc_grd = GridSearchCV(etc, param_grid = params, cv=3)

etc_grd.fit(x_train, y_train)
print('best params : ', etc_grd.best_params_)
```

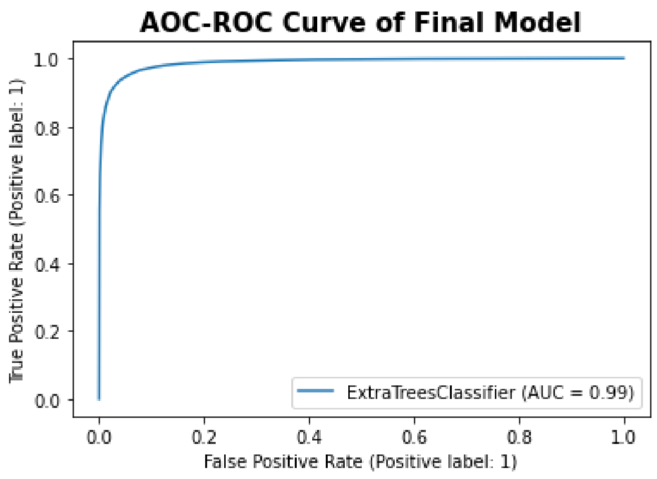
```
best params :  {'criterion': 'gini'}
```

```
accu score :  0.9475274123439139

cof_mat:  [[41959  1653]
 [ 2898 40221]]

classification report:
               precision    recall  f1-score   support

           0       0.94      0.96      0.95     43612
           1       0.96      0.93      0.95     43119

    accuracy                           0.95     86731
   macro avg       0.95      0.95      0.95     86731
weighted avg       0.95      0.95      0.95     86731

-----------
-----------
training score :  0.9999807833476178
testing score :  0.9475274123439139
```

Here accuracy score is slightly improved after using hyper parameter tuning. First, accuracy was 0.9472852843850527, but after applying hyper parameter tuning it is 0.9475274123439139.

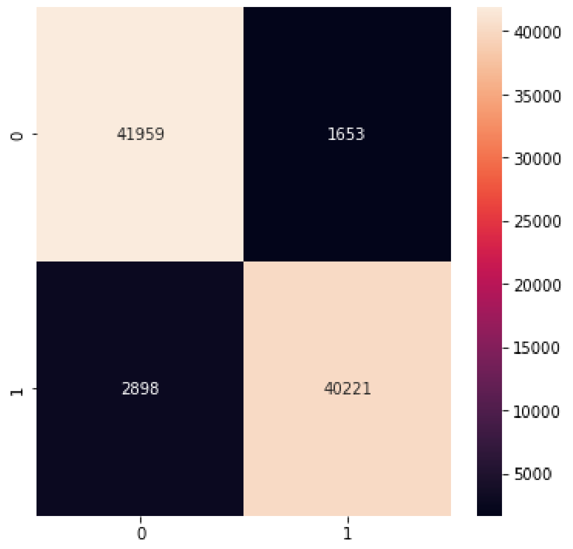
Final Model:

```python
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import plot_roc_curve
disp = plot_roc_curve(grid_etc_best,x_test,y_test)
plt.title('AOC-ROC Curve of Final Model',fontsize=15,fontweight='bold')
plt.show()
auc_score = roc_auc_score(y_test, grid_etc_best.predict(x_test))
print('Final AUC Score is: \n\n',auc_score)
```



Above graph is the final AUC-RUC graph. Let’s plot the **confusion matrix**.

```
conf = confusion_matrix (y_test, y_pred)
fig , ax = plt.subplots(figsize=(6,6))
sns.heatmap(conf, annot = True, fmt = ".0f")
plt.show()
```



Load the model:

Let’s save the model using pickle for future use. Then see the actual and predicted value of 6 random sample.

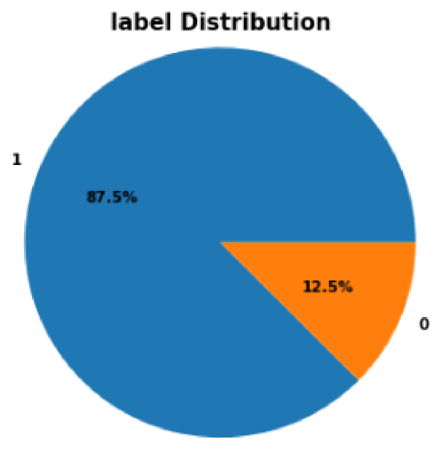
```
import pickle
pickle.dump(grid_etc_best, open("Micro_Credit_Defaulter_Classification_model", "wb"))
load_Micro_Credit_Defaulter_Classification_model= pickle.load(open("Micro_Credit_Defaulter_Classification_model", "rb"))
```

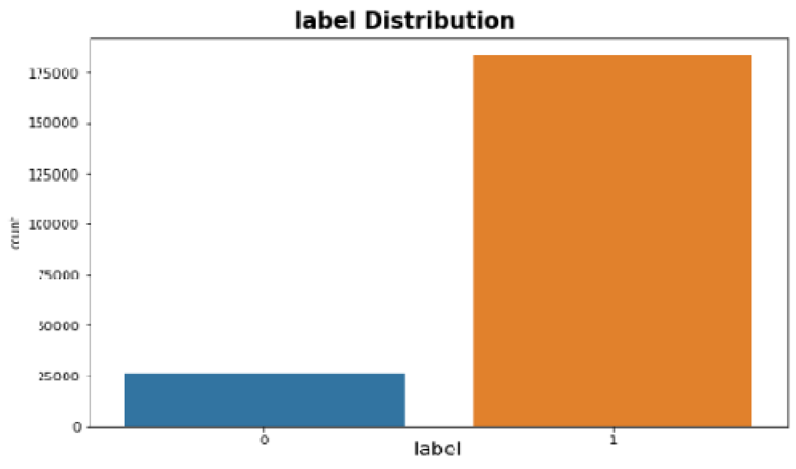
```
y_pred = load_Micro_Credit_Defaulter_Classification_model.predict(x_test)

y_test = np.array(y_test)
data_prediction_by_model = pd.DataFrame()
data_prediction_by_model["Predicted Values"] = y_pred
data_prediction_by_model["Actual Values"] = y_test
data_prediction_by_model.sample(n=6)
```

	Predicted Values	Actual Values
35196	0	0
21377	0	0
44845	1	1
32795	0	0
400	0	0
79787	0	0

Visualizations:

Let’s start the observation exploration of feature analysis.

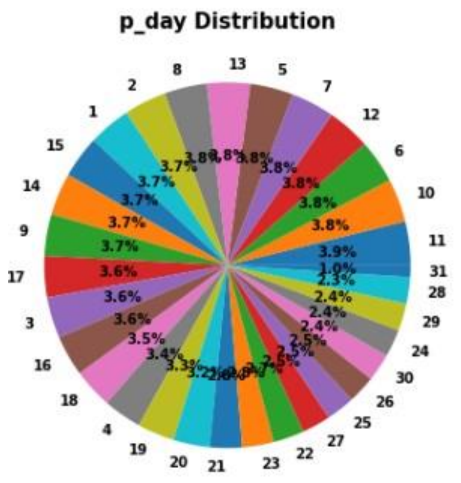


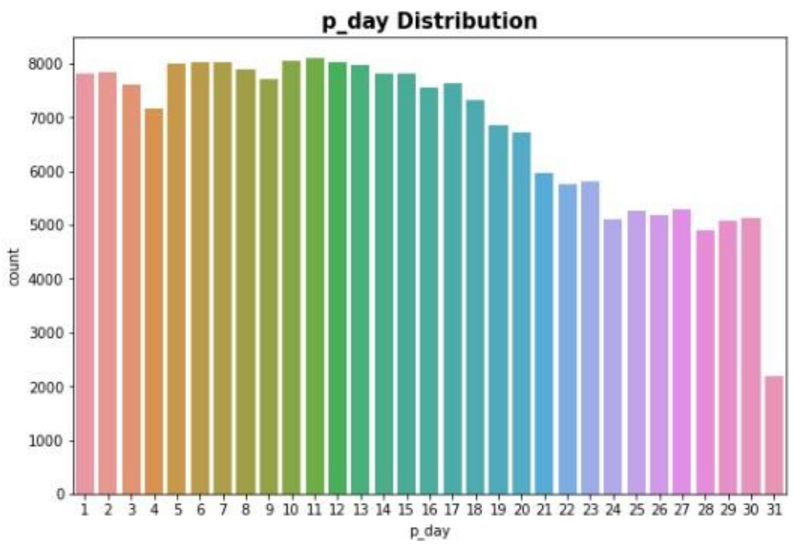


Observations:

Here 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.

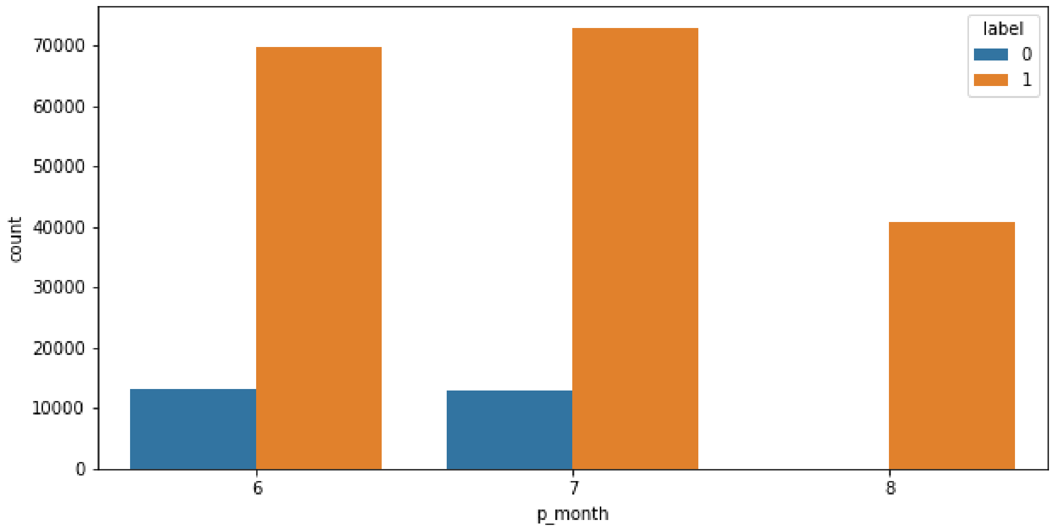
1. Maximum customers around 87.5% are Non-defaulter (label = 1).
2. Here the dataset is imbalanced.





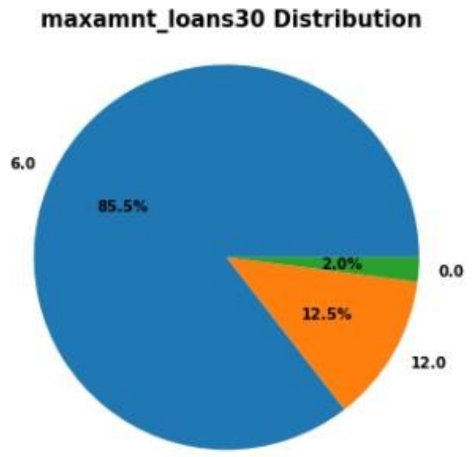
Observations:

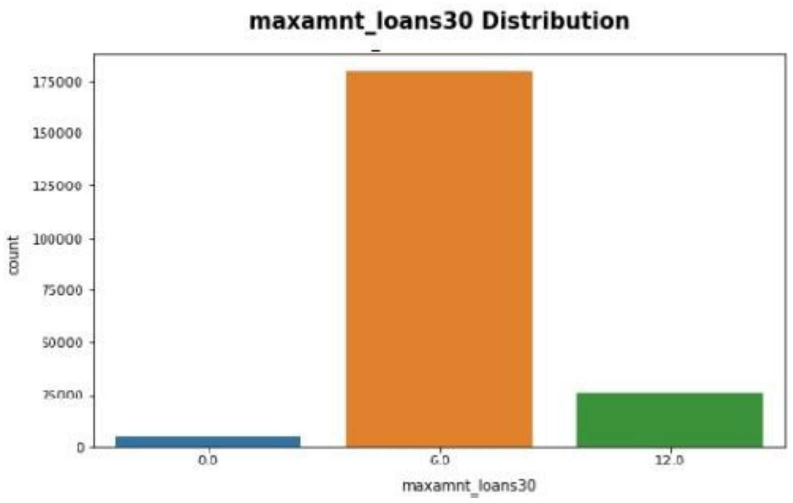
1. No significant observation is found from above plot.



Observations:

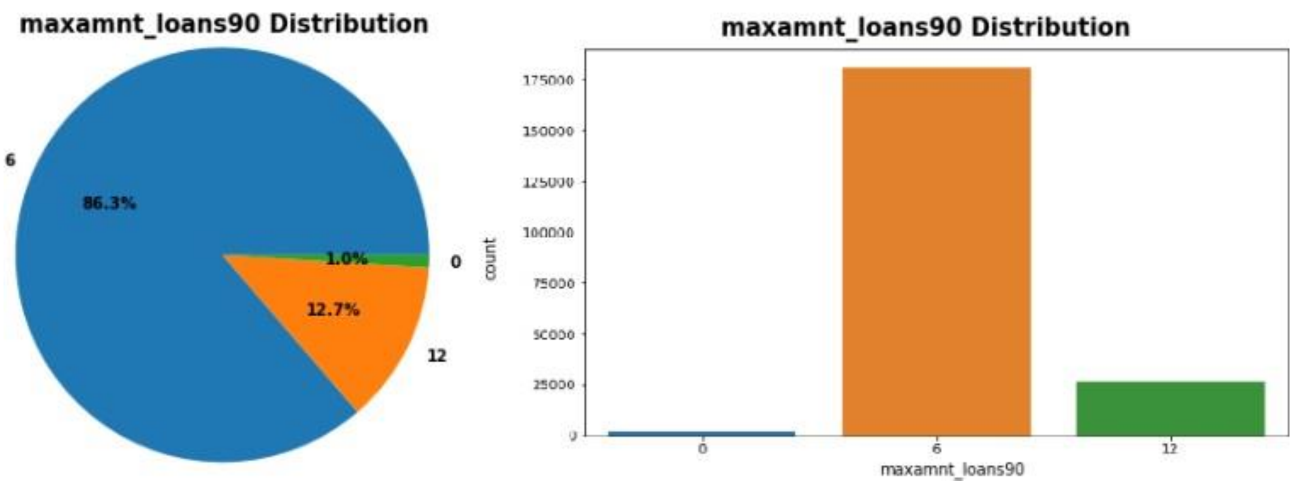
1. Maximum defaulter are from month 6 and 7.
2. There are no defaulter for month 8.





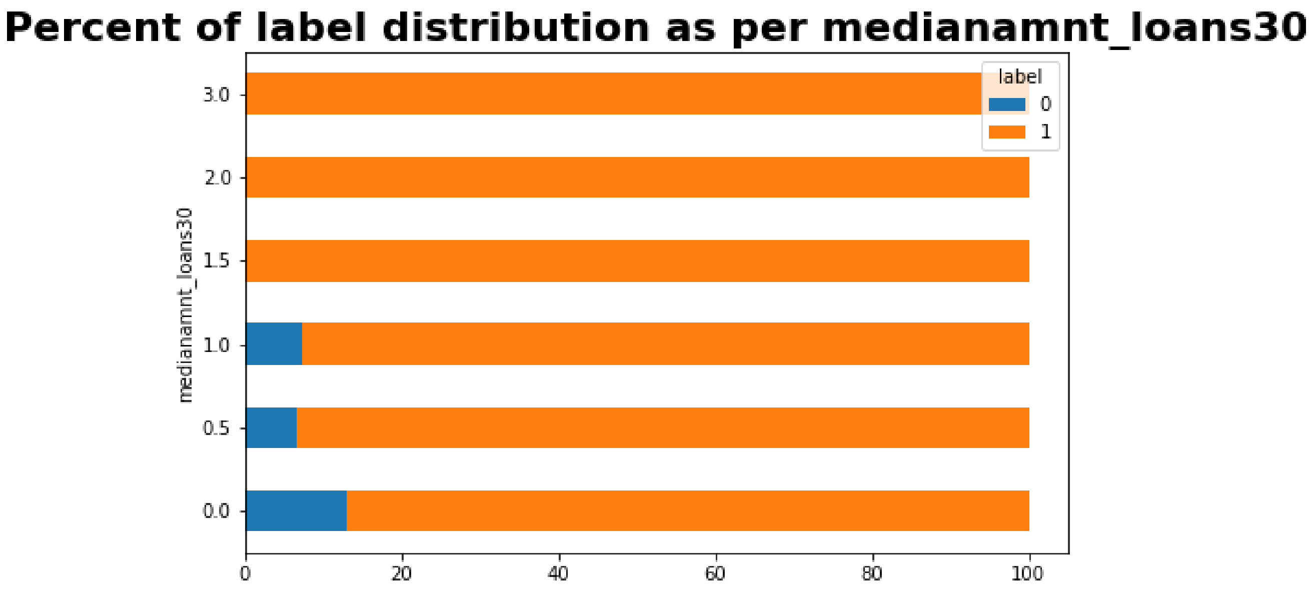
Observations:

1. Maximum number of people had taken maxamnt_loans30 equal to 6 as the loan amount and the percentage is 85.5%.
2. The number of people had not taken loan in last 30 days is 4291 which is 2.0%.
3. Obviously maximum defaulter are from maxamnt_loans30 = 6 group.
.



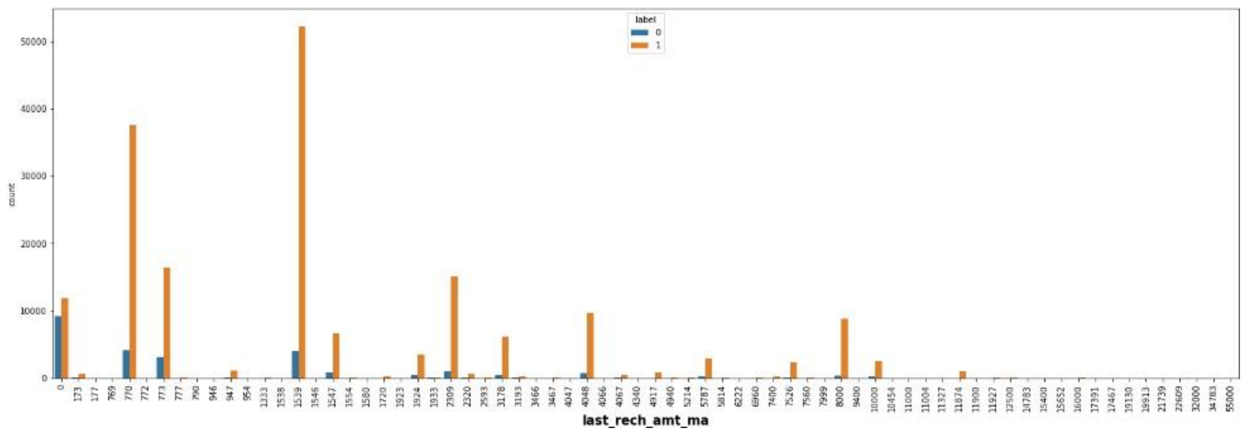
Observations:

1. Maximum number of people had taken maxamnt_loans90 equal to 6 as the loan amount and the percentage is 86.3%.
2. The number of people had not taken loan in last 90 days is 2043 which is 1.0%.
3. Obviously maximum defaulter are from maxamnt_loans90 = 6 group.
4. For maxamnt_loans30= 0, there are defaulter. But for maxamnt_loans90= 0, there are no defaulter.



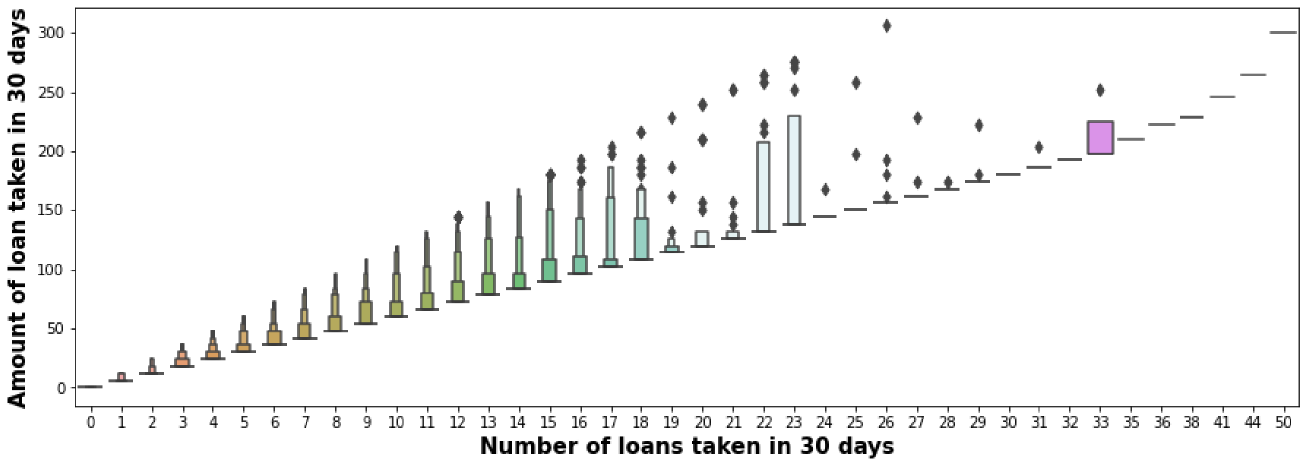
Observations:

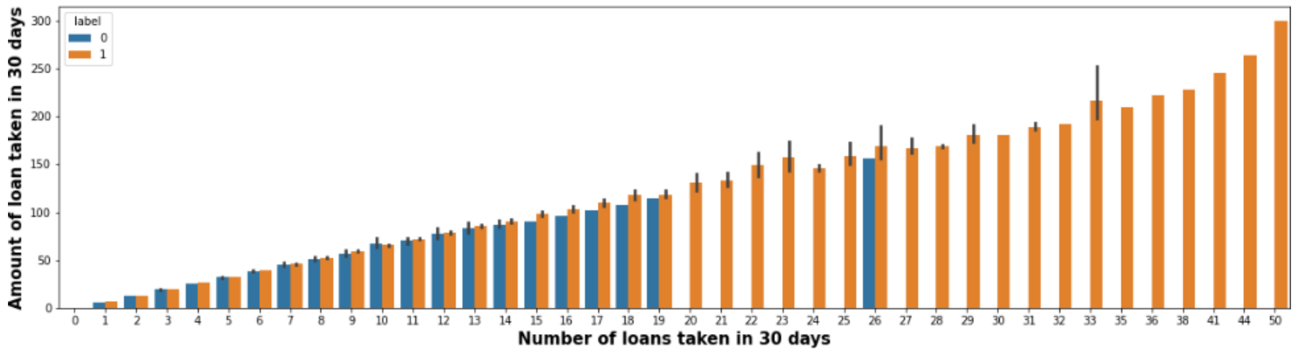
1. Maximum medianamnt_loans30 that is Median of amounts of loan taken by the user in last 30 days is 0.0.
2. No defaulter for medianamnt_loans30 equal to 1.5, 2.0, 3.0
3. Maximum medianamnt_loans90 that is Median of amounts of loan taken by the user in last 90 days is 0.0.
4. No defaulter for medianamnt_loans90 equal to 1.5, 2.0, 3.0



Observations:

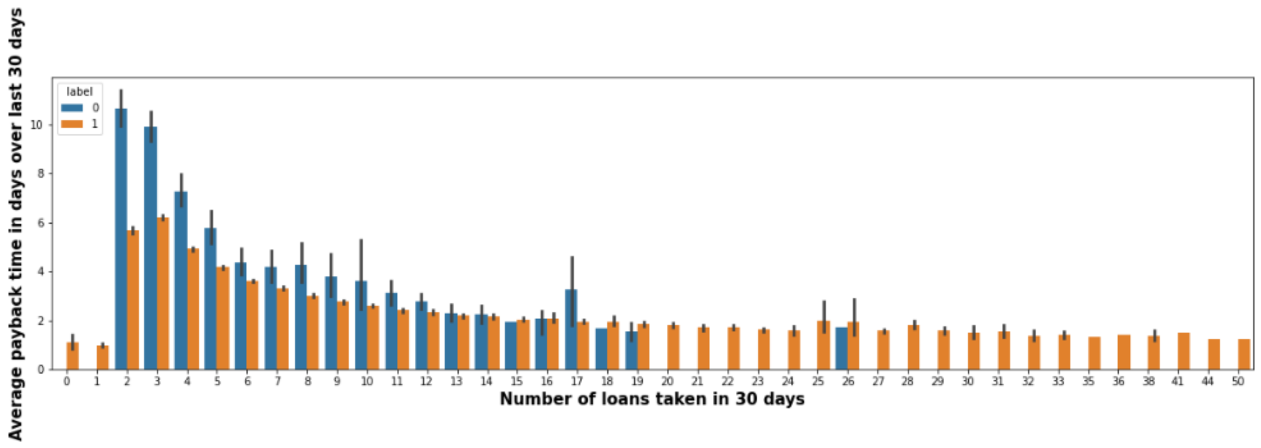
1. Amount of last recharge of main account is maximum for 770, 1539.
2. Max defaulter are from last_rech_amt_ma=0.





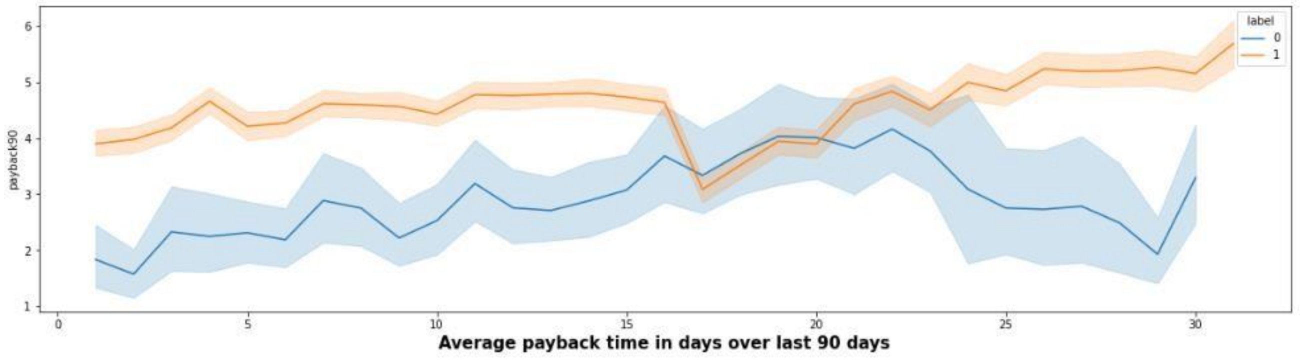
Observations:

1. Maximum number of loans taken by the people is 50 and the loan amount is equivalent to 300.
2. Minimum number of loans taken by the people is 0.
3. No defaulter are there if Number of loans taken in 30 days is in 30 to 50.



Observations:

1. The Average payback time over last 30 days is higher for people who had taken 2 and 3 times the loan in 30 days.
2. The users with less number of loan taking are more the defaulters.
3. If Number of loans taken in 30 days is increased, the defaulters are decreased.



Observations:

Average payback time in days over last 90 days is higher for Non- defaulter users.

Interpretation of the Results

After all the pre-processing steps, the training dataset is ready to train machine learning models. All unnecessary features are deleted as they might give overfitting problem as well as it also could increase the time complexity. Now apply this training dataset on different ML Classification Model (as discussed on part 3.4 - ‘Run and Evaluate selected models’) and check the best model for this particular dataset.

4. CONCLUSION

Key Findings and Conclusions of the Study

Tables of Findings using different algorithms after Hyper Parameter Tuning

Algorithm		Accuracy Score	Recall	Precision	F1 Score	Mean CV Score	Std Deviation
Logistic Regression		0.7670	0.77	0.77	0.77	0.7628	0.0017
DecisionTree Classifier		0.8728	0.87	0.87	0.87	0.8755	0.0004
GradientBoostingClassifier		0.800	0.80	0.80	0.80	0.7985	0.0033
RandomForest Classifier		0.9331	0.93	0.93	0.93	0.9355	0.0015
ExtraTrees Classifier		0.9472	0.95	0.95	0.95	0.9497	0.0007
Ada Boost Classifier		0.7779	0.78	0.78	0.78	0.7738	0.0029

Here Extra Trees Classifier giving maximum Accuracy Score, minimum RMSE Value, Maximum CV Score and minimum Standard Deviation. So Extra Trees Classifier is selected as best model.

- Accuracy score : 0.9475274123439139
- Final confusion matrix:

```
[[41959   1653]
 [ 2898  40221]]
```

- Final AUC Score is : 0.9474441184768293

Learning Outcomes of the Study in respect of Data Science

- Null removal is an important part of any problem.
- Scaling and standardization of data is mandatory.
- Feature selection played an important role in any ML problem. Unnecessary features and thefeatures correlated with another needs to be removed.
- Accuracy score can be improved after applying hyper parameter tuning.
- Dataset must be balanced. If the dataset is imbalanced, it may affect the final prediction. Sooversampling is an important technique to overcome this problem.
- Data needs to be much precise and detailed for much better score.
- First time I handle such a huge dataset. It is very time taken to run individual model.
- When I hyper tuned the final model, it take more than 10 hours. But still no outcomes arecoming. Then I reduce the parameter and finally it take around 7-8 hours to give the best result.

Limitations of this work and Scope for Future Work

- Primarily, the dataset is a huge dataset. But as the target variable is imbalanced, here we needto apply oversampling method. But then the dataset is bigger than earlier. If I use under sampling, the running time may be less as compared to now.

Accuracy score can increase with hyper parameter tuning with several different parameter.As it takes a lot of time, I am not able to use lot of parameters here.