FLIP ROBO

MICRO CREDIT DEFAULTER PROJECT

Submitted by:

CHRIS CHHOTAI

ACKNOWLEDGMENT

I'd want to express my heartfelt gratitude to Swati Mahasetg, my SME (Subject Matter Expert), as well as Flip Robo Technologies, for allowing me to work on this Micro Credit Loan Defaulter Project and for assisting me in conducting extensive research that allowed me to learn about many new things.

Also, I have utilized a few external resources that helped me to complete the project. I ensured that I learn from the samples and modify things according to my project requirement. All the external resources that were used in creating this project are listed below:

1) https://www.google.com/
2) https://www.youtube.com/
3) https://scikit-learn.org/stable/user_guide.html
4) https://github.com/
5) https://www.kaggle.com/
6) https://medium.com/
7) https://towardsdatascience.com/
8) https://www.analyticsvidhya.com/

INTRODUCTION

- Business Problem Framing:

The main goal of this project is to create a model that can be used to estimate if a customer will pay back the lent amount within 5 days of loan insurance in terms of probability for each loan transaction.

- Conceptual Background of the Domain Problem:

Microfinance Institutions (MFIs) are businesses that provide financial services to low-income people. When addressing unbanked poor families living in rural places with few sources of income, MFS becomes quite effective. Group Loans, Agricultural Loans, Individual Business Loans, and other Microfinance Services (MFS) are some of the Microfinance Services (MFS) provided by MFI.

Many microfinance institutions (MFI), experts, and donors promote the use of mobile financial services (MFS), which they believe are more convenient, efficient, and cost-effective than the traditional high-touch strategy used to deliver microfinance services for a long time. Despite the fact that the MFI industry focuses primarily on low-income households and is particularly valuable in these areas, MFS implementation has been uneven, with both substantial successes and failures.

Microfinance is now widely recognised as a strategy for poverty reduction, with $70 billion in outstanding loans and a global client base of 200 million people.

FlipRobo is now working with a client in the telecommunications industry. They are a provider of fixed wireless telecommunications networks.

They've released a number of products and built their business and organisation around the budget operator model, which entails providing better products at lower prices to all value-conscious clients via a disruptive innovation strategy that focuses on the subscriber.

They recognise the value of communication and how it influences a person's life, thus they focus on giving low-income families and

impoverished consumers with services and products that can assist them in their time of need.

- ## Review of Literature:

1. What is Microfinance and How Does It Work?

Microfinance is frequently thought of as financial services for the poor and low-income. In actuality, the phrase is more commonly used to refer to loans and other services provided by companies that call themselves "microfinance institutions" (MFIs). Microfinance can also be defined as a collection of different operators working together to meet the needs of the financially underserved in terms of poverty alleviation, social promotion, emancipation, and inclusion. Microfinance organisations use cutting-edge methods to reach and serve their target customers. Microfinance operations are fundamentally different from traditional financial disciplines such as general and entrepreneurial finance. This disparity can be explained by the fact that microcredit loans are often too little to fund growth-oriented business projects. The following are some of the unique characteristics of microfinance:

i. Unsalaried labourers are given relatively tiny loans.

ii. There are few, if any, collateral requirements.

iv. Liability and group lending

iv. The need to save money before taking out a loan.

v. Gradually increasing the magnitude of the loans.

If current loans are repaid in whole and on time, there is an implicit assurance of easy access to future loans. Microfinance is considered as a catalyst for poverty alleviation, especially in developing nations, when it is delivered in innovative and sustainable methods to help the underserved poor.

2. Microfinance Default

In microfinance, default refers to a client's failure to repay a loan. The default could be in terms of the amount to be paid or the timing of the payment.

- Motivation for the Problem Undertaken:

The main goal of this research is to create a model that can predict whether consumers will pay their loans on time or not. Machine Learning methods will be used to predict. We obtained the sample data from our client database. To put it another way, the client seeks to increase the credit selection of clients. Certain forecasts that could aid in future investment and improvement in customer selection

Analytical Problem Framing

- Analytical Modelling of the Problem:

On the basis of accounts that have been recharged in the recent 30 days, I have performed several analytics before moving further with exploratory study. I established a criteria that if a person does not recharge their main account within three months, I just delete their information because they are not important and may be old clients, but there is no revenue rotation. Then I looked at the date fields and saw that the data was from 2016. I extracted the month and day from the date, separated the data into columns, then attempted to visualise the data using months and days.

I looked at the maximum amount of loan taken by the folks and discovered that there were more outliers in the data. According to the client's description, the loan amount that the customer can pay is either rupiah 6 or 12, so I've removed all the loan amounts that suggest the loan was accepted for more than 12 rupiah.

Then I segregated the defaulters' data and verified the network's valuable clients, discovering that their monthly revenue exceeds 10,000 rupiah. We deleted some columns despite the fact that the data is extremely unbalanced and many columns do not have the expected maximum value. Before model processing, we checked the skewed data and attempted to treat the skewed data, which resulted in NaN.

When we tried to remove the undesired data, i.e. the outliers, we discovered that about 40000 records were chopped. Despite the fact that the data provided by the client included about 37 columns and over 2 lakh columns, I did not want to risk losing any valuable data, therefore I skipped the outlier reduction step as well. I sent my data to multiple classification models after scaling it and discovered that the Extra Trees Classifier Algorithm works effectively.

- Data Sources and their formats:

The information was provided by a client in the telecom business. They are a fixed wireless telecommunications network provider who has launched a variety of products and built their business and organisation around the budget operator model, providing better products at lower prices to all value-conscious customers through a disruptive innovation strategy that focuses on the subscriber.
The data was provided by an Indonesian telecom firm in the form of a CSV file with an excel sheet describing the data. They also gave a problem statement, outlining what they expect of us as well as the criteria that must be met.
Let's take a look at the facts right now. I've attached a screenshot to this message which gives an overview.

```
In [9]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
In [10]:
#Importing the dataset
df=pd.read_csv('Data file.csv')
df.head()
```

Out[10]:

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	...	maxamnt_loans30	media
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0	

5 rows × 37 columns

We're looking at the first five and last five rows of our dataset. It reveals that our dataframe contains a total of 209593 rows and 37 columns. This

is a Classification challenge because we have the label column that stores the defaulter and non-defaulter values labelled with 0 and 1.

- ## Data Preprocessing Done:

Checked for missing values to confirm the information of no null values present provided in the problem statement.

```
In [18]: #Checking null values in the dataset
         df.isnull().sum()
```

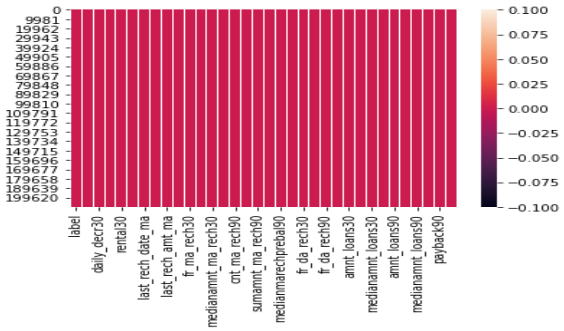
```
Out[18]: label                    0
         aon                      0
         daily_decr30             0
         daily_decr90             0
         rental30                 0
         rental90                 0
         last_rech_date_ma        0
         last_rech_date_da        0
         last_rech_amt_ma         0
         cnt_ma_rech30            0
         fr_ma_rech30             0
         sumamnt_ma_rech30        0
         medianamnt_ma_rech30     0
         medianmarechprebal30     0
         cnt_ma_rech90            0
         fr_ma_rech90             0
         sumamnt_ma_rech90        0
         medianamnt_ma_rech90     0
         medianmarechprebal90     0
         cnt_da_rech30            0
         fr_da_rech30             0
         cnt_da_rech90            0
         fr_da_rech90             0
         cnt_loans30              0
         amnt_loans30             0
         maxamnt_loans30          0
         medianamnt_loans30       0
         cnt_loans90              0
         amnt_loans90             0
         maxamnt_loans90          0
         medianamnt_loans90       0
         payback30                0
         payback90                0
         pdate                    0
         dtype: int64
```

There is no null values in our dataset.

Took a visual on the missing data information as well.

```
In [19]: import seaborn as sns
         sns.heatmap(df.isnull())
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x2209a44280>
```



We may confirm the non-null count details as well as the datatype information using the info method. We have 21 columns with float/decimal datatypes, 12 columns with integer datatypes, and three columns with object/categorical datatypes. Before we can use the

information in our machine learning models, we'll need to convert the object datatype columns to numerical data.

```
In [15]: #Checking the info about the dataset
         df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209593 entries, 0 to 209592
Data columns (total 37 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Unnamed: 0            209593 non-null  int64
 1   label                 209593 non-null  int64
 2   msisdn                209593 non-null  object
 3   aon                   209593 non-null  float64
 4   daily_decr30          209593 non-null  float64
 5   daily_decr90          209593 non-null  float64
 6   rental30              209593 non-null  float64
 7   rental90              209593 non-null  float64
 8   last_rech_date_ma     209593 non-null  float64
 9   last_rech_date_da     209593 non-null  float64
 10  last_rech_amt_ma      209593 non-null  int64
 11  cnt_ma_rech30         209593 non-null  int64
 12  fr_ma_rech30          209593 non-null  float64
 13  sumamnt_ma_rech30     209593 non-null  float64
 14  medianamnt_ma_rech30  209593 non-null  float64
 15  medianmarechprebal30  209593 non-null  float64
 16  cnt_ma_rech90         209593 non-null  int64
 17  fr_ma_rech90          209593 non-null  int64
 18  sumamnt_ma_rech90     209593 non-null  int64
 19  medianamnt_ma_rech90  209593 non-null  float64
 20  medianmarechprebal90  209593 non-null  float64
 21  cnt_da_rech30         209593 non-null  float64
 22  fr_da_rech30          209593 non-null  float64
 23  cnt_da_rech90         209593 non-null  int64
 24  fr_da_rech90          209593 non-null  int64
 25  cnt_loans30           209593 non-null  int64
 26  amnt_loans30          209593 non-null  int64
 27  maxamnt_loans30       209593 non-null  float64
 28  medianamnt_loans30    209593 non-null  float64
 29  cnt_loans90           209593 non-null  float64
 30  amnt_loans90          209593 non-null  int64
 31  maxamnt_loans90       209593 non-null  int64
 32  medianamnt_loans90    209593 non-null  float64
 33  payback30             209593 non-null  float64
 34  payback90             209593 non-null  float64
 35  pcircle               209593 non-null  object
 36  pdate                 209593 non-null  object
dtypes: float64(21), int64(13), object(3)
memory usage: 59.2+ MB
```

- ## Data Inputs- Logic- Output Relationships:

- Data description on each column present in our dataset.
- 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 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 Indonesian Rupiah)
- medianamnt_ma_rech90 : Median of amount 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
- 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

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

I had assumed that any telecom operator would keep customer data for three months, therefore I had cut off my data based on that assumption. Because the data is from 2016, just the date and months are different, I removed the 2016 year from the pdate columns. Months and days were divided into separate columns.
Then I looked at the data of defaulters separately and discovered that many valuable consumers are defaulters because they may have forgotten to pay or are too preoccupied with their lives. I separated them so that the company could deal with them courteously, as we cannot afford to lose these consumers.

Model/s Development and Evaluation :

- Identification of possible problem-solving approaches (methods)

- ➢ Performed EDA (Exploratory Data Analysis).
- ➢ Data Cleaning and dropping the columns which were not contributing to the dataset.
- ➢ Checked for the outliers and tried to remove the outliers of the dataset.
- ➢ Checked for the skewness in the dataset and removed the skewness for better model building.
- ➢ Train- Test the dataset into independent and dependent variables.
- ➢ Model Building.

➢ Cross validation score to check if the model is over-fitted.

- Testing of Identified Approaches (Algorithms)

1. Logistic Regression.
2. Decision Tree Classifier.
3. Bagging Classifier.
4. AdaBoost Classifier.

- Run and Evaluate selected models

Describe all the algorithms used along with the snapshot of their code and what were the results observed over different evaluation metrics.

1. Logistic Regression

Logistic Regression:

In [71]:

```
LR=LogisticRegression()
LR.fit(X_train,y_train)
preddt=LR.predict(X_test)
Accuracy_Score = accuracy_score(y_test, preddt)*100
print('Accuracy Score:',Accuracy_Score)
print('Confusion Matrix:',confusion_matrix(y_test, preddt))
print(classification_report(y_test,preddt))

#cross validation score
scores = cross_val_score(LR, X, y, cv = 5).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = Accuracy_Score - scores
print("Accuracy_Score - Cross Validation Score :", diff)
```

```
Accuracy Score: 88.595375170966
Confusion Matrix: [[ 1601  6329]
 [  842 54106]]
              precision    recall  f1-score   support

           0       0.66      0.20      0.31      7930
           1       0.90      0.98      0.94     54948

    accuracy                           0.89     62878
   macro avg       0.78      0.59      0.62     62878
weighted avg       0.87      0.89      0.86     62878


Cross validation score : 88.60362723740087
Accuracy_Score - Cross Validation Score : -0.008252066434877747
```

From Logistic Regression we got an accuracy of 89%

2. Decision Tree Classifier

Decision Tree Classifier:

In [73]:

```
DTC=DecisionTreeClassifier()
DTC.fit(X_train,y_train)
preddt=DTC.predict(X_test)
Accuracy_Score = accuracy_score(y_test, preddt)*100
print('Accuracy Score:',Accuracy_Score)
print('Confusion Matrix:',confusion_matrix(y_test, preddt))
print(classification_report(y_test,preddt))

#cross validation score
scores = cross_val_score(DTC, X, y, cv = 5).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = Accuracy_Score - scores
print("Accuracy_Score - Cross Validation Score :", diff)
```

```
Accuracy Score: 88.22799707369828
Confusion Matrix: [[ 4438  3492]
 [ 3910 51038]]
              precision    recall  f1-score   support

           0       0.53      0.56      0.55      7930
           1       0.94      0.93      0.93     54948

    accuracy                           0.88     62878
   macro avg       0.73      0.74      0.74     62878
weighted avg       0.88      0.88      0.88     62878


Cross validation score : 88.30733914196104
Accuracy_Score - Cross Validation Score : -0.07934206826276125
```

From this model we got an accuracy of 88%.

3. Bagging Classifier

Bagging Classifier:

In [75]:

```
BC=BaggingClassifier()
BC.fit(X_train,y_train)
predbc=BC.predict(X_test)
Accuracy_Score = accuracy_score(y_test, predbc)*100
print('Accuracy Score:',Accuracy_Score)
print('Confusion Matrix:',confusion_matrix(y_test, predbc))
print(classification_report(y_test,predbc))

#cross validation score
scores = cross_val_score(BC, X, y, cv = 5).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = Accuracy_Score - scores
print("Accuracy_Score - Cross Validation Score :", diff)
```

```
Accuracy Score: 91.30697541270396
Confusion Matrix: [[ 4678  3252]
 [ 2214 52734]]
              precision    recall  f1-score   support

           0       0.68      0.59      0.63      7930
           1       0.94      0.96      0.95     54948

    accuracy                           0.91     62878
   macro avg       0.81      0.77      0.79     62878
weighted avg       0.91      0.91      0.91     62878


Cross validation score : 91.36851022184459
Accuracy_Score - Cross Validation Score : -0.0615348091406247
```

From this model we got an accuracy of 91%.

4. AdaBoost Classifier

AdaBoost Classifier:

In [77]:

```
ABC=AdaBoostClassifier()
ABC.fit(X_train,y_train)
predab=ABC.predict(X_test)
Accuracy_Score = accuracy_score(y_test, predab)*100
print('Accuracy Score:',Accuracy_Score)
print('Confusion Matrix:',confusion_matrix(y_test, predab))
print(classification_report(y_test,predab))

#cross validation score
scores = cross_val_score(ABC, X, y, cv = 5).mean()*100
print("\nCross validation score :", scores)

#difference of accuracy and cv score
diff = Accuracy_Score - scores
print("Accuracy_Score - Cross Validation Score :", diff)
```

```
Accuracy Score: 91.00003180762747
Confusion Matrix: [[ 3182  4748]
 [  911 54037]]
              precision    recall  f1-score   support

           0       0.78      0.40      0.53      7930
           1       0.92      0.98      0.95     54948

    accuracy                           0.91     62878
   macro avg       0.85      0.69      0.74     62878
weighted avg       0.90      0.91      0.90     62878


Cross validation score : 90.99015691474838
Accuracy_Score - Cross Validation Score : 0.009874892879082608
```

From this model we got an accuracy of 91%.

- Key Metrics for success in solving problem under consideration:

  The accuracy score, cross val score, classification report, auc score, and confusion matrix were the main metrics employed in this study. We used Hyperparameter Tuning to identify the optimal parameters and to improve our scores, and we'll be using the GridSearchCV method to do it.

1. Cross Validation:

Cross-validation aids in determining the model's overfitting and underfitting. The model is constructed to run on several subsets of the dataset in cross validation, resulting in numerous

measurements of the model. If we fold the data five times, it will be separated into five pieces, each representing 20% of the total dataset. During the Cross-validation, the first part (20%) of the 5 parts will be left out as a holdout set for validation, while the rest of the data will be used for training. We'll acquire the initial estimate of the dataset's model quality this way. Further iterations are made in the same way for the second 20% of the dataset, which is kept as a holdout set while the remaining four portions are used for training data during the process. We'll get the second estimate of the dataset's model quality this way. During the cross-validation procedure, these stages are repeated to obtain the remaining estimate of model quality.

2. Confusion Matrix:

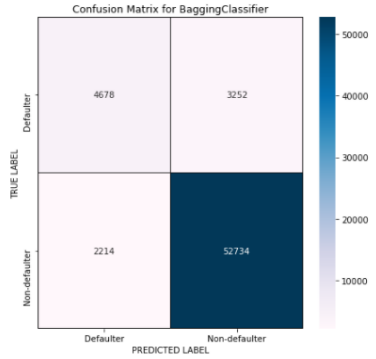
A confusion matrix, sometimes known as an error matrix, is a table structure that permits visualisation of an algorithm's performance, usually a supervised learning algorithm (in unsupervised learning it is usually called a matching matrix). The instances in a predicted class are represented by the rows of the matrix, whereas the instances in an actual class are represented by the columns (or vice versa). The name comes from the fact that it's simple to tell if the system is mixing up two types (i.e., commonly mislabelling one as another).

It's a unique type of contingency table, with two dimensions ("actual" and "predicted") and identical sets of "classes" in each (each combination of dimension and class is a variable in the contingency table).

```
In [76]:
cm = confusion_matrix(y_test, predbc)

x_axis_labels = ["Defaulter","Non-defaulter"]
y_axis_labels = ["Defaulter","Non-defaulter"]

f, ax = plt.subplots(figsize =(7,7))
sns.heatmap(cm, annot = True, linewidths=0.2, linecolor="black", fmt = ".0f", ax=ax, cmap="PuBu", xticklabels=x_axis_labels,
plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for BaggingClassifier')
plt.show()
```



3. Classification Report:

The precision, recall, F1, and support scores for the model are displayed in the classification report visualizer. There are four techniques to determine if the forecasts are correct or incorrect: 1. True Negative (TN): the case was negative and was expected to be negative. 2. TP (True Positive): the case was positive and the outcome was expected to be positive. 3. False Negative (FN): the case was positive, but the outcome was projected to be negative. 4. FP / False Positive: the case was negative, but a positive outcome was projected.

Precision refers to a classifier's ability to avoid labelling something positive that is truly negative. It is calculated as the ratio of true positives to the sum of true positives and false positives for each class. It is the precision with which optimistic predictions are made. The precision formula is as follows: Precision is defined as TP/(TP + FP).

Recall: A classifier's capacity to discover all positive cases is known as recall. It is calculated as the ratio of true positives to the sum of true positives and false negatives for each class. It's also the percentage of positives that were detected correctly. The recall formula is as follows: TP/(TP+FN) = Recall

The F1 score is a weighted harmonic mean of precision and recall, with 1.0 being the highest and 0.0 being the poorest. F1 scores are lower than accuracy measurements because they factor on precision and recall. To compare classifier models, utilise the weighted average of F1 rather than global accuracy as a rule of thumb. F1 Score = 2*(Recall * Precision) / (Recall + Precision) F1 Score = 2*(Recall * Precision) / (Recall + Precision) F1 Score = 2*(Recall * Precision) F1 Score

```
Accuracy Score: 91.30697541270396
Confusion Matrix: [[ 4678  3252]
 [ 2214 52734]]
              precision    recall  f1-score   support

           0       0.68      0.59      0.63      7930
           1       0.94      0.96      0.95     54948

    accuracy                           0.91     62878
   macro avg       0.81      0.77      0.79     62878
weighted avg       0.91      0.91      0.91     62878


Cross validation score : 91.36851022184459
Accuracy_Score - Cross Validation Score : -0.0615348091406247
```

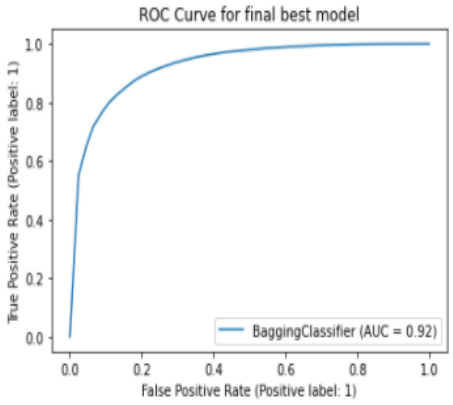
4. AUC ROC Curve:

The AUC - ROC (Receiver Operating Characteristics) curve is a performance evaluation for classification issues at different threshold values. AUC represents the degree or measure of separability, whereas ROC is a probability curve. It indicates how well the model can distinguish between classes. The AUC indicates how well the model predicts 0s as 0s and 1s as 1s. By analogy, the higher the AUC, the better the model distinguishes between people who have the condition and those who do not.

The ROC curve is plotted with TPR on the y-axis and FPR on the x-axis, with TPR on the y-axis and FPR on the x-axis.

The area under the Receiver Operating Characteristic Curve (ROC AUC) calculated from prediction scores is called score.

```
In [86]: #Ploting ROC curve for final best model
         plot_roc_curve(Final_mod, X_test, y_test)
         plt.title('ROC Curve for final best model')
         plt.show()
```



Once hyperparameter tuning is done we got improvement in the AUC ROC Curve as well.

5. Hyperparameter Tuning:

There is a list of several machine learning models available. They're all distinct in some way, yet the only thing that distinguishes them is the model's input parameters. Hyperparameters are the name given to these input parameters. These hyperparameters will establish the model's architecture, and the greatest thing is that you get to choose the ones you want for your model. Because the list of hyperparameters for each model differs, you must choose from a distinct list for each model.

We are unaware of the optimal hyperparameter settings that would produce the best model output. So we tell the model to automatically explore and select the best model architecture. Hyperparameter tuning is the term for the procedure of selecting hyperparameters. GridSearchCV can be used to tune the system.

GridSearchCV is a model selection function included in the Scikit-learn (or SK-learn) package. One thing to keep in mind is that the Scikit-learn library must be installed on the PC. This function aids in fitting your estimator (model) to your training set by looping

through predefined hyperparameters. Finally, we can choose the best parameters from the hyperparameters presented.

Hyper Parameter Tuning:

```
In [80]: #importing necessary libraries
         from sklearn.model_selection import GridSearchCV

In [81]: parameter = {'bootstrap':['True','False'],
                      'n_jobs': [-2,-1,1,2],
                      'n_estimators':[10,20,30,40],
                      'warm_start':['True','False']}

In [82]: GCV=GridSearchCV(BaggingClassifier(),parameter,cv=5)

In [83]: GCV.fit(X_train,y_train)

Out[83]: GridSearchCV(cv=5, estimator=BaggingClassifier(),
                      param_grid={'bootstrap': ['True', 'False'],
                                  'n_estimators': [10, 20, 30, 40],
                                  'n_jobs': [-2, -1, 1, 2],
                                  'warm_start': ['True', 'False']})

In [84]: GCV.best_params_

Out[84]: {'bootstrap': 'False', 'n_estimators': 40, 'n_jobs': 2, 'warm_start': 'True'}

In [85]: Final_mod=BaggingClassifier(bootstrap='True', n_jobs=-1,warm_start='True', n_estimators=40)
         Final_mod.fit(X_train,y_train)
         pred=Final_mod.predict(X_test)
         acc=accuracy_score(y_test, pred)

         print('Accuracy Score:',(accuracy_score(y_test,pred)*100))
         print('Confusion matrix:',confusion_matrix(y_test,pred))
         print(classification_report(y_test,pred))

Accuracy Score: 92.0083335983969
Confusion matrix: [[ 4402  3528]
 [ 1497 53451]]
              precision    recall  f1-score   support

           0       0.75      0.56      0.64      7930
           1       0.94      0.97      0.96     54948

    accuracy                           0.92     62878
   macro avg       0.84      0.76      0.80     62878
weighted avg       0.91      0.92      0.91     62878
```

The accuracy of the model has been increased from 91.36% to 92%.

- **<u>Visualizations:</u>**

  Now we'll look at the various plots created with this dataset in order to gain a better understanding of the data. The plots' codes, as well as the results, are listed below:

Univariate Analysis:

In [41]:
```
# checking for categorical columns
categorical_columns=[]
for i in df.dtypes.index:
    if df.dtypes[i]=='object':
        categorical_columns.append(i)
print(categorical_columns)
```

```
[]
```

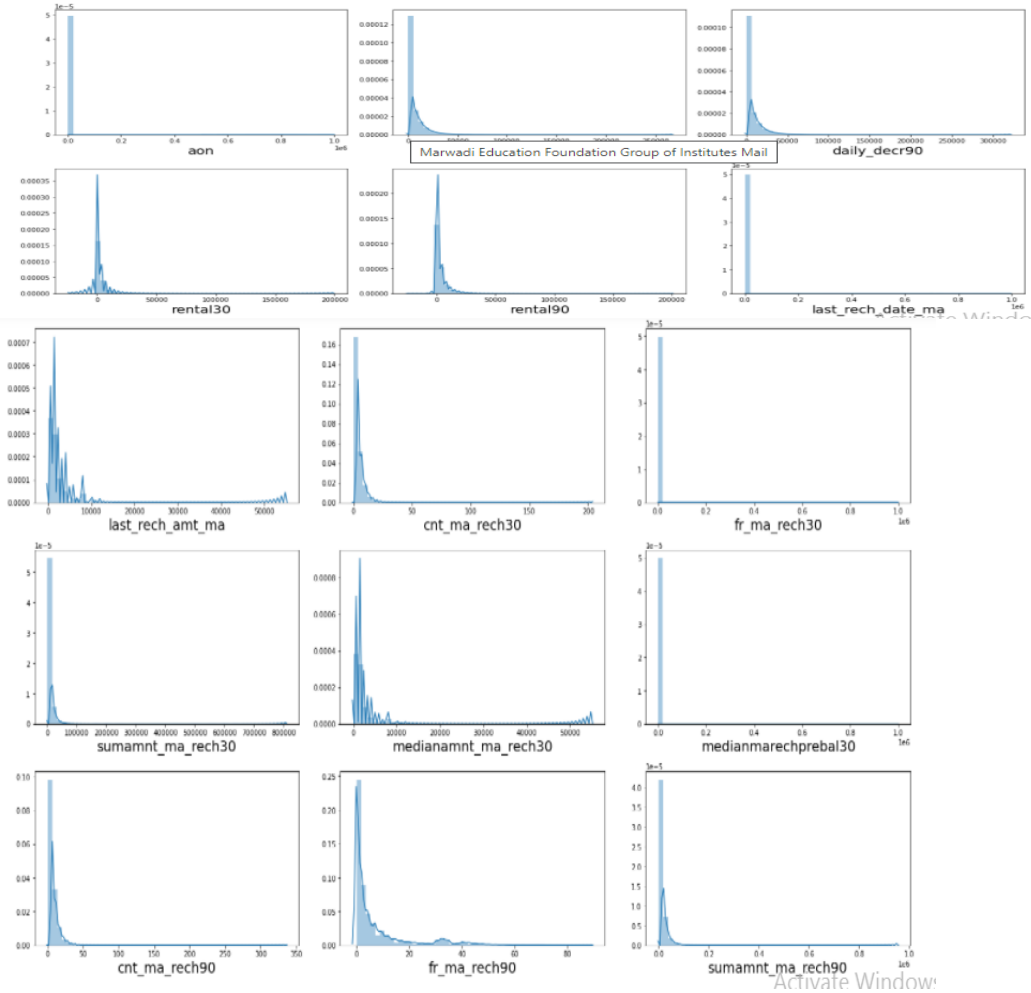
As observed there is no categorical data in the dataset.

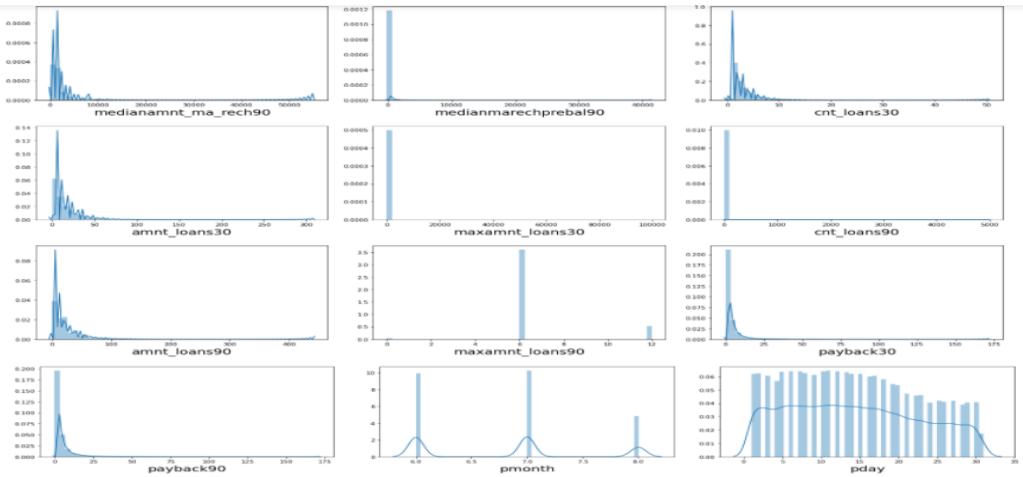
In [42]:
```
# Now checking for numerical columns
numerical_columns=[]
for i in df.dtypes.index:
    if df.dtypes[i]!='object':
        numerical_columns.append(i)
print(numerical_columns)
```

```
['label', 'aon', 'daily_decr30', 'daily_decr90', 'rental30', 'rental90', 'last_rech_date_ma', '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_loans30', 'amnt_loans30', 'maxamnt_loans30', 'cnt_loans90', 'amnt_loans90', 'maxamnt_loans90', 'payback30', 'payback90', 'pmonth', 'pday']
```

These are the columns which have numerical data.

In [44]:
```
#Distribution plot for all numerical columns except label
plt.figure(figsize = (20,40))
plotnumber = 1
for column in df[col]:
    if plotnumber <=30:
        ax = plt.subplot(10,3,plotnumber)
        sns.distplot(df[column])
        plt.xlabel(column,fontsize = 20)
    plotnumber+=1
plt.tight_layout()
```

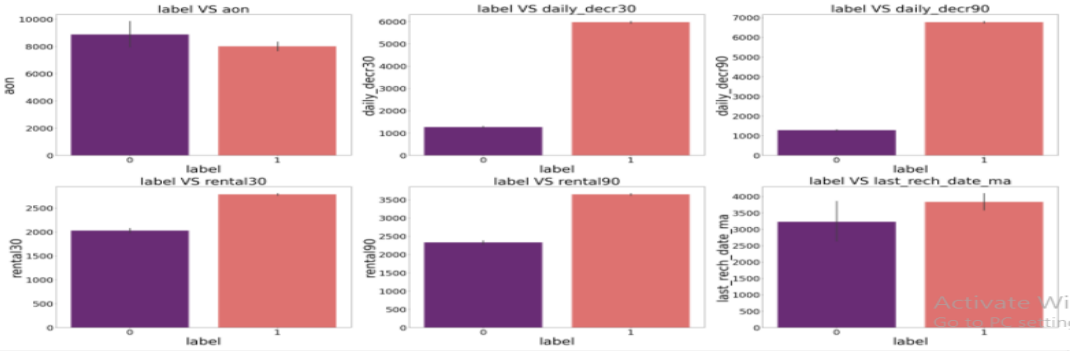


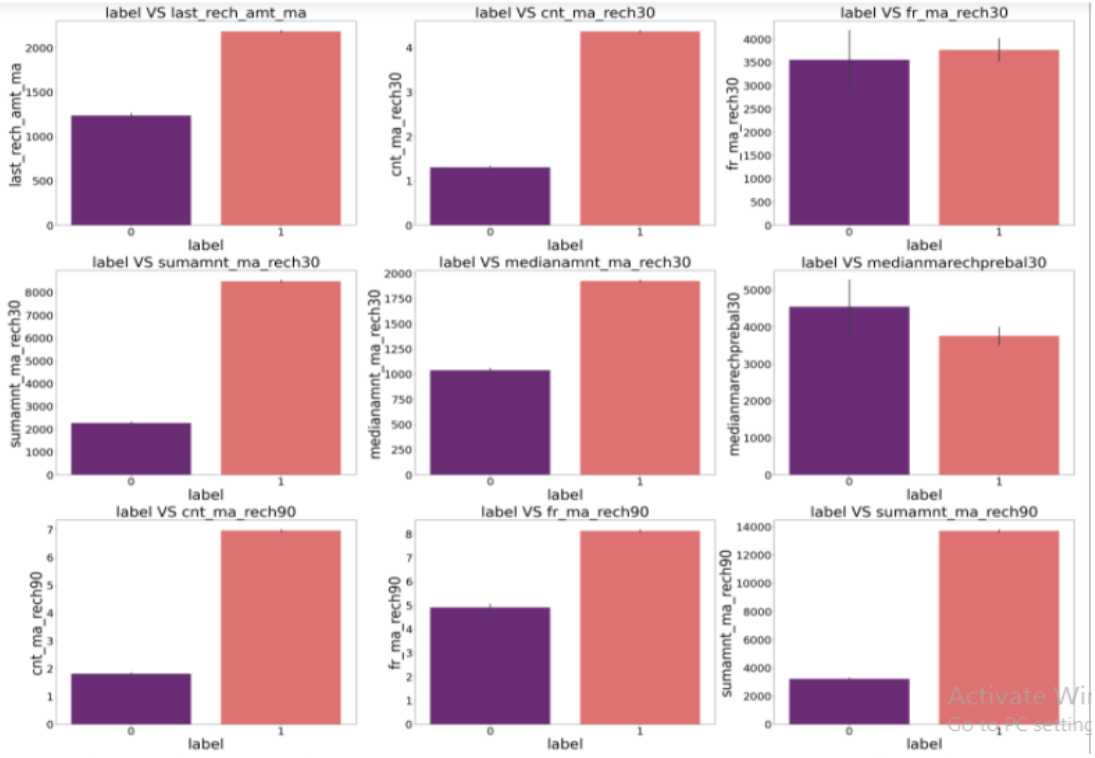


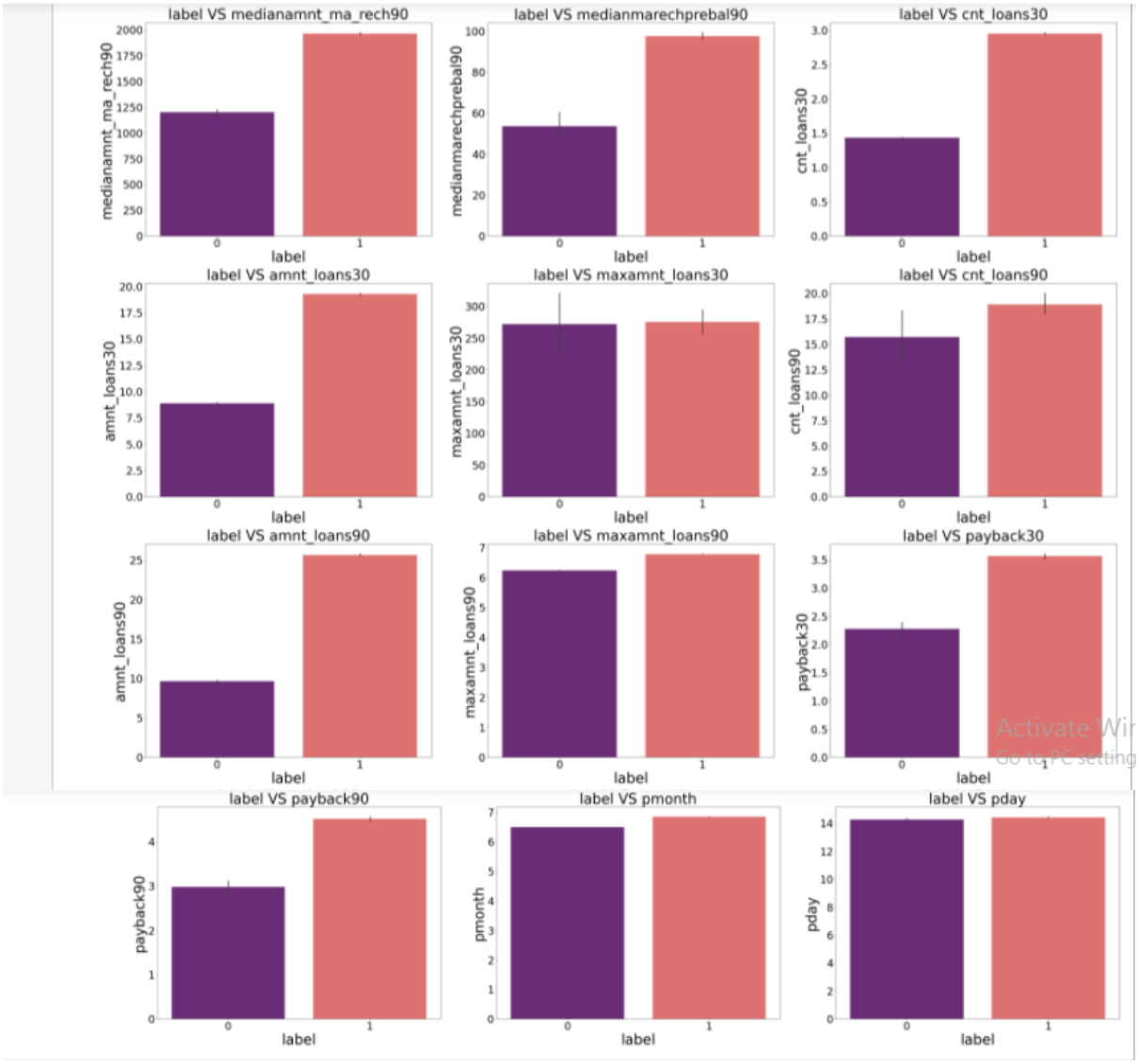
As we can see there is skewness in most of the data so we have to treat them.

Bivariate Analysis:

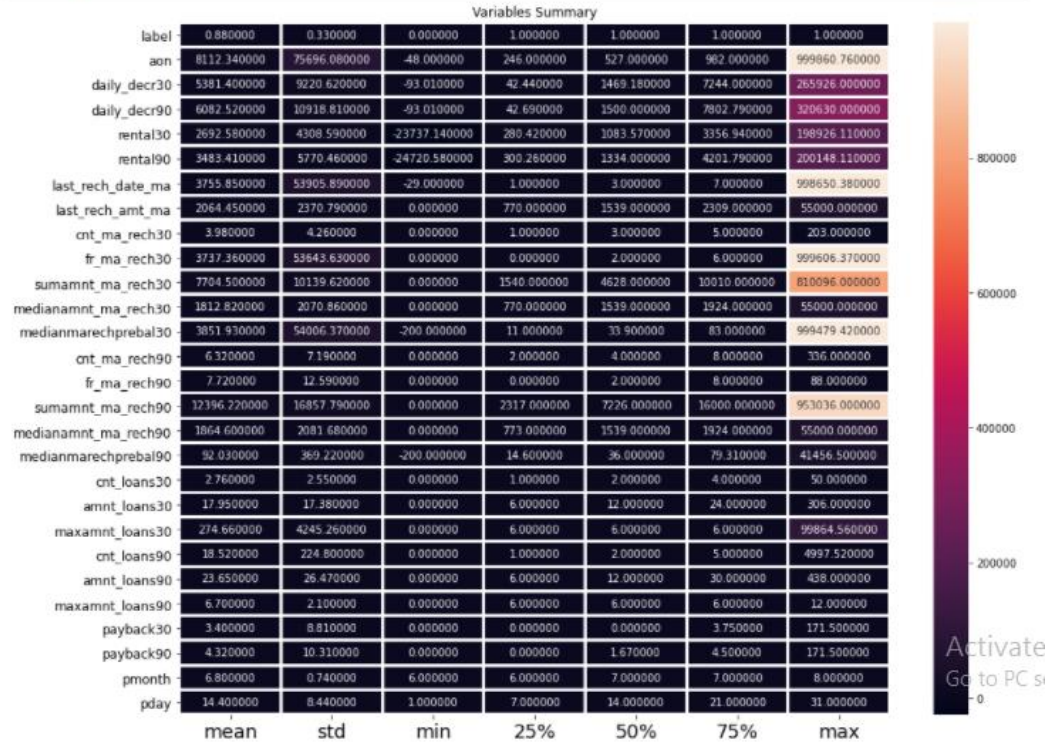
In [45]:
```
#barplot for numerical columns
plt.figure(figsize=(40,100))
for i in range(len(col)):
    plt.subplot(10,3,i+1)
    sns.barplot(x=df['label'], y=df[col[i]], palette="magma")
    plt.title(f"label VS {col[i]}",fontsize=40)
    plt.xticks(fontsize=30)
    plt.yticks(fontsize=30)
    plt.xlabel('label',fontsize = 40)
    plt.ylabel(col[i],fontsize = 40)
    plt.tight_layout()
```







In [37]:
```
plt.figure(figsize=(15,12))
sns.heatmap(round(df.describe()[1:].transpose(),2),linewidth=2,annot=True,fmt="f")
plt.xticks(fontsize=18)
plt.yticks(fontsize=12)
plt.title("Variables Summary")
plt.show()
```



- <u>Interpretation of the Results:</u>

Output:

In [95]:
```
import numpy as np
a=np.array(y_test)
predicted=np.array(predbc,y_test)
df_con=pd.DataFrame({"Original":a,"Predicted":predicted},index=range(len(a)))
df_con
```

Out[95]:

	Original	Predicted
0	1	1
1	1	1
2	1	1
3	1	1
4	1	1
...	...	...
62873	1	1
62874	1	1
62875	1	1
62876	1	1
62877	1	1

62878 rows × 2 columns

CONCLUSION

- Key Findings and Conclusions of the Study:

  Based on the numerous criteria taken into account, an MFI can determine whether or not a person will return money and whether or not an MFI should issue a loan to that individual.

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

  For greater accuracy, I developed many classification models rather than relying on a single model, and I used cross validation comparison to guarantee that the model did not suffer from overfitting or underfitting. To improve the scores, I chose the best one and did hyper parameter tuning on it.

- Limitations of this work and Scope for Future Work:

  The limitation is that it will only function for this specific use case, and it will need to be tweaked if used in a new scenario on a similar scale. The scope of this dataset is that we can use it in companies to determine whether we should provide a loan to a person or not, and we can also make predictions about a person purchasing an expensive service based on their personal details that we have in this dataset, such as the number of times their data account has been recharged in the last 30 days and the daily amount spent from their main account, averaged over the last 30 days (in Indonesian Rupiah), so even a marketing firm can use it.