MICRO CREDIT DEFAULTER PROJECT

Submitted By

Girija Chandra Mohan

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my SME (Subject Matter Expert) Khushboo Garg as well as Flip Robo Technologies who gave me the opportunity to do this project on Micro Credit Loan Defaulter Project, which also helped me in doing lots of research wherein I came to know about so 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

  A Microfinance Institution (MFI) is an organization that offers financial services to low-income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on. Many microfinance institutions (MFI), experts and donors are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient, and cost saving, than the traditional high-touch model used since long for the purpose of delivering microfinance services. Though, the MFI industry is primarily focusing on low-income families and are very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes.

- Conceptual Background of the Domain Problem

  As we are aware of the targeted customer of our client is the population with low income, we can say almost all of the customers would be mostly taking loan for calling and not for internet services. So, concentrating on the main balance should be the key here.

- Review of Literature

  Electronical Communication is the need of the day. For shopping for bread to hollering the Emergency Services there’s no second to communication.

The Companies belonging from the telecom industries are well aware of it, and they want to approach the customers furthermore than just providing telecom services. The Companies wants to lend a small amount of credit to those customers those who have their services expired and need to use the service right away without paying the complete bill at the point of time. This loan is to be paid back within a period of 5 days with a particular interest. If the customer does not pay the amount with interest within the given period of time, the consumer believed to be defaulter.

We have to build a model which can be used to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of loan.

- Motivation for the Problem Undertaken

Our main objective of doing this project is to build a model to predict whether the users are paying the loan within the due date or not. We are going to predict by using Machine Learning algorithms.

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

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  There are various analytics which I have done before moving forward with exploratory analysis, on the basis of accounts which got recharged in the last 30 days. I set the parameter that if the person is not recharging their main account within 3 months, I simply dropped their data because they are not valuable and they might be old customers, but there is no revenue rotating. Then I had checked the date columns and found that the data belongs to the year 2016. I extracted the month and day from the date, saved the data in separate columns, and tried to visualize the data on the basis of months and days.

  I had checked the maximum amount of loan taken by the people and found that the data had more outliers. As per the description given by the client, the loan amount can be paid by the customer is either rupiah 6 or 12 so that I have dropped all the loan amount that shows the loan is taken more than 12 rupiah.

  Then I separated the defaulter's data and checked the valuable customers in the network and we found that their monthly revenue is more than 10000 rupiah. Although the data is quite imbalanced and many columns doesn't have that expected maximum value, we dropped that columns. We checked the skewed data and try to treat the skewed data before model processing which caused NaN so avoided it.

When we try removing the unwanted data, i.e., the outliers, we found that almost 40000+ data has been chopped. Though the data given by the client had almost 37 columns and over 2 lakh columns I did not feel like losing on precious data so avoided the outlier removal part as well. After scaling my data, I have sent the data to various classification models and found that Random forest Classifier Algorithm is working well.

- Data Sources and their formats

  The data was provided by the client to "FlipRobo Technologies". The data is in the form of a comma separated file (CSV). The data i.e. the features and the target are in the single file.

```
df= pd.read_excel(r'C:\python\MCD.xlsx')
df.head()
```

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

Here we are taking a look at the first 5 rows of our dataset. It shows that we have a total of 209593 rows and 37 columns present in our dataframe. We have the label column that stores the defaulter and non-defaulter values marked with 0 and 1 making this a Classification problem!

- Data Preprocessing

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

```
df.isnull().sum()
```

```
label                   0
msisdn                  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
```

  Using the info method, we are able to confirm the non-null count details as well as the datatype information. We have 21 float/decimal datatype, 12 integer datatype and 3 object/categorical datatype columns. We will need to convert the object datatype columns to numerical data before we input the information in our machine learning models.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 209592 entries, 0 to 209592
Data columns (total 36 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   label                 209592 non-null  int64
 1   msisdn                209592 non-null  object
 2   aon                   209592 non-null  float64
 3   daily_decr30          209592 non-null  float64
 4   daily_decr90          209592 non-null  float64
 5   rental30              209592 non-null  float64
 6   rental90              209592 non-null  float64
 7   last_rech_date_ma     209592 non-null  float64
 8   last_rech_date_da     209592 non-null  float64
 9   last_rech_amt_ma      209592 non-null  int64
 10  cnt_ma_rech30         209592 non-null  int64
 11  fr_ma_rech30          209592 non-null  float64
 12  sumamnt_ma_rech30     209592 non-null  int64
 13  medianamnt_ma_rech30  209592 non-null  float64
 14  medianmarechprebal30  209592 non-null  float64
 15  cnt_ma_rech90         209592 non-null  int64
```

- 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 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 : 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

pcircle : Telecom circle

pdate : Date

Assumptions (if any) related to the problem under consideration

I had made an assumption that any telecom company keeps the data of customer within 3 months so I have chopped off my data on basis of that.

I have dropped the 2016 year from pdate columns because the data is from the year 2016, only the date and months are different. We separated months and days to different columns.

Then I separately checked the defaulter's data and found that many valuable users are defaulters as they might have forgotten to pay or they are having a busy life. I separated them so that company can deal politely, because we cannot lose these customers.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Describe the approaches you followed, both statistical and analytical, for solving of this problem.

- Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing.

The Algorithms used for testing, training and Validating the models are as follows:

- Logistic Regression
- Decision Tree Classifier
- K Nearest Neighbour
- GuassianNB
- Random Forest Classifier

Run and Evaluate selected models

- We have used a total of 5 Classification Models after choosing the random state amongst 1-100 number. Then it is checked for best fold for CV using the following codes.

LogisticRegression

```
maxacc=0
maxRS=0

for i in range(1,100):
    xtrain,xtest,ytrain,ytest= train_test_split(x1,y1,test_size=20,random_state=i)
    lg=LogisticRegression()
    lg.fit(xtrain,ytrain)
    predlg= lg.predict(xtest)
    acc=accuracy_score(ytest,predlg)
    if acc>maxacc:
        maxacc=acc
        maxRS=i
print("Best Accuracy_score is", maxacc, 'on Random_state',maxRS)
```

```
Best Accuracy_score is 0.95 on Random_state 33
```

```
pred_tr= lg.predict(xtrain)
pred_ts= lg.predict(xtest)
```

```
from sklearn.model_selection import cross_val_score
```

```
train_accuracy= accuracy_score(ytrain,pred_tr)
test_accuracy= accuracy_score(ytest,pred_ts)
```

```
for j in range(2,10):
    cv_score= cross_val_score(lg,x1,y1,cv=j)
    cv_mean=cv_score.mean()
    print(f"At cross fold {j} the cv score is {cv_mean} and accuracy score for training is {train_accuracy} and the accuracy for
    print("\n")
```

```
At cross fold 2 the cv score is 0.7354876519653273 and accuracy score for training is 0.7356204339766655 and the accuracy for t
esting is 0.85


At cross fold 3 the cv score is 0.7349452154436099 and accuracy score for training is 0.7356204339766655 and the accuracy for t
esting is 0.85
```

```
xtrain,xtest,ytrain,ytest= train_test_split(x,y,test_size=20,random_state=33)
```

Logistic Regression

```
#Logistic Regression
lg=LogisticRegression()
lg.fit(xtrain,ytrain)
predlg= lg.predict(xtest)
print( confusion_matrix(predlg,ytest))
print( classification_report(predlg,ytest))
print('Score:', lg.score(xtrain,ytrain))
print('Acc_score:', accuracy_score(predlg,ytest))
```

```
[[ 0  1]
 [ 1 18]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.95      0.95      0.95        19

    accuracy                           0.90        20
   macro avg       0.47      0.47      0.47        20
weighted avg       0.90      0.90      0.90        20

Score: 0.8768442349168782
Acc_score: 0.9
```

```
cv_score=cross_val_score(lg,x,y, cv=2)
print('Cross Validation Score of LogisticRegression is:', cv_score.mean())
```

```
Cross Validation Score of LogisticRegression is: 0.8767891904271156
```

GuassianNB

```
#GaussianNB
gnb=GaussianNB()
gnb.fit(xtrain,ytrain)
predgnb= gnb.predict(xtest)
print( confusion_matrix(predgnb,ytest))
print( classification_report(predgnb,ytest))
print('Acc_score:', accuracy_score(predgnb,ytest))
```

```
[[ 0 11]
 [ 1  8]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        11
           1       0.42      0.89      0.57         9

    accuracy                           0.40        20
   macro avg       0.21      0.44      0.29        20
weighted avg       0.19      0.40      0.26        20

Acc_score: 0.4
```

```
cv_score=cross_val_score(gnb,x,y, cv=2)
print('Cross Validation Score of GaussianNB is:', cv_score.mean())
```

```
Cross Validation Score of GaussianNB is: 0.5891446238406046
```

Decision Tree Classifier

```
#DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(xtrain,ytrain)
preddtc= dtc.predict(xtest)
print( confusion_matrix(preddtc,ytest))
print( classification_report(preddtc,ytest))
print('Acc_score:', accuracy_score(preddtc,ytest))
```

```
[[ 0  2]
 [ 1 17]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.89      0.94      0.92        18

    accuracy                           0.85        20
   macro avg       0.45      0.47      0.46        20
weighted avg       0.81      0.85      0.83        20

Acc_score: 0.85
```

```
cv_score=cross_val_score(dtc,x,y, cv=2)
print('Cross Validation Score of DecisionTreeClassifier is:', cv_score.mean())
```

```
Cross Validation Score of DecisionTreeClassifier is: 0.8840270621016069
```

Random Forest Classifier

```
#RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(xtrain,ytrain)
predrf= rf.predict(xtest)
print( confusion_matrix(predrf,ytest))
print( classification_report(predrf,ytest))
print('Acc_score:', accuracy_score(predrf,ytest))
```

```
[[ 0  0]
 [ 1 19]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.95      0.97        20

    accuracy                           0.95        20
   macro avg       0.50      0.47      0.49        20
weighted avg       1.00      0.95      0.97        20

Acc_score: 0.95
```

```
cv_score=cross_val_score(rf,x,y, cv=2)
print('Cross Validation Score of RandomForestClassifier is:', cv_score.mean())
```

```
Cross Validation Score of RandomForestClassifier is: 0.920836673155464
```

KNN

```
#knn
knn=KNeighborsClassifier()
knn.fit(xtrain,ytrain)
predknn= knn.predict(xtest)
print( confusion_matrix(predknn,ytest))
print( classification_report(predknn,ytest))
print('Acc_score:', accuracy_score(predknn,ytest))
```

```
[[ 0  0]
 [ 1 19]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.95      0.97        20

    accuracy                           0.95        20
   macro avg       0.50      0.47      0.49        20
weighted avg       1.00      0.95      0.97        20

Acc_score: 0.95
```

```
cv_score=cross_val_score(knn,x,y, cv=2)
print('Cross Validation Score of KNN is:', cv_score.mean())
```

```
Cross Validation Score of KNN is: 0.8870710714149395
```

Key Metrics for success in solving problem under consideration

The key metrics used here were accuracy_score, cross_val_score, classification report, and confusion matrix. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning and we will be using GridSearchCV method.

1. **Cross Validation**:

   Cross-validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross-

validation the 1st part (20%) of the 5 parts will be kept out as a holdout set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset. In the similar way further iterations are made for the second 20% of the dataset is held as a holdout set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality.

2. **Confusion Matrix**:

   A confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e., commonly mislabelling one as another). It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

3. **Classification Report**:

   The classification report visualizer displays the precision, recall, F1, and support scores for the model. There are four ways to check if the predictions are right or wrong:

1. TN / True Negative: the case was negative and predicted negative

2. TP / True Positive: the case was positive and predicted positive
3. FN / False Negative: the case was positive but predicted negative
4. FP / False Positive: the case was negative but predicted positive

Precision:

Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive. It is the accuracy of positive predictions. The formula of precision is given below:

$$\text{Precision} = TP/ (TP + FP)$$

Recall:

Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. It is also the fraction of positives that were correctly identified. The formula of recall is given below:

$$\text{Recall} = TP/(TP+FN)$$

F1 score:

The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy. The formula is:

$$\text{F1 Score} = 2*(\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

Support:

Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process.

4. **Hyperparameter Tuning:**

There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as Hyperparameters. These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.

We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as Hyperparameter Tuning. We can do tuning by using GridSearchCV.

GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

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

```
RandomForestClassifier()
```

```
RandomForestClassifier()
```

```
parameters={'max_features': ["auto", "sqrt", "log2"],
            'min_samples_leaf': [1,2,3],
            'criterion':["gini","entrophy"],
            'max_depth':[3,4,5,6],
            'min_samples_split': [2,3,4,5]}
```

```
GCV=GridSearchCV(RandomForestClassifier(), parameters, cv=2, scoring='accuracy')
GCV.fit(xtrain,ytrain)
GCV.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 6,
 'max_features': 'sqrt',
 'min_samples_leaf': 2,
 'min_samples_split': 2}
```

```
GCV.best_estimator_
```

```
RandomForestClassifier(max_depth=6, max_features='sqrt', min_samples_leaf=2)
```

```
GCV_pred=GCV.best_estimator_.predict(xtest)
accuracy_score(ytest,GCV_pred)
```

```
0.95
```

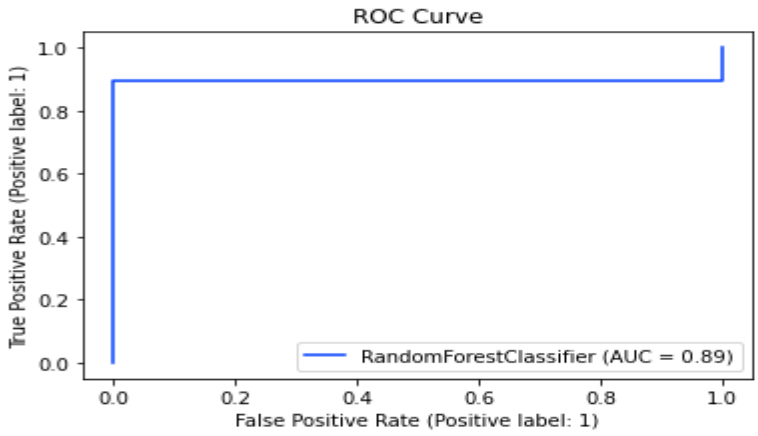
```
fnrf=RandomForestClassifier(max_depth=6, max_features='sqrt', min_samples_leaf=2)
fnrf.fit(xtrain,ytrain)
predfnrf= fnrf.predict(xtest)
print( confusion_matrix(predfnrf,ytest))
print( classification_report(predfnrf,ytest))
print('Acc_score:', accuracy_score(predfnrf,ytest))
```

```
[[ 0  0]
 [ 1 19]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.95      0.97        20

    accuracy                           0.95        20
   macro avg       0.50      0.47      0.49        20
weighted avg       1.00      0.95      0.97        20

Acc_score: 0.95
```

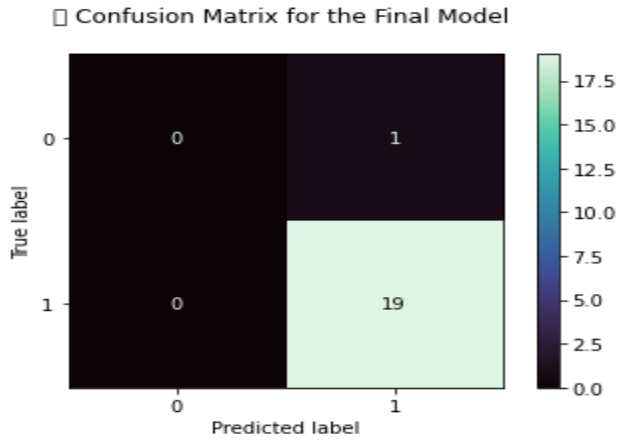
After applying Hyperparameter Tuning, we can see that RandomForestClassifier Algorithm is performing well as the scores are improved Now, we will finalize RandomForestClassifier algorithm model as the final model.

Final Model

```
from sklearn.metrics import plot_roc_curve
plot_roc_curve(rf,xtest,ytest)
plt.title('ROC Curve')
plt.show()
```




```
metrics.plot_confusion_matrix(fnrf, xtest, ytest, cmap='mako')
plt.title('\t Confusion Matrix for the Final Model \n')
plt.show()
```



```
import pickle
filename= 'MCDP.pkl'
pickle.dump(fnrf,open(filename, 'wb'))
```

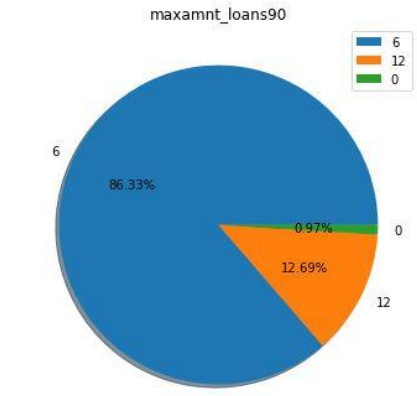
Compare Original results with the Predicted Results

```
x=np.array(ytest)
predicted= np.array(fnrf.predict(xtest))
df_con= pd.DataFrame({'original': x, 'Predicted': predicted}, index= range(len(x)))
df_con
```

	original	Predicted
0	1	1
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1
6	1	1
7	1	1
8	1	1

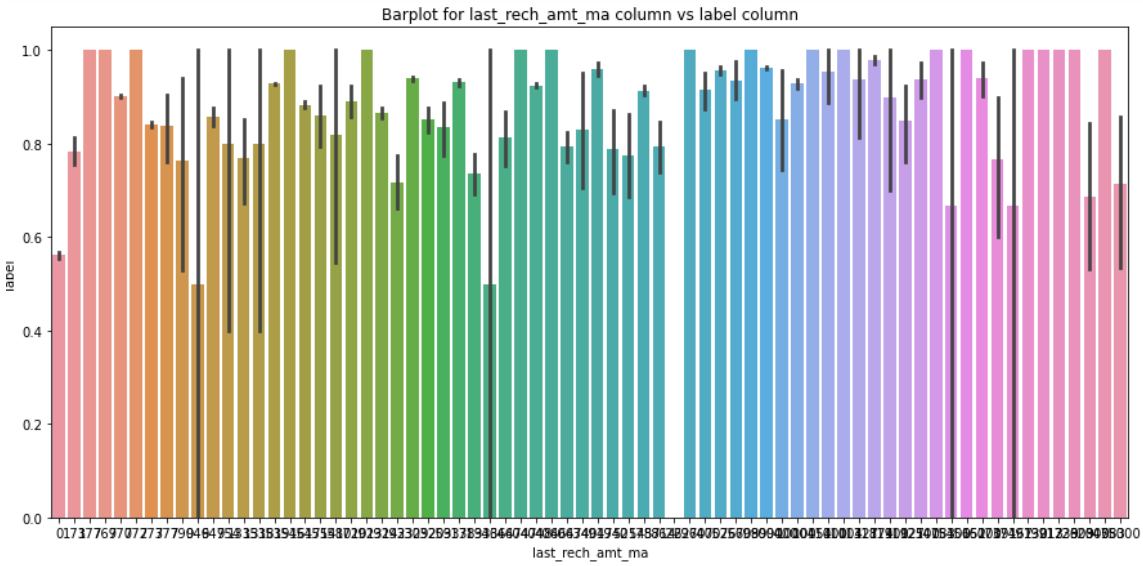
DATA VISUALIZATION

```
plt.figure(figsize=(10,5))
plt.pie(df['maxamnt_loans90'].value_counts(), labels=df['maxamnt_loans90'].value_counts().index, shadow= True, autopct='%1.2f%%')
plt.legend(prop={'size':10})
plt.title('maxamnt_loans90')
plt.tight_layout()
plt.show()
```



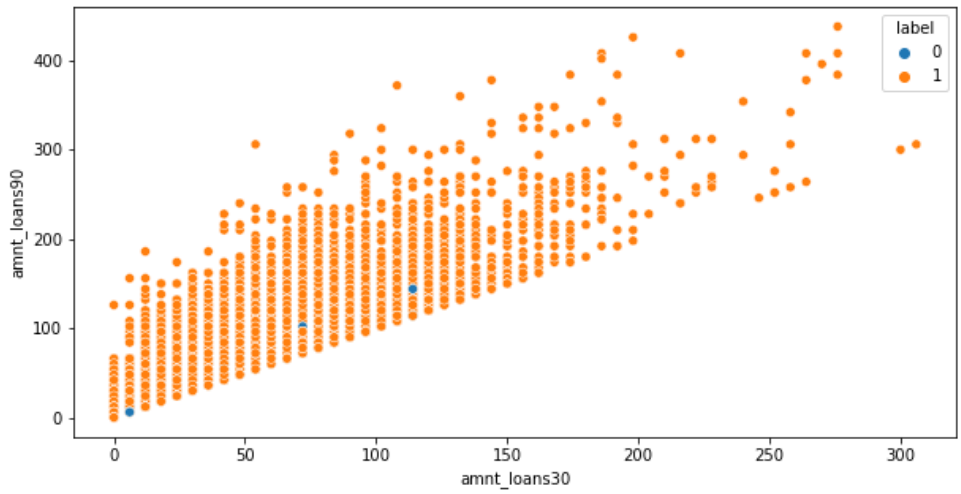
On-Screen Keyboard

```
y = 'label'

x = 'aon'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df)
plt.title(f"Barplot for {x} column vs {y} column")
plt.show()

x = 'last_rech_date_da'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df)
plt.title(f"Barplot for {x} column vs {y} column")
plt.show()

x = 'last_rech_date_ma'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df)
plt.title(f"Barplot for {x} column vs {y} column")
plt.show()

x = 'last_rech_amt_ma'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df)
plt.title(f"Barplot for {x} column vs {y} column")
plt.show()
```

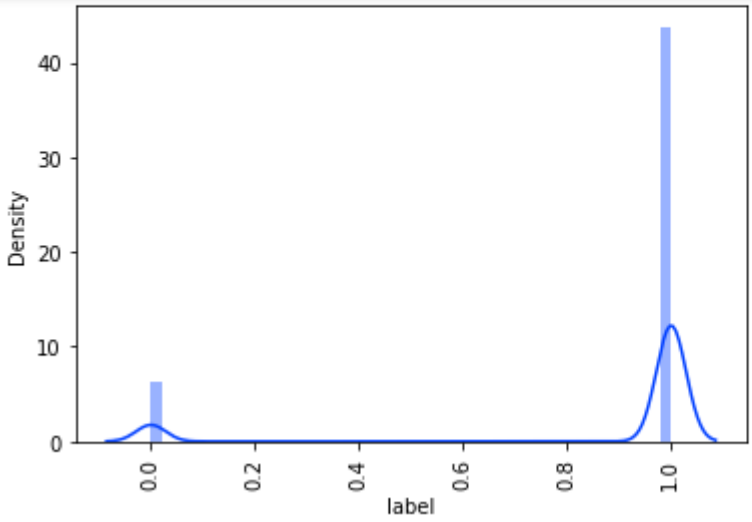


```
plt.figure(figsize=(10,5))
sns.scatterplot(x='amnt_loans30', y='amnt_loans90', data=df, hue='label')
```

```
<AxesSubplot:xlabel='amnt_loans30', ylabel='amnt_loans90'>
```

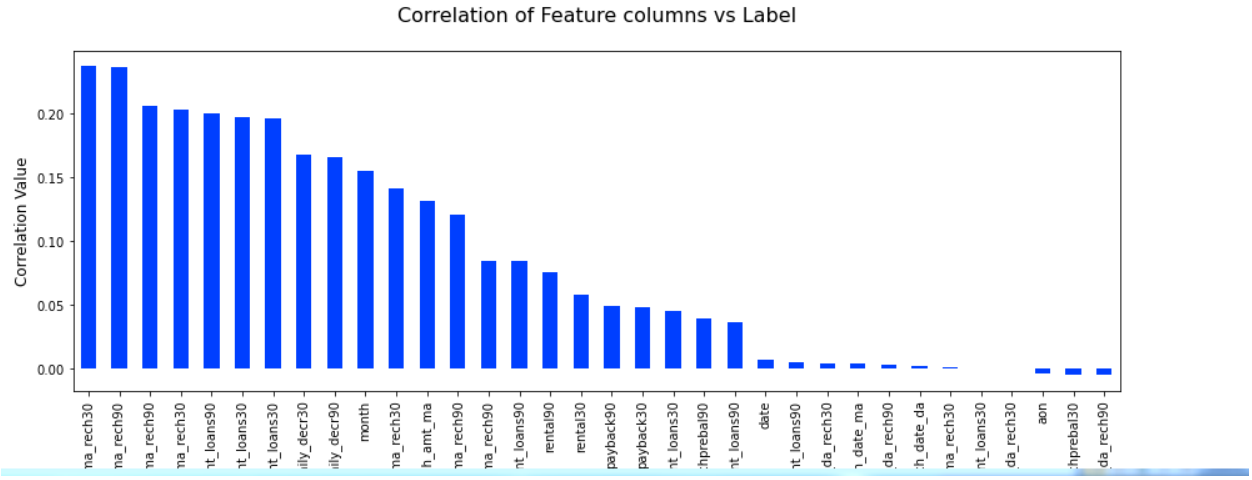



```
for i in df.columns:
    sns.distplot(df[i])
    plt.xticks(rotation=90)
    plt.show()
```



Correlation of columns with the Label

```
df_corr = df.corr()
plt.figure(figsize=(15,5))
df_corr['label'].sort_values(ascending=False).drop('label').plot.bar()
plt.title("Correlation of Feature columns vs Label\n", fontsize=16)
plt.xlabel("\nFeatures List", fontsize=14)
plt.ylabel("Correlation Value", fontsize=12)
plt.show()
```



CONCLUSION

- Key Findings and Conclusions of the Study

  From the final model we can find if a person will return money or not and should an MFI provide a loan to that person or not judging from the various features taken into consideration.

- Learning Outcomes of the Study in respect of Data Science

  We built multiple classification models and did not rely on one single model for getting better accuracy and using cross validation comparison and ensured that the model does not fall into overfitting and underfitting issues. The best one is selected and performed hyper parameter tuning on it to enhance the scores.

- Limitations of this work and Scope for Future Work

  Limitation is it will only work for this particular use case and will need to be modified if tried to be utilized on a different scenario but on a similar scale. Scope is that we can use it in companies to find whether we should provide loan to a person or not and we can also make prediction about a person buying an expensive service on the basis of their personal details that we have in this dataset like number of times data account got recharged in last 30 days and daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah) so even a marketing company can also use this.