MICRO CREDIT DEFAULTER MODEL

Submitted by:

MEGHA SINGH

ACKNOWLEDGEMENT

The internship opportunity I have with Flip Robo Technologies is a great chance for learning and professional development. I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills acknowledge in the best possible way.

I would like to extend my appreciation and thanks for the mentors from DataTrained and professionals from FlipRoboTechnologies who had extended their help and support.

References: www.scipy.org, Kaggle, Github

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.

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.

The sample data is provided to us from our client database. It is hereby given to you for this exercise. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

- ## Conceptual Background of the Domain Problem

The Company provide the previous loan data of customer. So we can build a **Machine Learning Model** which can help the company to determine who is defaulter customer. We use **Exploratory Data Analysis** and **Visualization** to determine the defaulter customer and using python programming we build a predictive model.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Let's import our csv data file into by importing some important library and loading into our Jupyter Notebook

Importing Important Library

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

```
#Loading the Dataset
pd.set_option('display.max_columns',None)
df=pd.read_csv('Data file.csv',parse_dates=['pdate'])
df.head()
```

PANDAS

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself. Wes McKinney started building what would become pandas at AQR Capital while he was a researcher there from 2007 to 2010.

Library features

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation[6] and frequency conversions, moving window statistics, moving window linear regressions, date shifting and lagging.
- Provides data filtration.

Use in this project:

Most of the work in this project was done with the help of the Pandas library.

Right from the first line of code after importing the required libraries was reading the dataset by reading it in the form of a dataframe(two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns)

df=pd.read_csv('micro_credit_defaulter.csv',sep='\t')

After the dataset was present in a tabular format,methods such as 'df.dtypes-used to check the datatypes of the features) and 'df.isnull().sum()-used to check for any missing values present in the dataset'.

Pandas methods used in this project are:

df.drop()
df.dtypes
pd.read_csv()
df.isnull().sum()
pd.to_datetime()
df.head()
df['feature'].unique()
df['feature'].nunique()
df.columns
df.skew()
df.corr()
df.shape
df.groupby()
df.describe()
df.reset_index()
df.value_counts()
pd.DataFrame()

Matplotlib library:

The methods used from this library played an important role in visualizing the data.The methods/plots used from this library were:

plt.boxplot()
plt.show()
plot.bar()
plt.xlabel([])
plt.ylabel('')
plt.title([])
plt.figure(figsize=(x,y))

Seaborn(sns)

The methods from this library were very effective in the univariate aswell as multivariate analysis.Important insights were received by plotting different features against the target feature.The methods/plots used in this analysis were:

sns.distplot()
sns.countplot()
sns.scatterplot()
sns.set_style()

Numpy(np):
The method np.where() from numpy helped in filtering only the data from the array which satisfied a particular condition.
np.abs() method was used to convert the negative values to positive while calculating the zscore of the dataset values.

Scipy
The function zscore was imported from the scipy.stats library in order to calculate the zscore value for each of the dataset values while removing the outliers(though it did not contribute much in the end as the outliers were not removed).

- The dataset features along with their definitions

Variable	Definition
label	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan{1:success, 0:failure}
msisdn	mobile number of user
aon	age on cellular network in days
daily_decr30	Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah) Daily amount spent from main account, averaged over last 30 days
daily_decr90	Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)
rental30	Average main account balance over last 30 days
rental90	Average main account balance over last 90 days
last_rech_date_ma	Number of days till last recharge of main account
last_rech_date_da	Number of days till last recharge of data account
last_rech_amt_ma	Amount of last recharge of main account (in Indonesian Rupiah)
cnt_ma_rech30	Number of times main account got recharged in last 30 days
fr_ma_rech30	Frequency of main account recharged in last 30 days
sumamnt_ma_rech30	Total amount of recharge in main account over last 30 days (in

	Indonesian Rupiah)
medianamnt_ma_rech30	Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
medianmarechprebal30	Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
cnt_ma_rech90	Number of times main account got recharged in last 90 days
fr_ma_rech90	Frequency of main account recharged in last 90 days
sumamnt_ma_rech90	Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)
medianamnt_ma_rech90	Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)
medianmarechprebal90	Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)
cnt_da_rech30	Number of times data account got recharged in last 30 days
fr_da_rech30 cnt_da_rech90	Frequency of data account recharged in last 30 days Number of times data account got recharged in last 90 days
fr_da_rech90	Frequency of data account recharged in last 90 days
cnt_loans30	Number of loans taken by user in last 30 days
amnt_loans30	Total amount of loans taken by user in last 30 days
maxamnt_loans30	maximum amount of loan taken by the user in last 30 days
medianamnt_loans30	Median of amounts of loan taken by the user in last 30 days
cnt_loans90	Number of loans taken by user in last 90 days
amnt_loans90	Total amount of loans taken by user in last 90 days
maxamnt_loans90	maximum amount of loan taken by the user in last 90 days
medianamnt_loans90	Median of amounts of loan taken by the user in last 90 days

payback30	Average payback time in days over last 30 days
payback90	Average payback time in days over last 90 days
pcircle	telecom circle
pdate	date

- Data Pre-processing Done

Data Cleaning

The biggest challenge faced in the working of this course was the cleaning of the data which was full of outliers and unrealistic values.

Most outliers however,were not removed as it would have lead to the loss of a large chunk of data(23% approximately) against which I was instructed along with the project use case.

Also,a lot of the features providing similar information were highly correlated with each other and could have led to poor model performance due to multicollinearity.

Apart from that,there were a few features that were providing the same information like :

1)The features -fr_ma_rech30 is providing the same information as 'cnt_ma_rech30'-Number of times main account got recharged in last 30 days is same as the 'fr_ma_rech30'-Frequency of main account recharged in last 30 days.

2) The features-fr_ma_rech90 Frequency of main account recharged in last 90 days and cnt_ma_rech90 Number of times main account got recharged in last 90 days are providing the same information as the frequency is same as the number of times the main account was recharged over 90 days.

3) The features -cnt_da_rech30(Number of times data account got recharged in last 30 days) and fr_da_rech30(Frequency of data account recharged in last 30 days) provide the same information

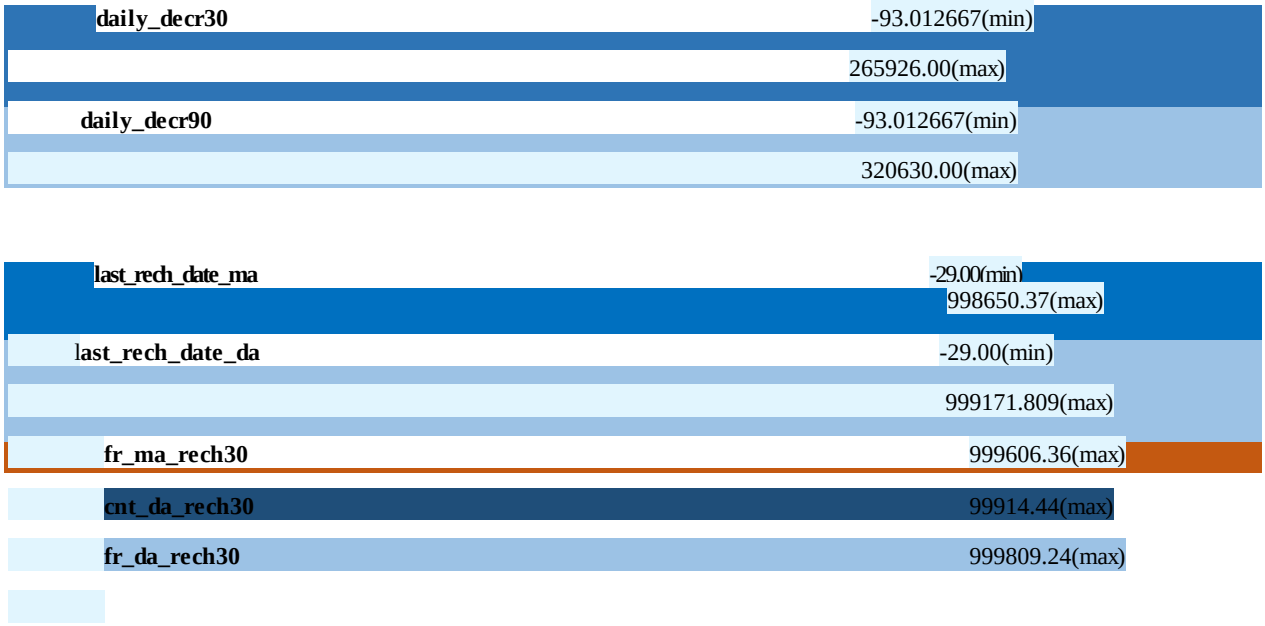
4) The features-cnt_da_rech90(Number of times data account got recharged in last 90 days) provide the same information as the fr_da_rech90(Frequency of data account recharged in last 90 days).

There were a lot of features depicting unreal maximum and minimum values:

The features -daily_decr30,daily_decr90 cannot be negative as the average amount spent from main account cannot be less than 0. 2)The feature-last_rech_date_ma,last_rech_date_da cannot be negative as the number of days since last recharge can never be negative.Also,the number of days since last recharge cannot be greater than (15,000days till 23/11/2021) as the first telecommunications network was established in japan in 1979,hence any value above it will be unrealistic.

2) The feature-daily_decr30(Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)) has unreal values present as the minimum value for the daily average expenditure cannot be less than zero.Also,the maximum value(265926.00) too seems quite unusual.

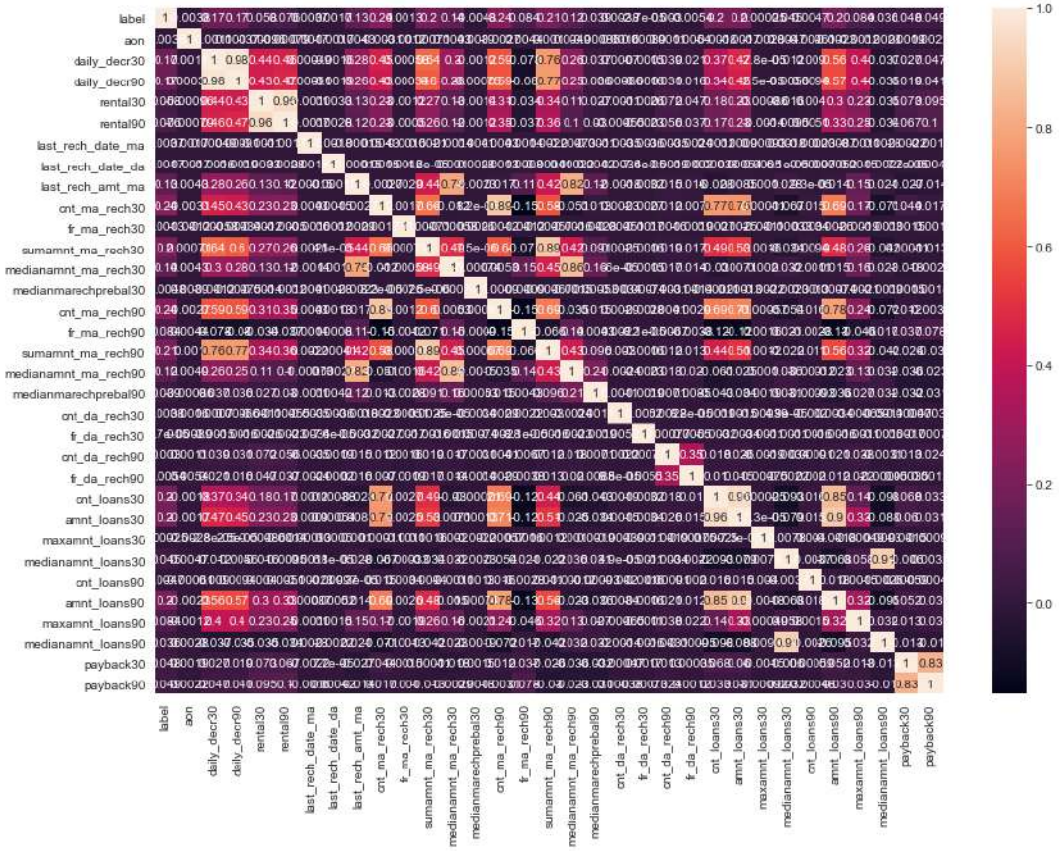
3) Features **Unreal Values**



Apart from these features,some more 'apparently unreal' values were visible in a few more features but were not confirmed due to the lack of the domain knowledge.

The features in the above table were cleaned off their unreal min/max data by dropping the rows containing them and at the same time keeping in mind that the data loss does not reach a significant level.

Correlation Between Target and Other Columns:



OBSERVATION:

Their appears to be a very high collinearity amongst the independent variables. The features that have been observed for 30 and 90 days are all highly correlated as the 90 day observation period appears to cover the information provided by the ones that are providing the same information for a 30 day period.hence,dropping the columns which provide data for 30 days as the information is already being provided by the feature with 90data of similar information.

The heatmap after the aforementioned features were dropped:

Data Preprocessing Done:

We learn about the data by columns. There are some columns available which contains some personal information about customers like Mobile no, Network Circle, Loan date these columns have no requirement in model building. So I dropped those columns.

After checking the dataset we also found that there are no null values. If null values are present we replace them with mean, median, mode according to Columns or attributes.

We also Check for Outliers in Dataset and found that there are outliers present in every column and we try to remove them with zscore, where threshold=3.Bt the outlier data contain more than 20% of whole data. If we remove those data we may lose the data about some of defaulters. So we could not remove outlier.

Balancing the dataset:

The dataset was largely imbalanced as Label '1' had approximately 87.5% records, while, label '0' had approximately 12.5% records.

y.value_counts()

1 176316
0 23500
Name: label, dtype: int64
Therefore,the Smote class from the imblearn.over_sampling

Library had to be used to upsample the '0' classification data.

Final data after upsampling was:

trainy.value_counts()

1 176316
0 176316
Name: label, dtype: int64

Feature Scaling(Standardization)

As the variation in the maximum and minimum value(range) within and among the features was very high,Feature scaling had to be performed in order to make the data suitable for distance based algorithmns such as KNeighbors Classifier and even Logistic Regression performed better with the scaled data.Standardization technique using StandardScaler was used for the same which had to be imported from the sklearn library.

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

Hardware:

Processor—Intel (R) Core(TM) i5-4210U CPU @1.70GHZ @2.40GHz

Installed Memory (RAM)—8.00 GB

System type—64-bit Operating System
Software: Windows 10

We have used Python Package because it is powerful and general purpose programming language.

NumPy—It is a math library to work with N dimensional arrays. It enables us to do computation effectively and regularly. For working with arrays, dictionary, functions data type we need to know NumPy

Pandas—It is high level Python library and easy to use for data importing, manipulation and data analysis.

Matplotlib—It is a plotting that provide 2D and 3D plotting.

Seaborn-- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

SciPy—It is a collection of numerical algorithm and domain specific tool boxes including optimization, statistics and much more.

Scikit-learn—It is a collection of tools and algorithm for machine learning. It works with NumPy and SciPy and it is easy to implement machine learning models.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

We know that It is a **Classification Problem** so we use accuracy score, classification report, confusion matrix for evaluation matrix. Then we use precision, recall, F1 score, auc roc score, auc roc score, cross validation score for finalizing the model.

- Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing

Logistic Regression

KNeighbors Classifier

GausianNB

Random Forest Classifier

Ada Boost Classifier

LOGISTIC REGRESSION:

Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical.

RANDOM FOREST CLASSIFIER

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

ADABOOST CLASSIFIER

Ada-boost or Adaptive Boosting is one of ensemble boosting classifier proposed by Yoav Freund and Robert Schapire in 1996. It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures

the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set.

Gradient Boosting Classifier

Gradient Boosted Decision Trees (GBDT) is a generalization of boosting to arbitrary differentiable loss functions. GBDT is an accurate and effective off-the-shelf procedure that can be used for both regression and classification problems in a variety of areas including Web search ranking and ecology.

GradientBoostingClassifier supports both binary and multi-class classification.

MACHINE LEARNING

Below are the libraries used in building model

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import accuracy_score,roc_auc_score,roc_curve,auc,confusion_matrix,classification_report
from sklearn.model_selection import GridSearchCV,cross_val_score
from sklearn.model_selection import train_test_split
```

```
models=[]
models.append(('LogisticRegression',lr))
models.append(('GaussianNB',gnb))
models.append(('RandomForestClassifier',rfc))
models.append(('GradientBoostingClassifier',gbc))
models.append(('AdaBoostClassifier',adc))
models.append(('KNeighborsClassifier',knn))
```

Train The Model for Best Accuracy

```
Model=[]
score=[]
CVS=[]
rocscore=[]
for name,model in models:
    print(name)
    print('\n')
    Model.append(name)
    model.fit(x_train,y_train)
    print(model)
    pre=model.predict(x_test)
    print('\n')
    AS=accuracy_score(y_test,pre)
    print('Accuracy_score=',AS)
    score.append(AS*100)
    print('\n')
    sc=cross_val_score(model,x,y,cv=10,scoring='accuracy').mean()
    print('Cross_Val_Score=',sc)
    CVS.append(sc*100)
    print('\n')
    false_positive_rate,true_positive_rate,threshold=roc_curve(y_test,pre)
    roc_auc= auc(false_positive_rate,true_positive_rate)
    print('roc_auc_score=',roc_auc)
    rocscore.append(roc_auc*100)
    print('\n')
    print('classification_report\n',classification_report(y_test,pre))
    print('\n')
    cm=confusion_matrix(y_test,pre)
    print(cm)
    print('\n')
    plt.figure(figsize=(10,40))
    plt.subplot(911)
    plt.title(name)
    plt.plot(false_positive_rate,true_positive_rate,label='AUC = %0.2f'% roc_auc)
    plt.plot([0,1],[0,1],'r--')
    plt.legend(loc='lower right')
    plt.ylabel('True positive Rate')
    plt.xlabel('False Positive Rate')
    print('\n\n')
```

Logistic Regression

accuracy score: 0.7600305923398843
[[47090 11058]
 [16867 41354]]

Confusion matrix:

	precision	recall	f1-score	support
0	0.74	0.81	0.77	58148
1	0.79	0.71	0.75	58221
accuracy			0.76	116369

macro avg	0.76	0.76	0.76	116369
weighted avg	0.76	0.76	0.76	116369

Cross_val_score

mean score: 0.9178644310895997
cross val score: [0.91870183 0.91684808 0.91887496 0.91787403 0.91702325]

accuracy_score: 0.9460938909847124

Confusion matrix
[[54642 3506]
[2767 55454]]

Classification report:

	precision	recall	f1-score	support
0	0.95	0.94	0.95	58148
1	0.94	0.95	0.95	58221
accuracy			0.95	116369
macro avg	0.95	0.95	0.95	116369
weighted avg	0.95	0.95	0.95	116369

Cross_val_score

mean score: 0.921147456127175
cross val score: [0.92105395 0.92030128 0.92070165 0.92222806 0.92145234]

roc_auc_score

Random Forest test roc-auc: 0.9871270738446035

In [208]:

Adaboost Classifier

accuracy_score: 0.8748893605685363

confusion matrix

[[50811 7337]
 [7222 50999]]

Classification report:

	precision	recall	f1-score	support
0	0.88	0.87	0.87	58148
1	0.87	0.88	0.88	58221
accuracy			0.87	116369
macro avg	0.87	0.87	0.87	116369
weighted avg	0.87	0.87	0.87	116369

cross_val_score

mean score: 0.9104876517334434
cross val score: [0.90986888 0.91006681 0.91171834 0.91061732 0.9101669]

roc_auc_score

Adaboost test roc-auc: 0.9469172369281446

Gradient Boosting Classifier

accuracy_score: 0.8983664034235922

confusion matrix
[[52170 5978]
 [5849 52372]]
Classification report:

	precision	recall	f1-score	support
0	0.90	0.90	0.90	58148
1	0.90	0.90	0.90	58221
accuracy			0.90	116369
macro avg	0.90	0.90	0.90	116369
weighted avg	0.90	0.90	0.90	116369

cross_val_score

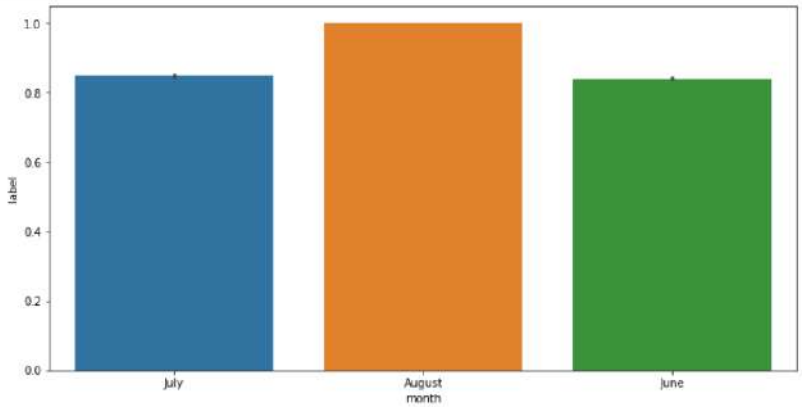
mean score: 0.9178644310895997
cross val score: [0.91870183 0.91684808 0.91887496 0.91787403 0.91702325]

roc_auc_score

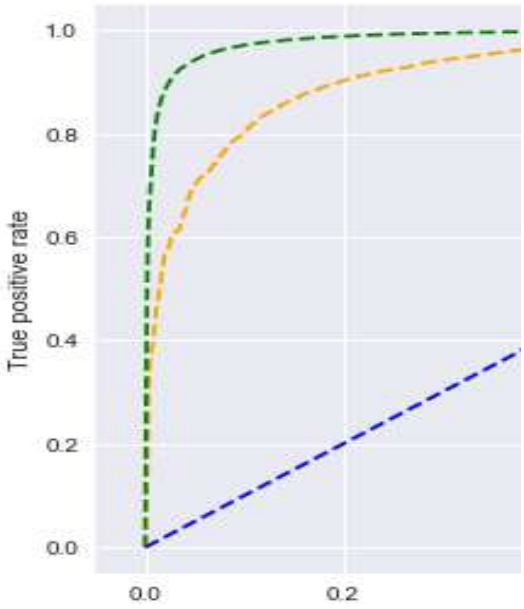
GradientBoostingClassifier test roc-auc: 0.9628719795709024

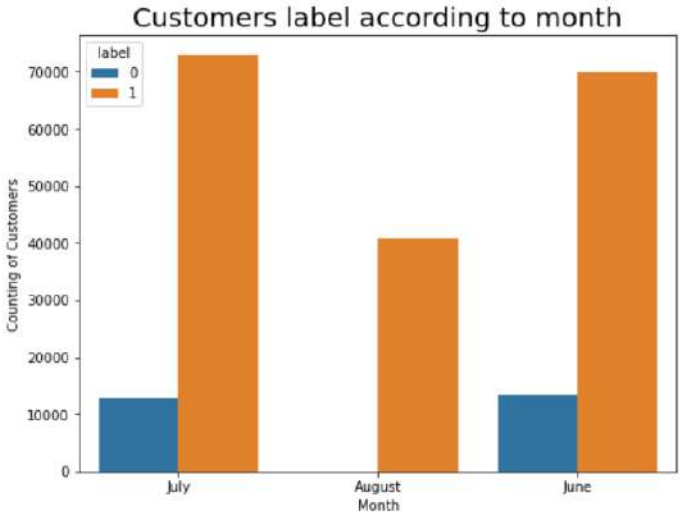
- **Visualization**
  The visualization metric used for testing how well the model can identify the probable defaulters.The visualizations were:

```
# average label during month of an year
plt.figure(figsize=(12,6))
sns.barplot(x='month',y='label',data=df_dates)
plt.show()
# it shows average label is increaseing from jun to august
```



RandomForestClassifier vs AdaboostClassifier





Observation:

Only Jun,July,August of 2016 data available.

Number of Non-Defaulter are greater than number of Defaulter.

In Month August there are no defaulters

CONCLUSION

- ## Key Findings and Conclusions of the Study

  There is no Null value.

  The dataset is imbalanced. Label '1' has approximately 86% records, while, label '0' has approximately 14% records.

  maxamnt_loans90 columns gives information about customers with no loan history.

  msisdn feature some values which might not be realistic. So drop the row which contain not realistic value.

  There are some rows which is repeated means duplicate entries are present in our dataset.

  The collected data is only for one area circle(UPW).

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

  Here I learned about the micro credit industry, visualization, data cleaning, handling outliers and using various algorithms on huge dataset. This was the first time I worked on such huge dataset. It took a lot of time to train all the algorithms to find out the best one to work with. Working with such huge dataset that took a lot of time to train the algorithms and tuning it for the best prams was worth knowing in this project.
  And I also know that the best way to finalize the model is trying every possibility with patience

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

  As the data set content more than 20% of Outliers we cannot remove than so that is a big limitation. If outliers were removed, we can build more efficient model.

  If there were only one-month data available, we can tune the data more efficiently.

Here I learned about the micro credit industry, visualization, data cleaning, handling outliers and using various algorithms on huge dataset. This was the first time I worked on such huge dataset. It took a lot of time to hyper tune all the algorithms to find out the best one to work with. Working with such huge dataset that took a lot of time to train the algorithms and tuning it for the best parameter was worth knowing in this project.