Micro-Credit Defaulter Model

Submitted by:

Aaron D’souza

ACKNOWLEDGMENT

Resources:

- **Link:** https://www.youtube.com/user/krishnaik06
- **Link:** https://www.udemy.com/course/data-science-for-business-6-real-world-case-studies/learn/lecture/19343060?start=900#overview
- **Link:** https://www.udemy.com/course/python-for-data-science-and-machine-learning-bootcamp/learn/lecture/5733454?start=15#overview
- **Link:** https://www.youtube.com/watch?v=6OPf1lYmJhg
- **Link:** https://github.com/krishnaik06/Gaussian-Trnasformaion/blob/master/Untitled1.ipynb
- **Link:** https://github.com/krishnaik06/Feature-Engineering-Live-sessions/blob/master/Outliers.ipynb

INTRODUCTION

- ## Business Problem Framing

A Microfinance institution is an organization that offers financial services to low-income populations.

Almost all Microfinance institutions give loans to their members, and many offers insurance, deposit and other services. 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.

It is very important for a Microfinance Institution (MFI) to know whether an individual will be able to repay the loan provided to them within a certain timeframe or not.

- ## Conceptual Background of the Domain Problem

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 allows people to take on reasonable small business loans safely, and in a manner that is consistent with ethical lending practices.

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.

- ## Review of Literature

Link: https://www.investopedia.com/terms/m/microfinance.asp

The main motivation behind Microfinance Institution (MFI) is to provide financial services to low-income populations. Poverty is the main cause of concern in improving the economic status of developing countries.

The experts from Microfinance Institutions (MFI) are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient in terms of costs as compared to traditional high touch model.

MFIs are the pivotal overseas organizations in each country that make individual microcredit loans directly to villagers, microentrepreneurs, impoverished women and poor families.

- ## Motivation for the Problem Undertaken

The main motivation behind the problem is to improve the quality of life of the poor and the less fortunate Individuals by providing them access to financial and support services to improve their lifestyle.

The goal is to create opportunities for self-employment for the underprivileged.

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

The goal is 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 terms of model building the target variable (label) is categorical in nature so we need to solve this as a classification problem.

If the 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.

A total of 7 classification models were used in order to classify the target variable Label.

Logistic Regression.

Naive Bayes.

Random Forest Classification.

Decision Tree Classification.

ADA Boost Classifier.

K-NN Classifier.

Gradient Boosting Classifier.

- ## Data Sources and their formats

The sample data is provided to us from our client database.

The goal is 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.

A total of three datatypes in the dataset Float, Int and Object.

The data provided was well organized structured data in .CSV (comma-separated values) format.

The data set has a total of 37 columns:

- **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 users.**
- **Aon: age on cellular network in days.**
- **Daily-decr30: Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah).**
- **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 number 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 Indonesian Rupiah).**
- **Medianamnt_ma_rech90: Median of number of recharges done in main account over last 90 days at user level (in Indonesian Rupiah).**
- **Medianmarechprebal90: Median of main account balance just before recharge in last 90 days at user level (in Indonesian Rupiah).**
- **Cnt_da_rech30: Number of times data account got recharged in last 30 days.**
- **Fr_da_rech30: Frequency of data account recharged in last 30 days.**

- **Cnt_da_rech90: 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.**
- **P-circle: telecom circle.**
- **P-date: date.**

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90

Fig.1 Sample data in Jupyter notebook.

- ## Data Pre-processing Done

No Null values in the data set.

There were no duplicated values in the dataset.

```
In [10]: 1 df.duplicated().sum()
Out[10]: 0
```

No duplicated values in the dataset

The columns “Unnamed”,”Pcircle” were dropped as both the columns did not have any significance for further analysis of data.

The MSISDN is Mobile Station International Subscriber Directory Number it consists of 3 sections Country code, national destination code, and subscriber number. The data type of this column must be int but due to the presence of the alphabet” I” in the mobile number it was showing object. We extracted the alphabet “I” from the column and converted the column to an Int datatype.

```
In [18]: 1 df["msisdn"] = df["msisdn"].str.split().str[0]
In [19]: 1 df['msisdn'] = df['msisdn'].apply(lambda x: x.split()[0].replace("I",""))
```

The code above will remove the alphabet "I" from the MSISDN column

```
In [20]: 1 df["msisdn"].dtype
Out[20]: dtype('O')
```

We need to convert the object datatype to Int datatype for the column "msisdn"

Using Pandas date time function, we extracted Month and day information from “P-date” column.

We need to extract "Month" and "Day" information from pdate column

```
In [25]: 1 df['pdate'] = pd.to_datetime(df['pdate'])
```

Creating two new columns "Month" and "Day" from pdate column

```
In [26]: 1 df['Month']=df['pdate'].apply(lambda a:a.month)
         2 df['Day']=df['pdate'].apply(lambda a:a.day)
```

I created two new data frames “Loan_Paid” and “Not_paid” and with the help of those data frame we conclude that a total of 88% of the

individuals paid the loan and 13 % were the individuals who failed to pay the loan back.

In [29]:
```
print("Total customers =", len(df))

print("Number of customeres who paid the loan =", len(Loan_Paid))

print("Percentage of customeres who paid the loan =", 1.*len(Loan_Paid)/len(

print("Number of customeres who did not pay the loan =", len(Not_Paid))

print("Percentage of customeres who did not Pay the loan =", 1.*len(Not_Paid
```

```
Total customers = 209593
Number of customeres who paid the loan = 183431
Percentage of customeres who paid the loan = 87.5177129007171 %
Number of customeres who did not pay the loan = 26162
Percentage of customeres who did not Pay the loan = 12.482287099282896 %
```

The data set is highly imbalanced

The AON (Age of cellular network in days cannot be negative) so I replaced all the negative values with zero and then replaced them with the median of “Aon” column itself.

In [31]:
```
df['aon'] = df['aon'].apply(lambda x : x if x > 0 else 0)
```

In [32]:
```
df["aon"].describe()
```

Out[32]:
```
count    209593.000000
mean       8112.576081
std       75696.057541
min           0.000000
25%         246.000000
50%         527.000000
75%         982.000000
max      999860.755168
Name: aon, dtype: float64
```

Replacing all the zeros in "aon" column with the median value of "aon" column

daily_decr30 is the daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah). All the negative values in the daily_decr30 column were replaced with the medians of the column itself.

Similarly, in most of the columns which had non-realistic values were imputed with medians of the column itself.

```
In [49]: df['daily_decr90'] = df['daily_decr90'].apply(lambda x : x if x > 0 else 0)

In [50]: df["daily_decr90"] = df["daily_decr90"].replace(0,df['daily_decr90'].median(

In [51]: df["daily_decr90"].describe()

Out[51]: count    209593.000000
         mean       6124.768610
         std       10898.074485
         min           0.000667
         25%          53.592000
         50%        1500.000000
         75%        7802.790000
         max      320630.000000
         Name: daily_decr90, dtype: float64
```

The column “maxamnt_loans30” must only have 3 values either 6, 12 or zero all the remaining values were removed and replaced with zero.

There were a lot of “Outliers” in the data set. Z score technique was used to remove the outliers but the overall data loss was more than 23% so we could not go-ahead with the Z score technique.

Percentage data loss

```
In [132]: loss = (209593-160475)/209593*100

In [133]: print("The data loss is ",loss," %")

          The data loss is  23.434942960881326  %
```

We cannot go ahead with z score technique as there is 23 % data loss. the data loss is very high

We created a function to remove and replace outliers with the median values of the data.

Lets create a function to replace the outliers with median values of data

```
In [136]: def replace_outliers(col):

              q = df[col].quantile(0.95)
              df[col] = df[col].apply(lambda x : x if x <= q else 0)
              df[col] = df[col].replace(0,df[col].median())

In [137]: replace_outliers("aon")
```

There was high Skewness in the data which was reduced with the help of log transformation and power transformation most of the skewness

values were adjusted to be in the range (- 0.50, + 0.50) to achieve normal distribution.

Checking Skewness of data

```
In [178]: 1 df.skew()
```

```
Out[178]: msisdn                0.018124
          aon                   0.791593
          daily_decr30          1.631028
          daily_decr90          1.733318
          rental30              1.793629
          rental90              1.845694
          last_rech_date_ma     2.003807
          last_rech_date_da    14.814857
          last_rech_amt_ma      2.282330
          cnt_ma_rech30         1.292975
          fr_ma_rech30          1.635904
```

The columns “last_rech_date_da”,” cnt_da_rech30”,” fr_da_rech30”,” cnt_da_rech90” and “fr_da_rech90” as they had almost zero correlation with the target variable.

Based on correlation results we need to drop certain columns

```
In [299]: 1 df = df.drop(["last_rech_date_da","cnt_da_rech30","fr_da_rech30","cnt_da_rec
```

```
In [300]: 1 df.skew()
```

```
Out[300]: msisdn                0.018124
          aon                  -0.535053
          daily_decr30         -0.483433
```

Sklearn’s Train Test split was used. And the initial random state was taken as zero.

Train Test Split

```
In [301]: 1 df.head()
```

Out[301]:

	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last
0	2140870789	5.609472	8.024879	8.028178	-1.069255	-1.041421	1.098612	
1	7646270374	6.569481	9.402860	9.403087	1.015349	0.808155	3.044522	
2	1794370372	6.284134	7.243513	7.243513	-0.146513	-0.272648	1.386294	
3	5577370781	5.488938	3.101353	3.101353	-1.248368	-1.299169	1.386294	
4	381382730	6.854355	5.021373	5.021373	0.003125	-0.133371	1.609438	

```
In [302]: 1 from sklearn.model_selection import train_test_split
```

The data was Scaled using standard scalar technique from Sklearn.

Scaling the data

In [312]:
```
from sklearn.preprocessing import StandardScaler
```

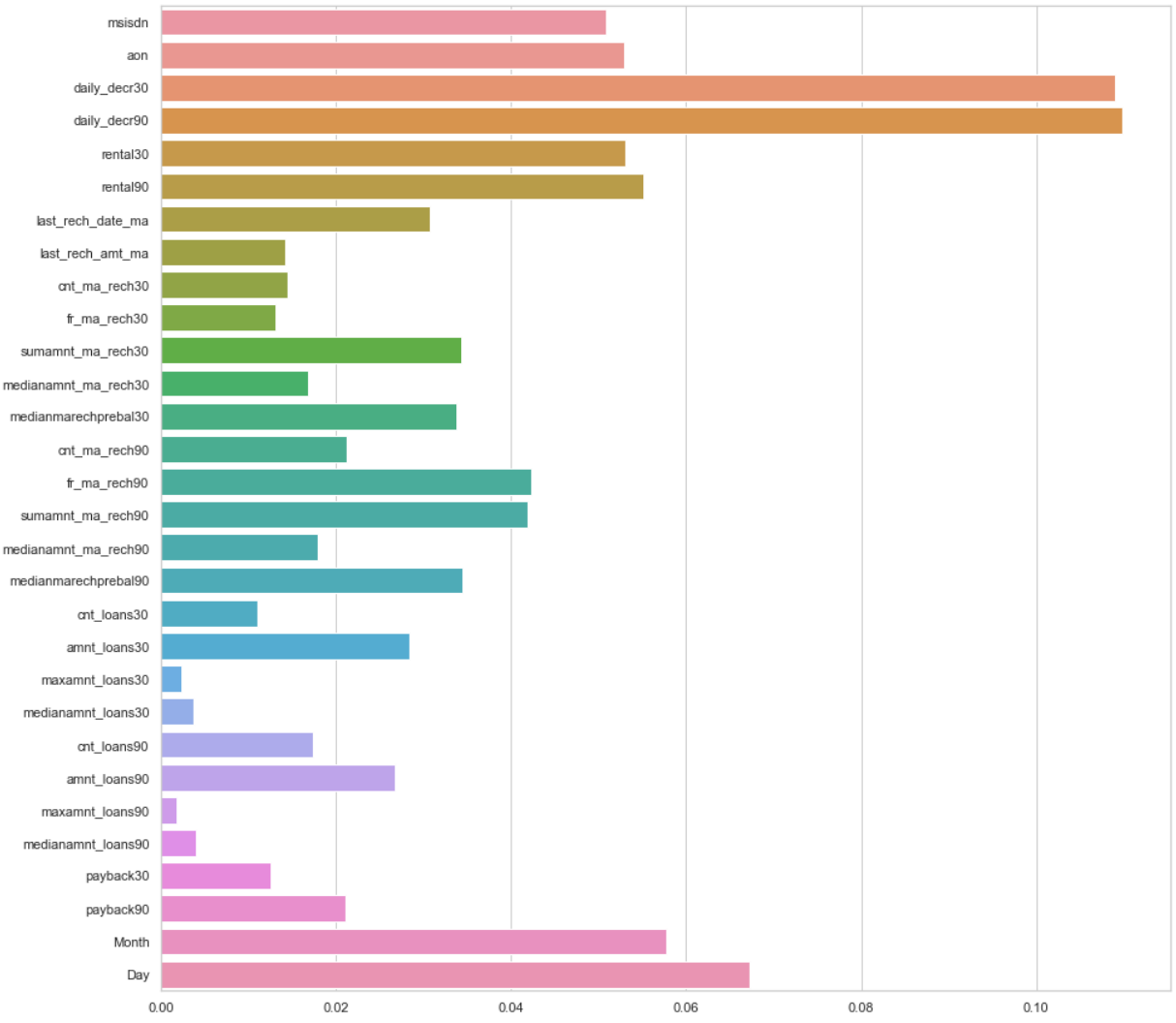
In [313]:
```
sc = StandardScaler()
```

In [314]:
```
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [315]:
```
X_train
```

Out[315]:
```
array([[ 0.77625116,  0.97272996,  0.40403907, ..., -1.04248779,
         1.62448747,  0.42706287],
       [-0.82712238, -1.08306352, -1.34338781, ..., -1.04248779,
```

- **Data Inputs- Logic- Output Relationships**



The above plot, is a bar plot visualizing the feature importance's of all the independent variables in predicting the dependent variable Label and we can clearly conclude that the columns Daily Decrease 30, Daily

Decrease 90 and Day column are turning out to be the most important features in predicting the target variable.

- ## Hardware and Software Requirements and Tools Used

Hardware used:

- **OS: Windows 10 Home Single Language 64 bit**
- **Ram: 8 GB**
- **Processor: Intel I5**

Software used:

- **Jupyter Notebook**

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

- **Given the number of inputs the Machine learning Algorithm has to predict the Label column.**
- **If Label is 1 then the loan has been paid. if 0 then the loan is unpaid.**
- **Target Variable is Label. The target variable only has two distinct values.**
- **The best approach is to solve this as a classification problem using various classification algorithms and find out which algorithm gives the best results**

- ## Testing of Identified Approaches (Algorithms)

A total of 7 classification models were used in order to classify the target variable Label.

- **Logistic Regression: Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic**

regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

- **Naive Bayes: Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naïve) independence assumptions between the features (see Bayes classifier). They are among the simplest Bayesian network models, but coupled with kernel density estimation, they can achieve higher accuracy levels.**
- **Random Forest Classification: Random Forest is a robust machine learning algorithm that can be used for a variety of tasks including regression and classification. It is an ensemble method, meaning that a random forest model is made up of a large number of small decision trees, called estimators, which each produce their own predictions. The random forest model combines the predictions of the estimators to produce a more accurate prediction.**
- **Decision Tree Classification: Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees**
- **ADA Boost Classifier: AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique that is used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights to incorrectly classified instances.**
- **K-NN Classifier: A k-Nearest-Neighbor algorithm, often abbreviated k- Nearest-Neighbor, is an approach to data classification that estimates how likely a data point is to be a member of one group or the other depending on what group the data points nearest to it are in.**

- **Gradient Boosting Classifier: Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting.**

- # Run and evaluate selected models

Logistic Regression.

In [320]:
```
# Taking 8 as best random state
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra
mod_1 = LogisticRegression()
mod_1.fit(X_train,y_train)
train_score_1 = mod_1.score(X_train,y_train)
pred_1 = mod_1.predict(X_test)
test_score_1 = accuracy_score(y_test,pred_1)

print("The training accuracy is :",train_score_1)
print("The testing accuracy is :",test_score_1)

```

```
The training accuracy is : 0.8752306067661616
The testing accuracy is : 0.8750166987919616
```

Naive Bayes.

In [329]:
```
# Taking 64 as best random state
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra
mod_2 = GaussianNB()
mod_2.fit(X_train,y_train)
train_score_2 = mod_2.score(X_train,y_train)
pred_2 = mod_2.predict(X_test)
test_score_2 = accuracy_score(y_test,pred_2)

print("The training accuracy is :",train_score_2)
print("The testing accuracy is :",test_score_2)
print("\n")
```

```
The training accuracy is : 0.8751415448426785
The testing accuracy is : 0.8752838794633485
```

Random Forest Classification.

In [337]:

```
# Taking the best random state as 28
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra
mod_3 = RandomForestClassifier()
mod_3.fit(X_train,y_train)
train_score_3 = mod_3.score(X_train,y_train)
pred_3 = mod_3.predict(X_test)
test_score_3 = accuracy_score(y_test,pred_3)

print("The training accuracy is :",train_score_3)
print("The testing accuracy is :",test_score_3)
print("\n")
```

```
The training accuracy is : 0.9999681921701846
The testing accuracy is : 0.920723677932785
```

Decision Tree Classification.

In [343]:

```
# Taking 81 as best random state
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra
mod_4 = DecisionTreeClassifier()
mod_4.fit(X_train,y_train)
train_score_4 = mod_4.score(X_train,y_train)
pred_4 = mod_4.predict(X_test)
test_score_4 = accuracy_score(y_test,pred_4)

print("The training accuracy is :",train_score_4)
print("The testing accuracy is :",test_score_4)
print("\n")
```

```
The training accuracy is : 0.9999809153021108
The testing accuracy is : 0.8816389625756217
```

ADA Boost Classifier.

In [351]:

```
 # Taking the best random state as 2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra
mod_6 = AdaBoostClassifier()
mod_6.fit(X_train,y_train)
train_score_6 = mod_6.score(X_train,y_train)
pred_6 = mod_6.predict(X_test)
test_score_6 = accuracy_score(y_test,pred_6)

print("The training accuracy is :",train_score_6)
print("The testing accuracy is :",test_score_6)
print("\n")
```

```
The training accuracy is : 0.9083934501316844
The testing accuracy is : 0.9059714880054963
```

K-NN Classifier.

In [359]:

```
# Taking the best random state as 32
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra
mod_7 = KNeighborsClassifier()
mod_7.fit(X_train,y_train)
train_score_7 = mod_7.score(X_train,y_train)
pred_7 = mod_7.predict(X_test)
test_score_7 = accuracy_score(y_test,pred_7)

print("The training accuracy is :",train_score_7)
print("The testing accuracy is :",test_score_7)
print("\n")
```

```
The training accuracy is : 0.878551344198888
The testing accuracy is : 0.8658943872974675
```

Gradient Boosting Classifier.

In [367]:

```
# Taking 91 as the best random state
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra
mod_8 = GradientBoostingClassifier()
mod_8.fit(X_train,y_train)
train_score_8 = mod_8.score(X_train,y_train)
pred_8 = mod_8.predict(X_test)
test_score_8 = accuracy_score(y_test,pred_8)

print("The training accuracy is :",train_score_8)
print("The testing accuracy is :",test_score_8)
print("\n")
```

```
The training accuracy is : 0.9180312225657468
The testing accuracy is : 0.916048016183515
```

Algorithm Result Table

	Algorithm	Training_Acc	Accuracy	Cross_validation
0	Logistic Regression	0.875231	0.875017	0.875177
1	Naive Bayes	0.875142	0.875284	0.875177
2	Random Forest Classi	0.999968	0.920724	0.920222
3	Decision Tree	0.999981	0.881639	0.881012
4	ADA Boost	0.908393	0.905971	0.908337
5	K Means	0.878551	0.865894	0.862906
6	Gredient Boosting	0.918031	0.916048	0.916805

- ## Key Metrics for success in solving problem under consideration

Median:

It is the middle value in the sorted(ordered) data. Median is a better measure of centre than mean as it is not affected by outliers. Most of the data had enormous non-realistic values, those values were replaced with the medians of the data in order to reduce the outliers.

Measures of spread:

Quartiles were used in order to detect the outliers. The lower quartile (Q1) is the value between the lowest 25% of values and the highest 75% of values. It is also called the 25th percentile.

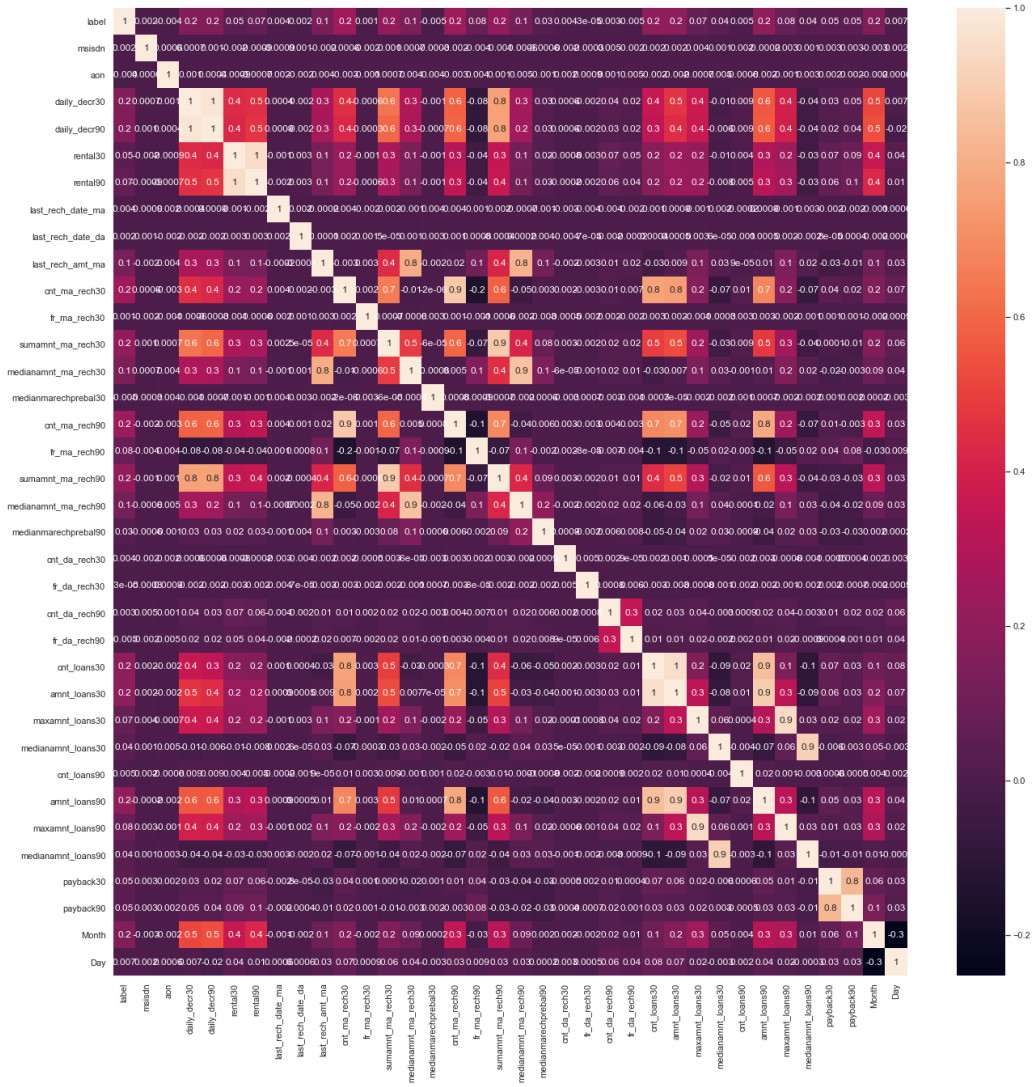
The second quartile (Q2) is the middle value of the data set. It is also called the 50th percentile, or the median.

The upper quartile (Q3) is the value between the lowest 75% and the highest 25% of values. It is also called the 75th percentile.

Correlation:

Correlation is simply, a metric that measures the extent to which variables (or features or samples or any groups) are associated with one another.

Pearson Correlation Coefficient is a statistic that also measures the linear correlation between two features.



```
In [89]:  1  cor["label"].sort_values(ascending=False)
```

```
Out[89]: label                  1.000000
         cnt_ma_rech30          0.237331
         cnt_ma_rech90          0.236392
         sumamnt_ma_rech90      0.205793
         sumamnt_ma_rech30      0.202828
         amnt_loans90           0.199788
         amnt_loans30           0.197272
         cnt_loans30            0.196283
         daily_decr30           0.158490
         daily_decr90           0.157658
         Month                  0.154949
         medianamnt_ma_rech30   0.141490
         last_rech_amt_ma       0.131804
         medianamnt_ma_rech90   0.120855
         fr_ma_rech90           0.084385
         maxamnt_loans90        0.084144
```

Stratified sampling:

In this method, the population is first divided into subgroups (or strata) which share a similar characteristic. It is used when we might

reasonably expect the measurement of interest to vary between the different subgroups, and we want to ensure representation from all the subgroups.

Stratified Cross Validation

```
In [369]:
1  kfold = StratifiedKFold(n_splits=10)
2
3  K_results = cross_val_score(mod_8,X,y,cv=kfold)
4
5  kfold_accuracy_GB = np.mean(abs(K_results))
```

```
In [370]:
1  kfold_accuracy_GB
```

```
Out[370]: 0.9168054360221284
```

- **Visualizations**

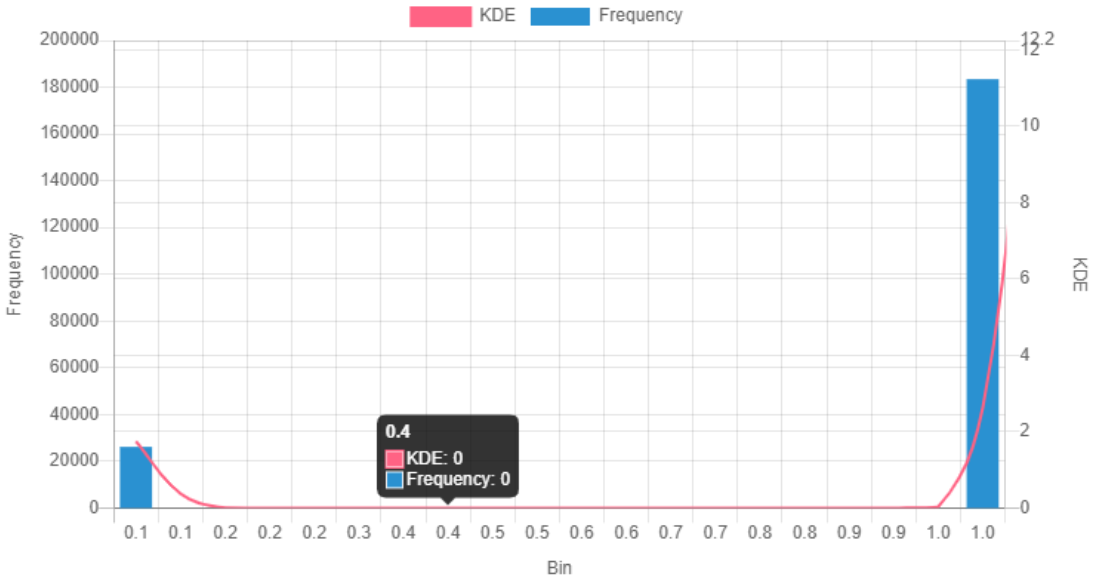


Fig. 1

The plot in Fig. 1 describes the distribution of "Label" column in the data set. A total amount of 26,162 individuals were not able to repay the loan amount on time. 1,83,431 individuals repaid their loan on time. The Loan repayment rate was 88 %.

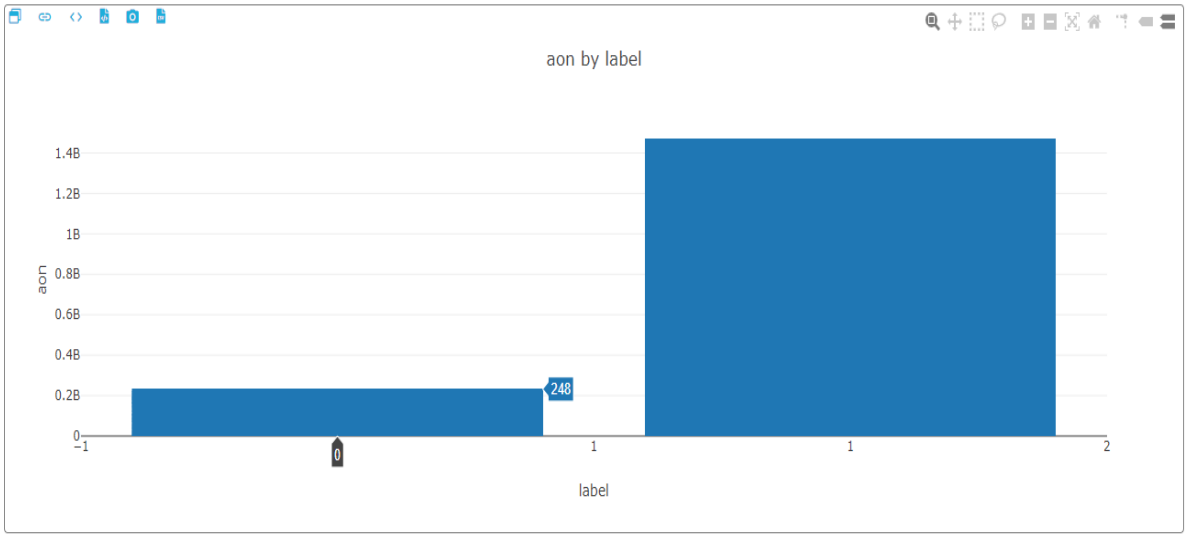


Fig.2(aon vs label)

The plot in Fig.2 shows the bivariate relationship between aon (age on cellular network in days) and the Label column. We can distinctly conclude that the age on cellular network in days is lower for the individuals who have failed to payback their loan in time.

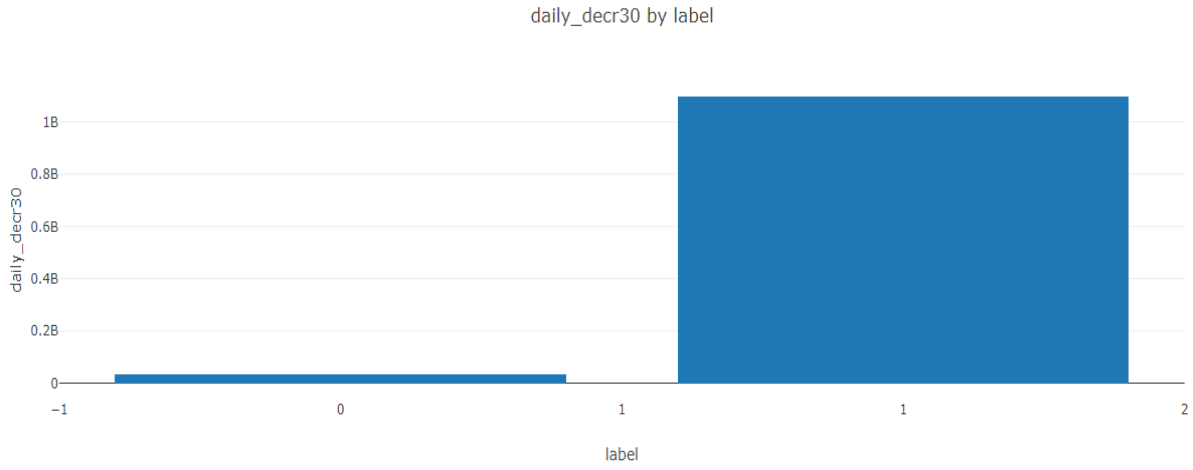


Fig. 3

The plot in Fig.3 shows the bivariate relationship between daily decrease 30 (Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah) and the Label column. We can distinctly conclude that the average amount spent by an individual per 30 days is very low for those individuals who have failed to repay the loan. So, basically the loan provider should check the average spending's of an individual before granting them loan.

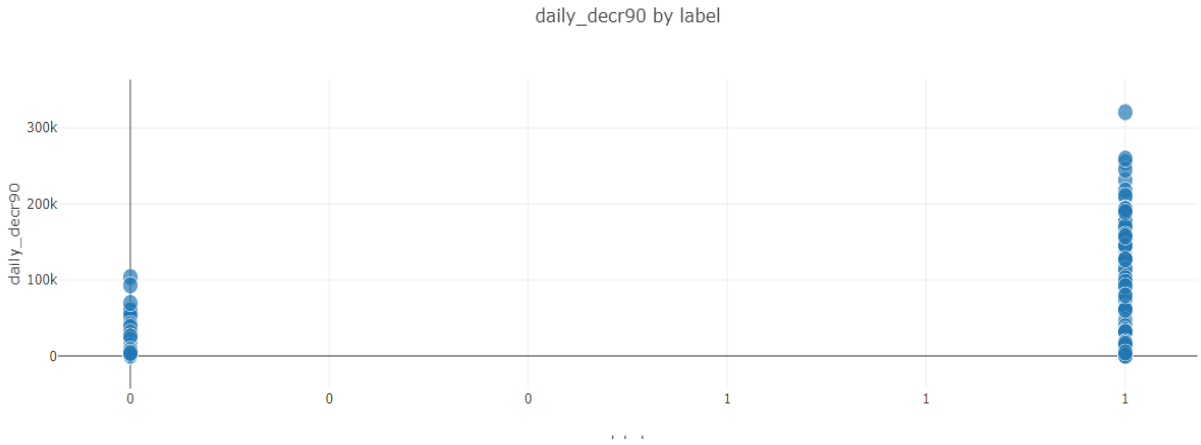


Fig. 4

The plot in Fig.4 shows the scatter plot of the bivariate relationship between daily decrease 90 (Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah) and the Label column. We can distinctly conclude that the average amount spent by an individual per 90 days (over 3 months). Individuals who have failed to repay the loan have very lower spending's as compared to the individuals who have paid their loan on time.

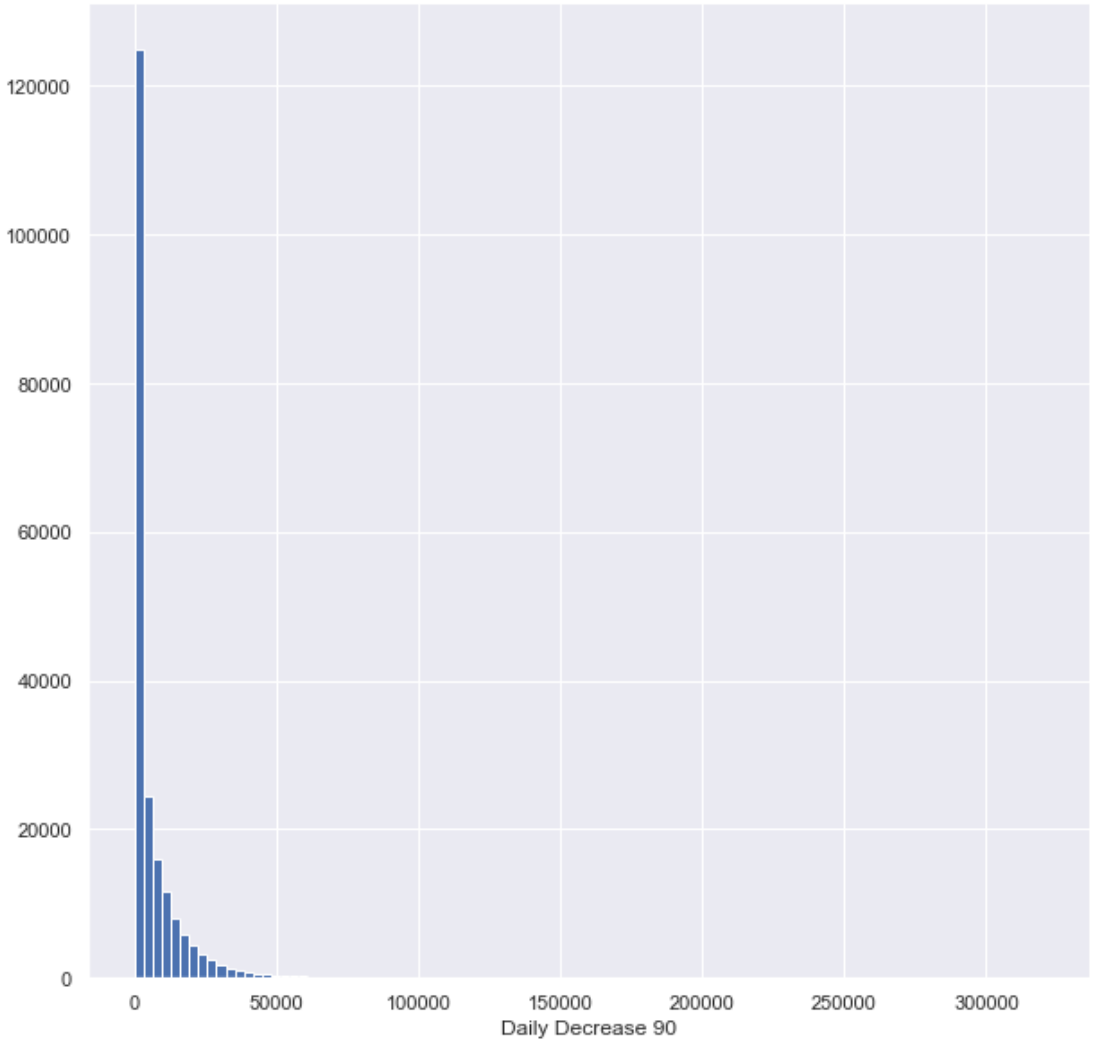


Fig. 5

The plot in Fig.5 shows the Hist plot for the column daily decrease 90 (Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah). 6124.76 (Indonesian Rupiah) was the average amount spent by an individual over 3 months.

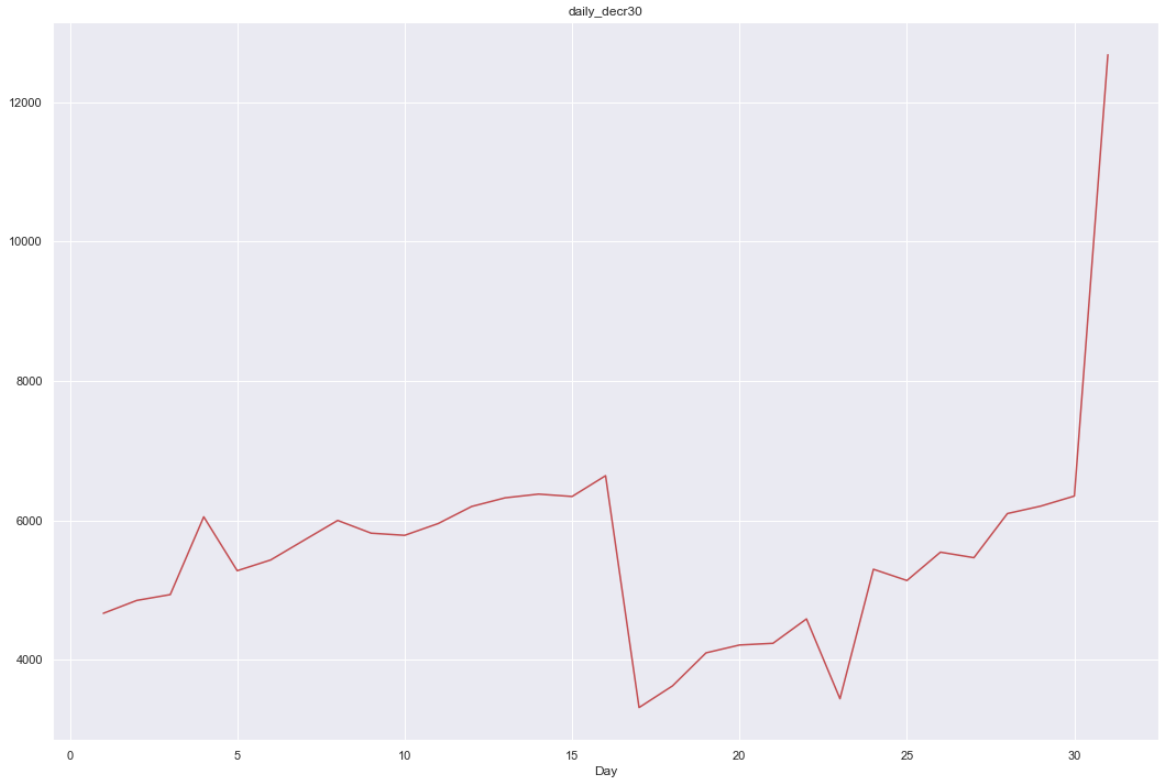


Fig. 6

The plot in Fig.6 is for the daily_decr30 i.e., the daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah). It is the average of the amount spent by an individual per month. The average amount spent by an individual per month is ~ 5423.35(in Indonesian Rupiah). From the above pattern we can observe that there is an exponential peak for the average amount spent during the last weeks of the month.

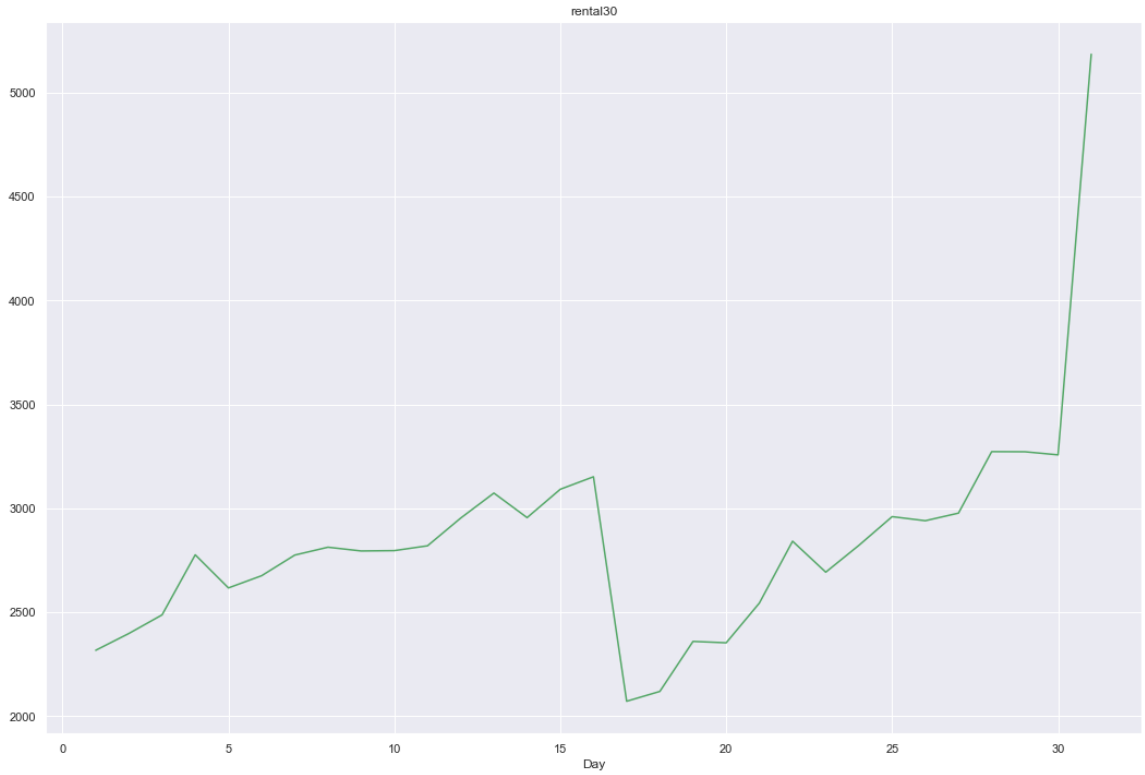


Fig. 7

The plot in Fig.7 is for the rental_30 i.e., the Average main account balance over last 30 days (in Indonesian Rupiah). The average main account balance per individual is ~ 2772.19. From the above plot we can say that, in the beginning of the month the main account balance is pretty low. As the month end approaches the account balance has an exponential increase.

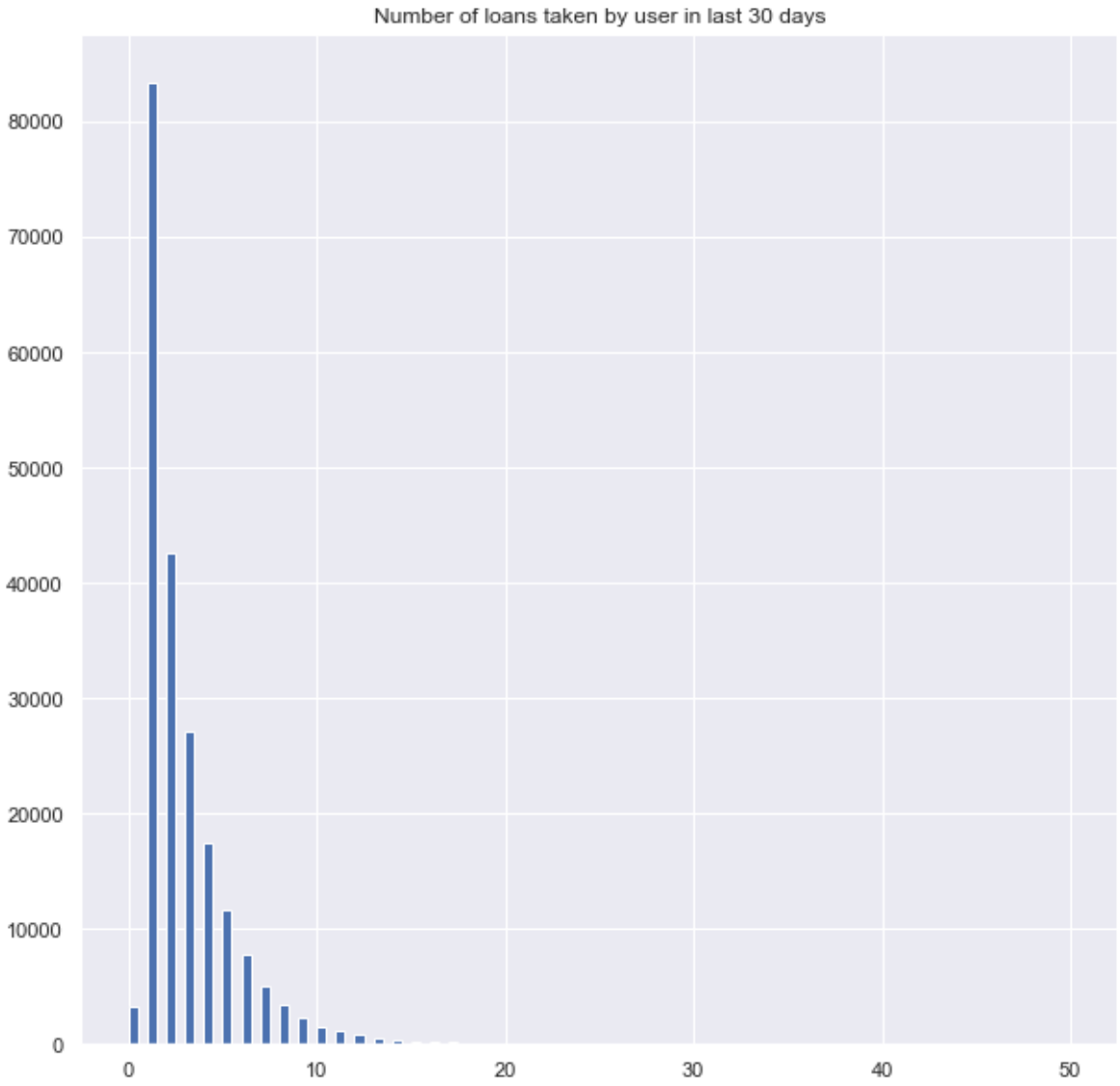


Fig. 8

The plot in Fig.8 is the Hist plot for the cnt_loans30 i.e., the Number of loans taken by user in last 30 days. On an average an individual took ~ 3 loans in a month (over 30 days).

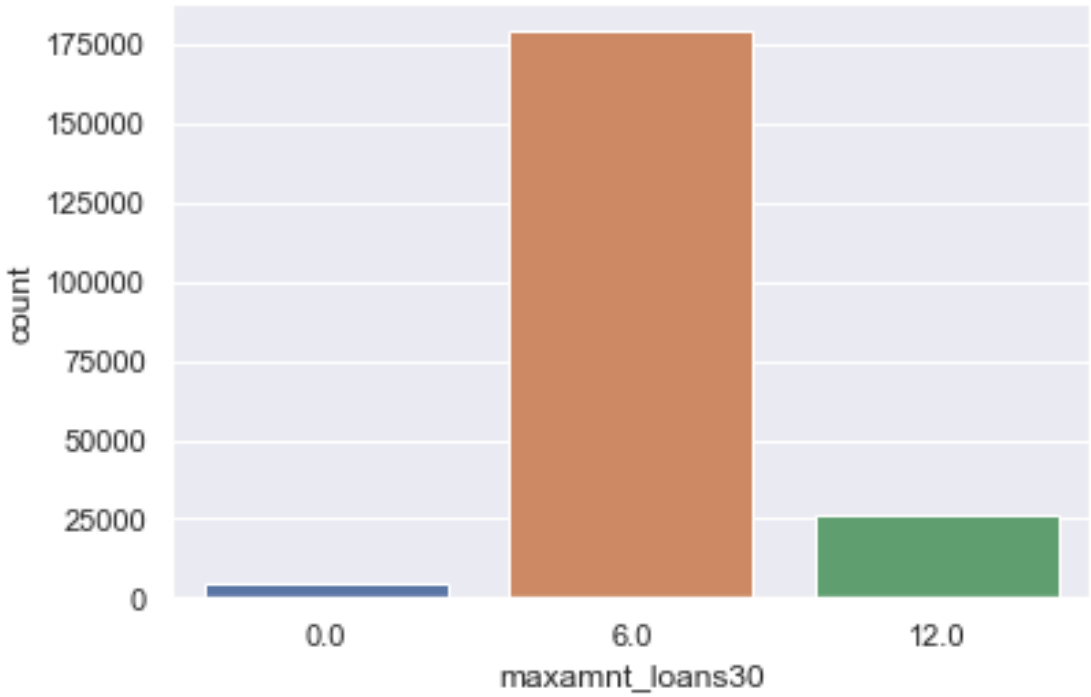


Fig. 8

The plot in Fig.8 is the seaborn count plot for the maxamnt_loans30 i.e., the maximum amount of loan taken by the user in last 30 days. User has two options 5 & 10 Rs., for which the user needs to pay back 6 & 12 Rs. Respectively. From the count plot above, we can say that most of the individuals have taken 5 Rs (in Indonesian Rupiah) loan so they have to repay 6 Rs. Very few Individuals have taken 12 Rs loan.

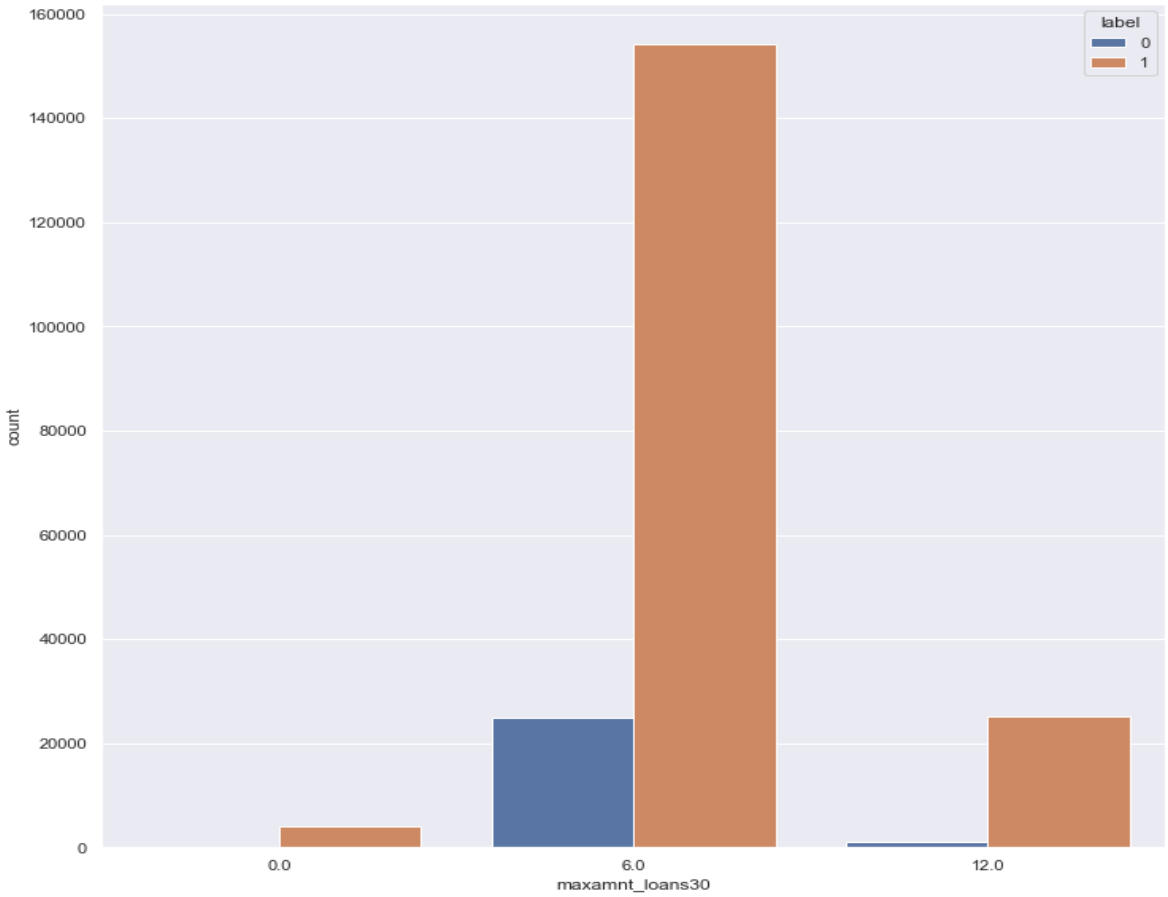


Fig. 9

The plot in Fig.9 is the seaborn count plot with hue as label for the maxamnt_loans30 i.e., the maximum amount of loan taken by the user in last 30 days. There is a high chance that individuals who have taken 5 Rs loan .i.e., those individuals who have to repay Rs 6 have a high chance that they might not repay the loan. Where else individuals who took 10 Rs loan .i.e., who have to repay Rs 12 are repaying the loan.

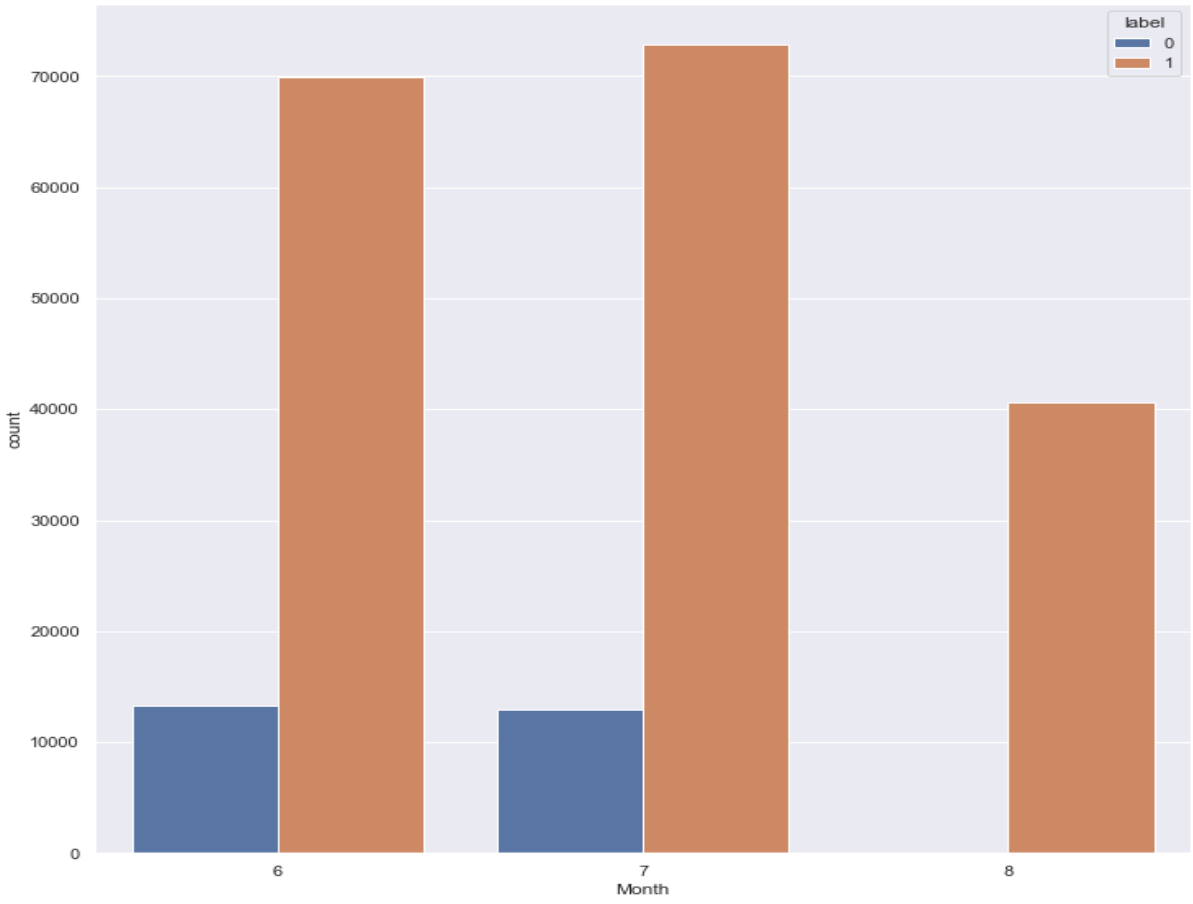


Fig. 10

The plot in Fig.10 is the seaborn count plot with hue as label for the Month column. We have the data for the months June, July, and August and we can conclude that the highest number of unpaid loans were in the month of June. No unpaid loans in the month of august.

- ## Interpretation of the Results

A total of 7 classification models were used in order to classify the target variable Label. Evaluation Metrix Like F1 Score, Recall, Precision, Accuracy was compared to find the optimum model

Logistic Regression:

Training Score	Testing Score	Cross Validation Score
0.87 %	0.875 %	0.875%

Confusion Matrix

In [321]:
```
1 print(confusion_matrix(y_test,pred_1))
2 print("\n")
3 print(classification_report(y_test,pred_1))
4 print("\n")
```

```
[[    0  6549]
 [    0 45850]]
```

Classification Report

	precision	recall	f1-score	support
0	0.00	0.00	0.00	6549
1	0.88	1.00	0.93	45850
accuracy			0.88	52399
macro avg	0.44	0.50	0.47	52399
weighted avg	0.77	0.88	0.82	52399

Naive Bayes:

Training Score	Testing Score	Cross Validation Score
0.87 %	0.875 %	0.875%

Confusion Matrix

In [330]:
```
1 print(confusion_matrix(y_test,pred_2))
2 print("\n")
3 print(classification_report(y_test,pred_2))
4 print("\n")
```

```
[[    0  6535]
 [    0 45864]]
```

Classification Report

	precision	recall	f1-score	support
0	0.00	0.00	0.00	6535
1	0.88	1.00	0.93	45864
accuracy			0.88	52399
macro avg	0.44	0.50	0.47	52399
weighted avg	0.77	0.88	0.82	52399

Random Forest Classification:

Training Score	Testing Score	Cross Validation Score
0.99 %	**0.92 %**	**0.92%**

Confusion Matrix

```
In [338]:
1 print(confusion_matrix(y_test,pred_3))
2 print("\n")
3 print(classification_report(y_test,pred_3))
4 print("\n")
```

```
[[ 3303  3189]
 [  965 44942]]
```

Classification Report

	precision	recall	f1-score	support
0	0.77	0.51	0.61	6492
1	0.93	0.98	0.96	45907
accuracy			0.92	52399
macro avg	0.85	0.74	0.78	52399
weighted avg	0.91	0.92	0.91	52399

Decision Tree Classification.

Training Score	Testing Score	Cross Validation Score
0.99 %	0.88 %	0.88%

Confusion Matrix

In [344]:
```
print(confusion_matrix(y_test,pred_4))
print("\n")
print(classification_report(y_test,pred_4))
print("\n")
```

```
[[ 3499  2896]
 [ 3306 42698]]
```

Classification Report

	precision	recall	f1-score	support
0	0.51	0.55	0.53	6395
1	0.94	0.93	0.93	46004
accuracy			0.88	52399
macro avg	0.73	0.74	0.73	52399
weighted avg	0.88	0.88	0.88	52399

ADA Boost Classifier.

Training Score	Testing Score	Cross Validation Score
0.90 %	0.90 %	0.90 %

Confusion Matrix

In [352]:
```
print(confusion_matrix(y_test,pred_6))
print("\n")
print(classification_report(y_test,pred_6))
print("\n")
```

```
[[ 2598  4017]
 [  910 44874]]
```

Classification Report

	precision	recall	f1-score	support
0	0.74	0.39	0.51	6615
1	0.92	0.98	0.95	45784
accuracy			0.91	52399
macro avg	0.83	0.69	0.73	52399
weighted avg	0.90	0.91	0.89	52399

K-NN Classifier.

Training Score	Testing Score	Cross Validation Score
0.87 %	0.86 %	0.86 %

Confusion Matrix

```
In [360]:
1 print(confusion_matrix(y_test,pred_7))
2 print("\n")
3 print(classification_report(y_test,pred_7))
4 print("\n")
```

```
[[  114  6294]
 [  733 45258]]
```

Classification Report

	precision	recall	f1-score	support
0	0.13	0.02	0.03	6408
1	0.88	0.98	0.93	45991
accuracy			0.87	52399
macro avg	0.51	0.50	0.48	52399
weighted avg	0.79	0.87	0.82	52399

Gradient Boosting Classifier.

Training Score	Testing Score	Cross Validation Score
0.91 %	**0.91 %**	**0.91 %**

Confusion Matrix

In [368]:
```
print(confusion_matrix(y_test,pred_8))
print("\n")
print(classification_report(y_test,pred_8))
print("\n")
```

```
[[ 3145  3363]
 [ 1036 44855]]
```

Classification Report

	precision	recall	f1-score	support
0	0.75	0.48	0.59	6508
1	0.93	0.98	0.95	45891
accuracy			0.92	52399
macro avg	0.84	0.73	0.77	52399
weighted avg	0.91	0.92	0.91	52399

From all the above algorithms Logistic Regression and Naive Bayes are not giving appropriate results in terms of classification. Random forest classifier, Ada Boost Classifier and Gradient boosting algorithms are giving better results.

Based on the F1 Score we performed Hyper parametric tuning on Random Forest classification.

Random Forest Classification (Hyper parametric tuning)

Parameters

In [376]:
```
n_estimators = [100,150,200,250,300]
criterion = ["gini","entropy"]
max_features = ["auto","sqrt","log2"]
class_weight = ["balanced","balanced_subsample"]
min_samples_leaf = [1,2,3,4,5]
```

In [377]:
```
parameters = {
    "n_estimators":n_estimators,
    "criterion":criterion,
    "max_features":max_features,
    "class_weight":class_weight,
    "min_samples_leaf":min_samples_leaf
   }
```

Random Search CV

```
In [381]:  1  random_search_rfr.fit(X_train,y_train)

Fitting 3 folds for each of 10 candidates, totalling 30 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed: 12.5min finished

Out[381]: RandomizedSearchCV(cv=3, estimator=RandomForestClassifier(), n_jobs=-1,
                   param_distributions={'class_weight': ['balanced',
                                                         'balanced_subsample'],
                                        'criterion': ['gini', 'entropy'],
                                        'max_features': ['auto', 'sqrt',
                                                         'log2'],
                                        'min_samples_leaf': [1, 2, 3, 4, 5],
                                        'n_estimators': [100, 150, 200, 250,
                                                         300]},
                   verbose=2)
```

Best Estimators

Best Estimators

```
In [382]:  1  random_search_rfr.best_estimator_

Out[382]: RandomForestClassifier(class_weight='balanced_subsample', criterion='entropy',
                       max_features='sqrt', min_samples_leaf=2)
```

Best Parameters

Best Parameters

```
In [383]:  1  random_search_rfr.best_params_

Out[383]: {'n_estimators': 100,
           'min_samples_leaf': 2,
           'max_features': 'sqrt',
           'criterion': 'entropy',
           'class_weight': 'balanced_subsample'}
```

Best Score 0.92 %

Classification Report:

Classification Report

```
7]:  1 print(classification_report(y_test,predictions_rfr))
     2 print("\n")
```

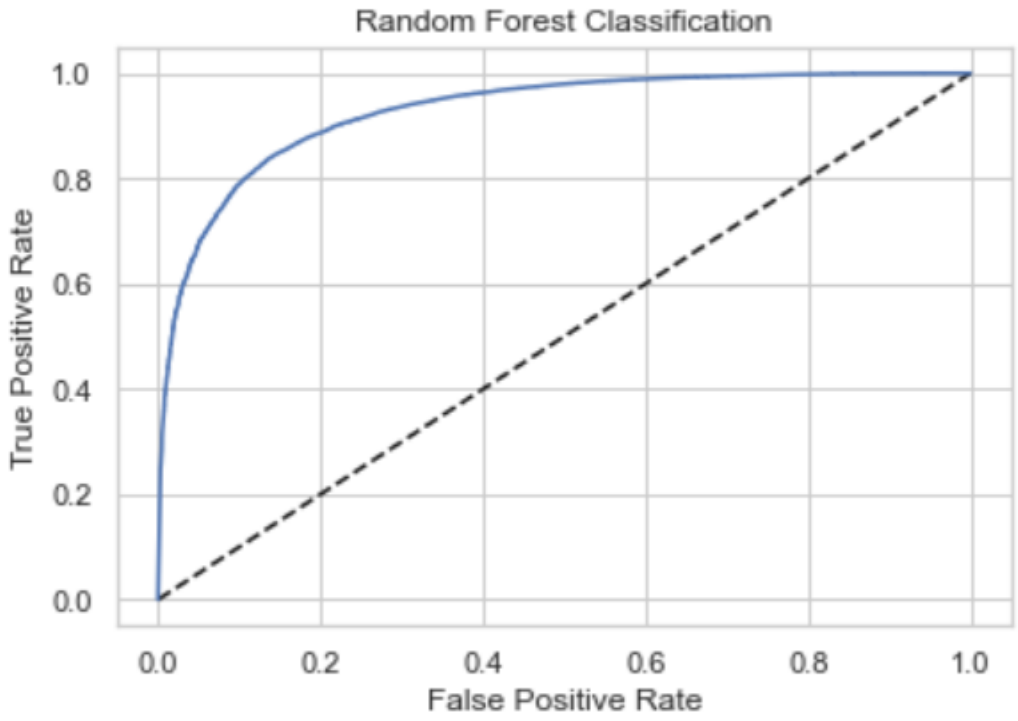
	precision	recall	f1-score	support
0	0.72	0.57	0.64	6508
1	0.94	0.97	0.95	45891
accuracy			0.92	52399
macro avg	0.83	0.77	0.80	52399
weighted avg	0.91	0.92	0.92	52399

CONCLUSION

- ## Key Findings and Conclusions of the Study

The first step I took, was to visualize the distribution of each feature and its effect on the Label (whether an Individual will repay the loan in time or not). From the analysis, I conclude that the most useful features for predictions were "Daily Decrease 30", "Daily Decrease 90", "Day". The Random Forest Classifier proved to be the best model for classification based on the Cross-Validation scores. An accuracy of 0.92 % was achieved by hyperparametric tuning of the model.

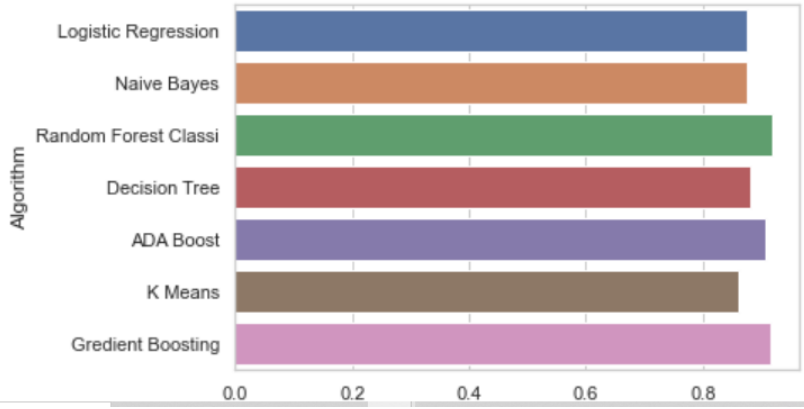
ROC AUC Curve



The more the area under the curve the better is the model performance.

Algorithm Performance:

In [373]:
```
# Algorithm performance
sns.set(style="whitegrid")
ax = sns.barplot(y="Algorithm",x="Cross_validation",data=final_card)
```



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

New Skills Acquired: Learned to perform EDA with the help of D Tale Library.

Algorithms Used: Logistic regression, Naive bayes, Random Forest classifier, Decision Tree, Ada Boost, K-NN Neighbor, Gradient boosting classifier.

Challenges Faced: The Feature Engineering and data pre-processing was quit challenging as there were a lot of non-realistic values in the dataset.

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

Even after getting a F1 Score for 1 to be 0.95 % the F1 score for 0 was just 64 % which can be improved. Even the Recall for 0 can be improved. But considering how unbalanced the data set was I think the results are somewhat fair. We can implement over sampling for handling imbalanced data.