FLIP ROBO

NAME OF THE PROJECT

Micro-credit Defaulters

Submitted by:

Kamran Ashraf

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped you and guided you in completion of the project.

- I would like to thank FlipRobo Technologies for providing me this opportunity and guidance throughout the project and all the steps that are implemented.

- I have primarily referred to various articles scattered across various websites for the purpose of getting an idea on "*Micro-credit defaulters*" project.

- I would like to thank the technical support team also for helping me out and reaching out to me on clearing all my doubts as early as possible.

- I would like to thank my project SME M/S Sapna Verma for providing the flexibility in time and also for giving us guidance in creating the project.

- I have referred to various articles in Towards Data Science and Kaggle

INTRODUCTION

- ## Business Problem Framing
- A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on.
- Many microfinance institutions (MFI), experts and donors are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient, and cost saving, than the traditional high-touch model used since long for the purpose of delivering microfinance services. Though, the MFI industry is primarily focusing on low income families and are very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes.
- Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients.

- ## Conceptual Background of the Domain Problem
- We are working with client that is in Telecom Industry .They have launched various products and have developed its business and organization based on the budget operator model, offering better products at Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.
- Telecom Industry understand the importance of communication and how it affects a person’s life, thus, focusing on providing their services and products to low income families and poor customers that can help them in the need of hour.
- They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah). The sample data is provided to us from our client database. It is hereby given to you for this exercise. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

- Review of Literature

An attempt has been made in this report to review the available literature in the area of microfinance. Approaches to microfinance, issues related to measuring social impact versus profitability of MFIs, issue of sustainability, variables impacting sustainability, which affect the regulations of profitability and impact assessment of MFIs have been summarized in the below report. We hope that the below report of literature will provide a platform for further research and help the industry to combine theory and practice to take microfinance forward and contribute to alleviating the poor from poverty
The various applications and methods which inspired us to build our project. We did a background survey regarding the basic ideas of our project and used those ideas for the collection of information like the technological stack, algorithms, and shortcomings of our project which led us to build a better project.
I have built a model which can be used to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan. In this case, Label '1' indicates that the loan has been payed i.e. Non-defaulter, while, Label '0' indicates that the loan has not been payed i.e. defaulter.

- Motivation for the Problem Undertaken

I have to model the micro credit defaulters with the available independent variables. This model will then be used by the management to understand how the customer is considered as defaulter or non-defaulter based on the independent variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand whether the customer will be paying back the loaned amount within 5 days of insurance of loan. The relationship between predicting defaulter and the economy is an important motivating factor for predicting micro credit defaulter model

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

I am working with the micro credit defaulters dataset that contains various features and information about it. Using the data in form of 'read_csv' function provided by the Pandas package, which can import the data into our python environment. After importing the data, I have used the 'head' function to get a glimpse of our dataset.

In this label is used as my target column and it was having two classes Label '1' indicates that the loan has been paid i.e. Non-defaulter, whereas Label '0' indicates that the loan has not been paid i.e. defaulter. It's clarify the binary classification problem, classification of algorithms for building model. There is no null values in the dataset and observed some unnecessary entries in some columns like in some columns it found more than 90% , zero values so dropped those columns. Those columns will create high skewness in the model.

To get better insight on the features uses plotting function like distribution plot, bar plot and count plot. With these plotting it is able to understand the relation between the features in better manner. Also outliers and skewness found in the dataset so it is removed outliers using percentile method and skewness using yeo-johnson method. classification algorithms while building model then tuned the best and saved the best model. Lastly predicted the label using saved model.

- ## Data Sources and their formats

Dataset has been provided by internship company – Flip Robo technologies in excel format. The sample data is provided from our client database. Data given is only for academic use, not for any commercial. 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. Also, dataset was having 209593 rows and 36 columns including target. In this particular datasets I have object, float and integer types of data. The dataset is in both numerical as well as categorical data. There may be some customers with no loan history. The dataset is imbalanced. Label '1' has approximately 87.5% records, while, label '0' has approximately 12.5% records.
Link for Dataset description: Micro-Credit-Defaulter-Project/Data_Description.xlsx at main · DS0003/Micro-Credit-Defaulter-Project (github.com)

- ## Data Pre-processing Done

- In order to get a better understanding of the data, we plotted a histogram of the data. We noticed that the dataset had many outliers, so removing outliers using percentile method and skewness using yeo-johnson method. however, there were many data points that did not conform to this. This is because accident history and condition can have a significant effect of defaulter or non- defaulter, we pruned our dataset to standard deviations around the mean in order to remove outliers. We converted the Make, Model and State into one-hot vectors.

Median and maximum amount of loan taken by the user in last 30 days:

```
df.loc[(df['maxamnt_loans30']!=6.0)&(df['maxamnt_loans30']!=12.0)&(df['maxamnt_loans30']!=0.0),'maxamnt_loans30']
```

```
118       61907.697372
125       22099.413732
146       98745.934048
369       58925.364061
374       78232.464324
              ...
209189    50824.996349
209262    17324.994582
209331    92864.501728
209392    54259.265687
209424    96927.243252
Name: maxamnt_loans30, Length: 1047, dtype: float64
```

- Maximum loans in 30 & 90 days

```
# maximum amount of loan taken by the user in last 30 days:
df['maxamnt_loans30'].value_counts()
```

```
6.0     179193
12.0     26109
0.0       4291
Name: maxamnt_loans30, dtype: int64
```

```
# maximum amount of loan taken by the user in last 90 days
df['maxamnt_loans90'].value_counts()
```

```
6     180945
12     26605
0       2043
Name: maxamnt_loans90, dtype: int64
```

- Data Inputs- Logic- Output Relationships
  - Since all data has numerical columns and plotted dist plot to see the distribution of each column data. So box plot is used for each pair of categorical features that shows the relation between label and independent features. Also we can observe whether the person pays back the loan within the date based on features.
  - In maximum features relation with target I observed Non-defaulter count is high compared to defaulters.

Exploratory Data Analysis (*EDA*)

- This section shows the exploration done on the dataset, which is what motivated the use of the algorithm. The following are the questions explored in this project and for the sake of writing I will only show some of the visuals here while I will provide the codes that shows the full visualization of all the questions explored.
- Is there a significant relationship between Non-defaulter & defaulter? It was used to check for this and we can see that there is a relationship between Label '1' as Non-defaulter, whereas Label '0' indicates that the loan has not been paid i.e. defaulter.
- Dataset is imbalanced. Label '1' has approximately 87.5% records, while, label '0' has approximately 12.5% records. Need to balance.
- There are two primary phases in the system:
  1. Training phase: The system is trained by using the data in the data set and fits a model (line/curve) based on the algorithm chosen accordingly.
  2. Testing phase: the system is provided with the inputs and is tested for its working. The accuracy is checked. And therefore, the data that is used to train the model or test it, has to be appropriate. The system is designed to detect and

predict and hence appropriate algorithms must be used to do the two different tasks. Before the algorithms are selected for further use, different algorithms were compared for its accuracy. The well-suited one for the task was chosen.

Data cleaning:

Remove outliers

```
# feature contain outliers
featr=df[['aon', 'daily_decr30', 'daily_decr90', 'rental30', 'rental90', 'last_rech_date_ma', 'last_rech_amt_ma', 'cnt_ma_rech30
```

Steps :

- Importing the required packages into our python environment
- Importing the data to do some EDA on it
- Dataset having 209593 rows and 36 columns including target.
- Data Visualization
- Feature Selection & Data Split
- Modelling the data using the algorithms
- Evaluating the built model using the evaluation metrics

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

- Finally, we conclude which model is best suitable for the given case by evaluating each of them using the evaluation metrics provided by the scikit-learn package. This model will be a good way for the management to understand whether the customer will be paying back the loaned amount within 5 days of insurance of loan. The relationship between predicting defaulter and the economy is an important motivating factor for predicting micro credit defaulter
Technological stack, algorithms, and shortcomings of the project which led to build this project.

• Hardware and Software Requirements and Tools Used

- Listing down the hardware and software requirements along with the tools, libraries and packages used.
- Windows 10 64bit
- Anaconda 2021 / Python version – Python 3.9.5 (latest)
- Software: Jupyter notebook, Python, Panda library, numpy library, Matplotlib library, Seaborn library

- **Python**: Python is a general-purpose, and high-level programming language which is best known for its efficiency and powerful functions. Its ease to use, which makes it more accessible. Python provides data scientists with an extensive amount of tools and packages to build machine learning models. One of its special features is that we can build various machine learning with less-code.
- **Matplotlib** is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
- **Seaborn** is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data.
- **NumPy** is a general-purpose array-processing package. it provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.
- **Scikit-learn** provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built
- **Jupyter** notebook: The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It includes data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning.

  The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. It includes data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

- The factors need to be found which can impact the micro credit. This can be done by analysing the various factors and the stores the respondent prefers. This will be done by checking each of the factors impacts the respondents in decision making.
- Machine Learning Algorithms:

  Machine learning-based systems are growing in popularity in research applications in most disciplines. Considerable decision-making knowledge from data has been acquired in the broad area of machine learning, in which decision-making tree-based ensemble techniques are recognized for supervised classification problems. Thus, classification is an essential form of data analysis in data mining that formulates models while describing significant data.

- ## Testing of Identified Approaches (Algorithms)

- We utilized several classic and state-of-the-art methods, including ensemble learning techniques, with a 90% - 10% split for the training and test data. To reduce the time required for training, we used over 20 thousand examples in our dataset with 209593 rows and 36 columns.

Z-score

```
from scipy.stats import zscore

z_score = zscore(df[['label']])
abs_zscore = np.abs(z_score)

filtering_entry = (abs_zscore < 3).all(axis=1)

df = df[filtering_entry]


# the data now seems much better than before.

df.describe()
```

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_amt_ma
count	206334.000000	206334.000000	206334.000000	206334.000000	206334.000000	206334.000000	206334.000000	206334.000000
mean	0.873206	8083.775238	5336.954688	6021.166259	2664.367446	3431.671363	3747.842543	2051.860580
std	0.332743	75547.449680	9212.622214	10903.651687	4262.266027	5690.281853	53835.726141	2359.664033
min	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	0.000000
25%	1.000000	246.000000	41.300000	41.484667	276.130000	299.700000	1.000000	770.000000
50%	1.000000	526.000000	1389.035667	1407.000000	1070.795000	1308.900000	3.000000	1539.000000
75%	1.000000	981.000000	7170.000000	7671.877500	3314.222500	4136.532500	7.000000	2309.000000
max	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	55000.000000

Skewness:

```
# Remove skewness
fetr=['aon', 'daily_decr30', 'daily_decr90', 'rental30', 'rental90', 'last_rech_date_ma', 'last_rech_amt_ma', 'cnt_ma_rech30', '
```

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

df[fetr] = pt.fit_transform(df[fetr].values)
```

```
df[fetr].skew()
```

```
aon                  1.681185
daily_decr30        -6.827117
daily_decr90        -7.346018
rental30            -1.002671
rental90            -0.982559
last_rech_date_ma   -5.330198
last_rech_amt_ma    -0.064244
```

- Random Forest Classifier, Decision Tree Classifier, AdaBoost Classifier, GradientBoosting Classifier, Bagging Classifier, XGB Classifier, SGD Classifier were our baseline methods. For most of the model implementations, the open-source Scikit-Learn package was used.

Train - Test split:

```
x=df.drop('label',axis=1)
y=df['label']

y.value_counts()
```

```
1    180172
0     26162
Name: label, dtype: int64
```

Accuracy_score of train-test:

```
y_pred=clf.predict(x_test)
print(accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
0.8784026106201839
[[  857  6948]
 [  579 53517]]
              precision    recall  f1-score   support

           0       0.60      0.11      0.19      7805
           1       0.89      0.99      0.93     54096

    accuracy                           0.88     61901
   macro avg       0.74      0.55      0.56     61901
weighted avg       0.85      0.88      0.84     61901
```

- Run and Evaluate selected models

- Our primary packages for this project are going to be pandas for data processing, NumPy to work with arrays, matplotlib & seaborn for data visualizations, and finally scikit-learn for building an evaluating our ML model.

  Models:

Decision Tree Classifier

```
classifiers(dtc)
```

```
DecisionTreeClassifier()
 99.9991546342948

 cross value score 90.6240567829937
ACCURACY SCORE:
 87.32815301852959
ROC AUC SCORE:
 76.57293610561791
CONFUSION MATRIX:
 [[ 4854  2951]
 [ 4893 49203]]
CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.50      0.62      0.55      7805
           1       0.94      0.91      0.93     54096

    accuracy                           0.87     61901
   macro avg       0.72      0.77      0.74     61901
weighted avg       0.89      0.87      0.88     61901
```

SGD Classifier

```
classifiers(sgd)
```

```
SGDClassifier()
 75.34279579346025

 cross value score 75.38971372948411
ACCURACY SCORE:
 73.34453401399008
ROC AUC SCORE:
 74.60239842671325
CONFUSION MATRIX:
 [[ 5954  1851]
 [14649 39447]]
CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.29      0.76      0.42      7805
           1       0.96      0.73      0.83     54096

    accuracy                           0.73     61901
   macro avg       0.62      0.75      0.62     61901
weighted avg       0.87      0.73      0.78     61901
```

Random Forest Classifier

```
RandomForestClassifier()
 score:  99.9991546342948
 F1 score: 94.77884873568327
Accuracy scoer: 90.89029256393273

 Cross value score 94.58543941124461
Confusion_matrix:
 [[ 5080  2725]
 [ 2914 51182]]
Classification report:
              precision    recall  f1-score   support

           0       0.64      0.65      0.64      7805
           1       0.95      0.95      0.95     54096

    accuracy                           0.91     61901
   macro avg       0.79      0.80      0.80     61901
weighted avg       0.91      0.91      0.91     61901
```

Ada-Boost Classifier

```
AdaBoostClassifier()
 score:  85.12325431981876
 F1 score: 89.87216146370702
Accuracy scoer: 83.34921891407248

 Cross value score 84.94699988034758
Confusion_matrix:
 [[ 5863  1942]
 [ 8365 45731]]
Classification report:
              precision    recall  f1-score   support

           0       0.41      0.75      0.53      7805
           1       0.96      0.85      0.90     54096

    accuracy                           0.83     61901
   macro avg       0.69      0.80      0.72     61901
weighted avg       0.89      0.83      0.85     61901
```

Gradient Boosting Classifier

```
GradientBoostingClassifier()
 score:  90.06906637811517
 F1 score: 92.35420585864016
Accuracy scoer: 87.13106411851182

 Cross value score 89.88182327467366
Confusion_matrix:
 [[ 5824  1981]
 [ 5985 48111]]
Classification report:
              precision    recall  f1-score   support

           0       0.49      0.75      0.59      7805
           1       0.96      0.89      0.92     54096

    accuracy                           0.87     61901
   macro avg       0.73      0.82      0.76     61901
weighted avg       0.90      0.87      0.88     61901
```

Bagging Classifier

```
BaggingClassifier()
 score:  99.60859567849052
 F1 score: 94.03492345863957
Accuracy scoer: 89.70775916382611

 Cross value score 93.31950463387034
Confusion_matrix:
 [[ 5313  2492]
 [ 3879 50217]]
Classification report:
              precision    recall  f1-score   support

           0       0.58      0.68      0.63      7805
           1       0.95      0.93      0.94     54096

    accuracy                           0.90     61901
   macro avg       0.77      0.80      0.78     61901
weighted avg       0.91      0.90      0.90     61901
```

XGB Classifier

```
classifiers(xgb)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              eval_metric='logloss', gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=4,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
 95.13534304940318

 cross value score 93.3190885097386
ACCURACY SCORE:
 91.15361625821878
ROC AUC SCORE:
 79.82511989978288
CONFUSION MATRIX:
 [[ 5048  2757]
 [ 2719 51377]]
CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.65      0.65      0.65      7805
           1       0.95      0.95      0.95     54096

    accuracy                           0.91     61901
   macro avg       0.80      0.80      0.80     61901
weighted avg       0.91      0.91      0.91     61901
```

- Key Metrics for success in solving problem under consideration
- Using the sklearn.metrics calculated Adjusted R2 squared ,Mean Absolute Error (MAE),Mean Squared Error (MSE),Root Mean Squared Error (RMSE)
  - Precision can be seen as a measure of quality, higher precision means that an algorithm returns more relevant results than irrelevant ones.
  - Recall is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.
  - Accuracy score is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.
  - F1-score is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.
  - Cross_val_score: To run cross-validation on multiple metrics and also to return train scores, fit times and score times. Get predictions from each split of cross-validation for diagnostic purposes. Make a scorer from a performance metric or loss function.
  - AUC_ROC_score: ROC curve. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0

  Using Hyper-parameter : model parameters are estimated from data automatically and model hyper-parameters are set manually and are used in processes to help estimate model and Grid search is a basic method for hyper-parameter tuning. It performs an exhaustive search on the hyper-parameter set specified by users.

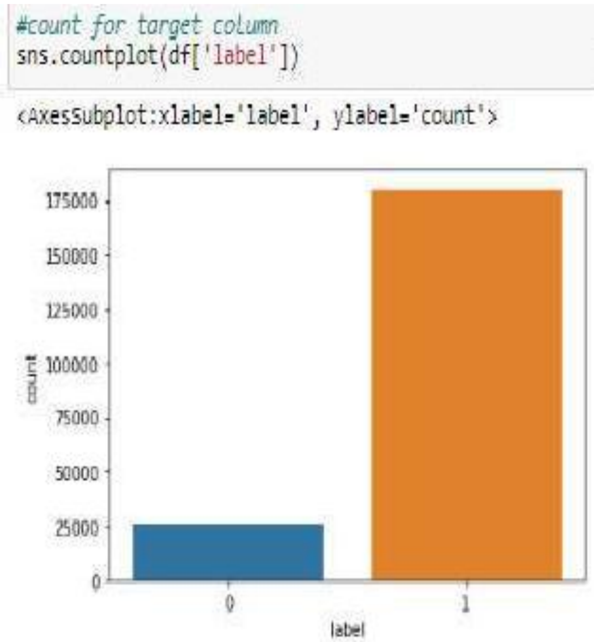
VIF (Inspect VIF Factors)

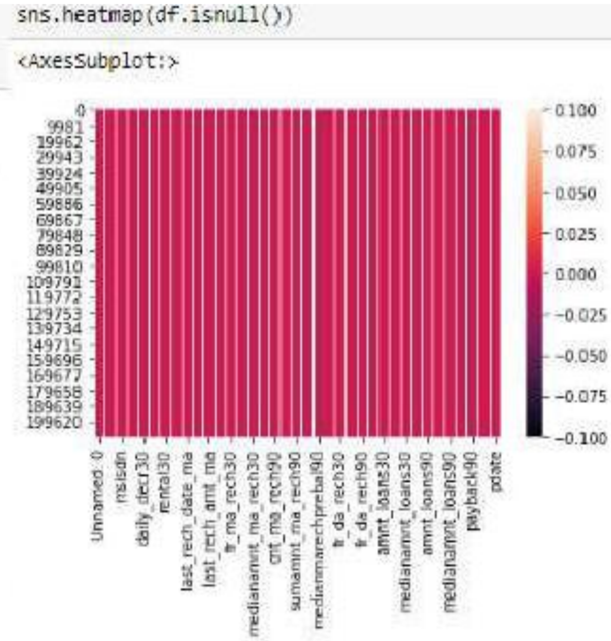
```
# For each X, calculate VIF and save in dataframe

from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(x1.values, i) for i in range(x1.shape[1])]
vif["features"] = x.columns

vif.round(1)
```

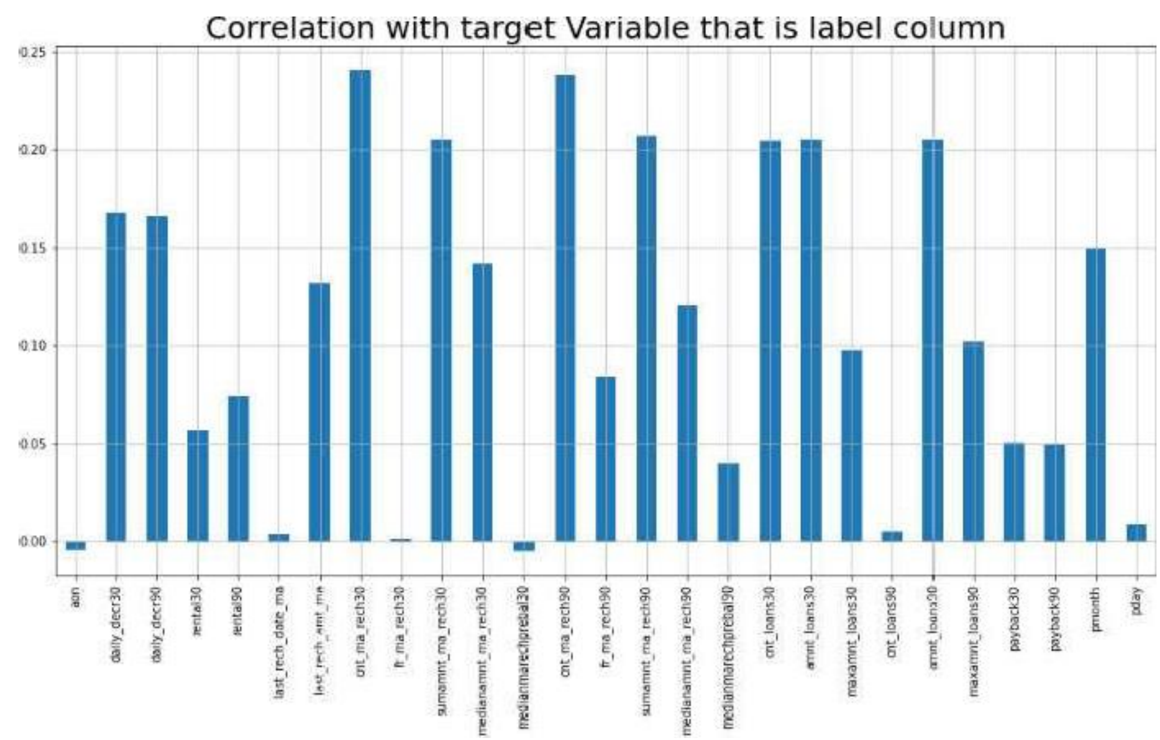
	VIF Factor	features
0	110.3	aon
1	2650698.9	daily_decr30
2	2684678.2	daily_decr90
3	95568.6	rental30
4	96076.1	rental90
5	190.3	last_rech_date_ma

- Visualizations

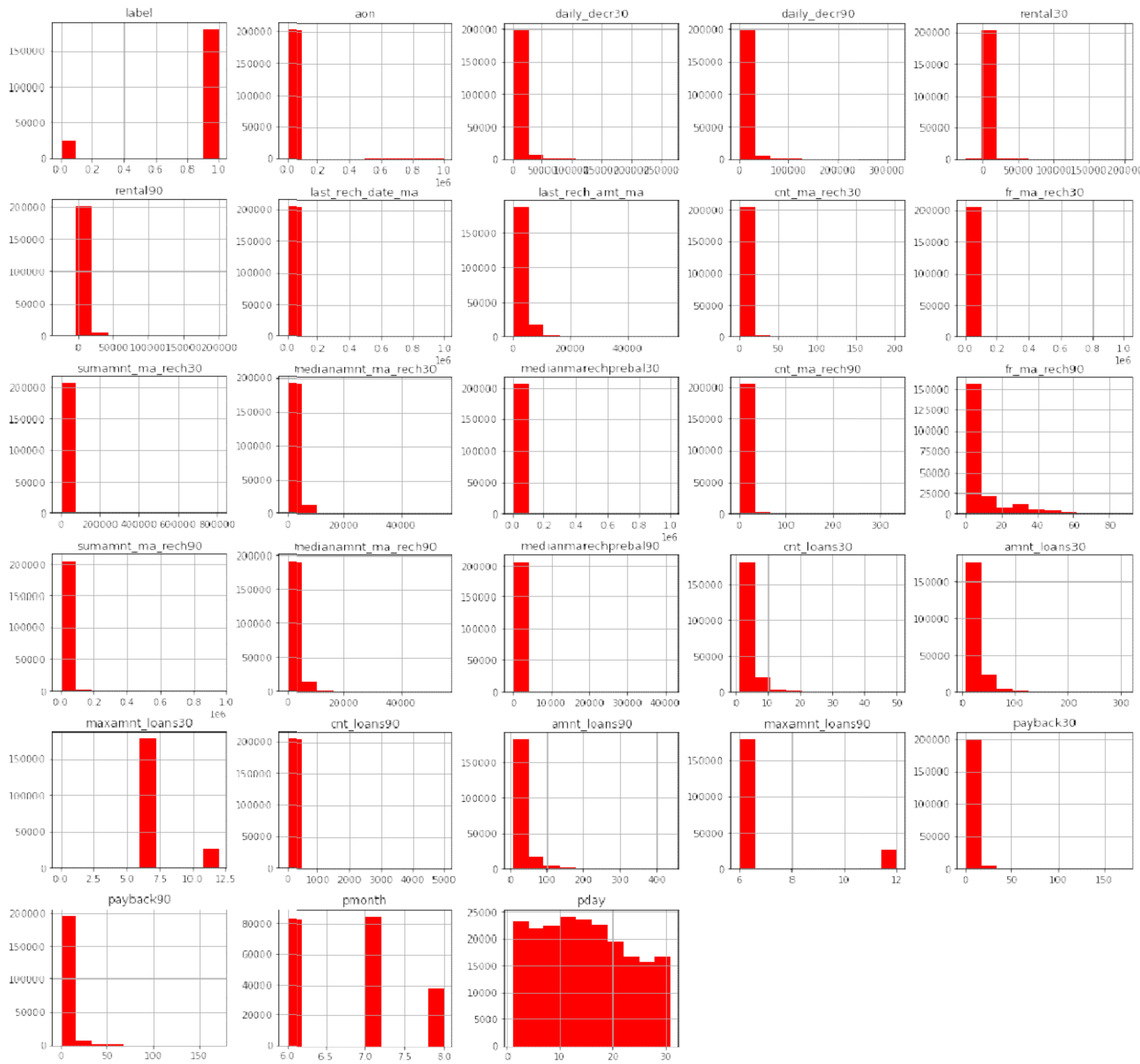
- As the value counts observation I find imbalance dataset in which defaulter values is less and Non defaulter values is high. About to 15% and 85% respectively
- Dataset is imbalanced. Label ‘1’ has approximately 87.5% records, while, label ‘0’ has approximately 12.5% records. Need to balance.





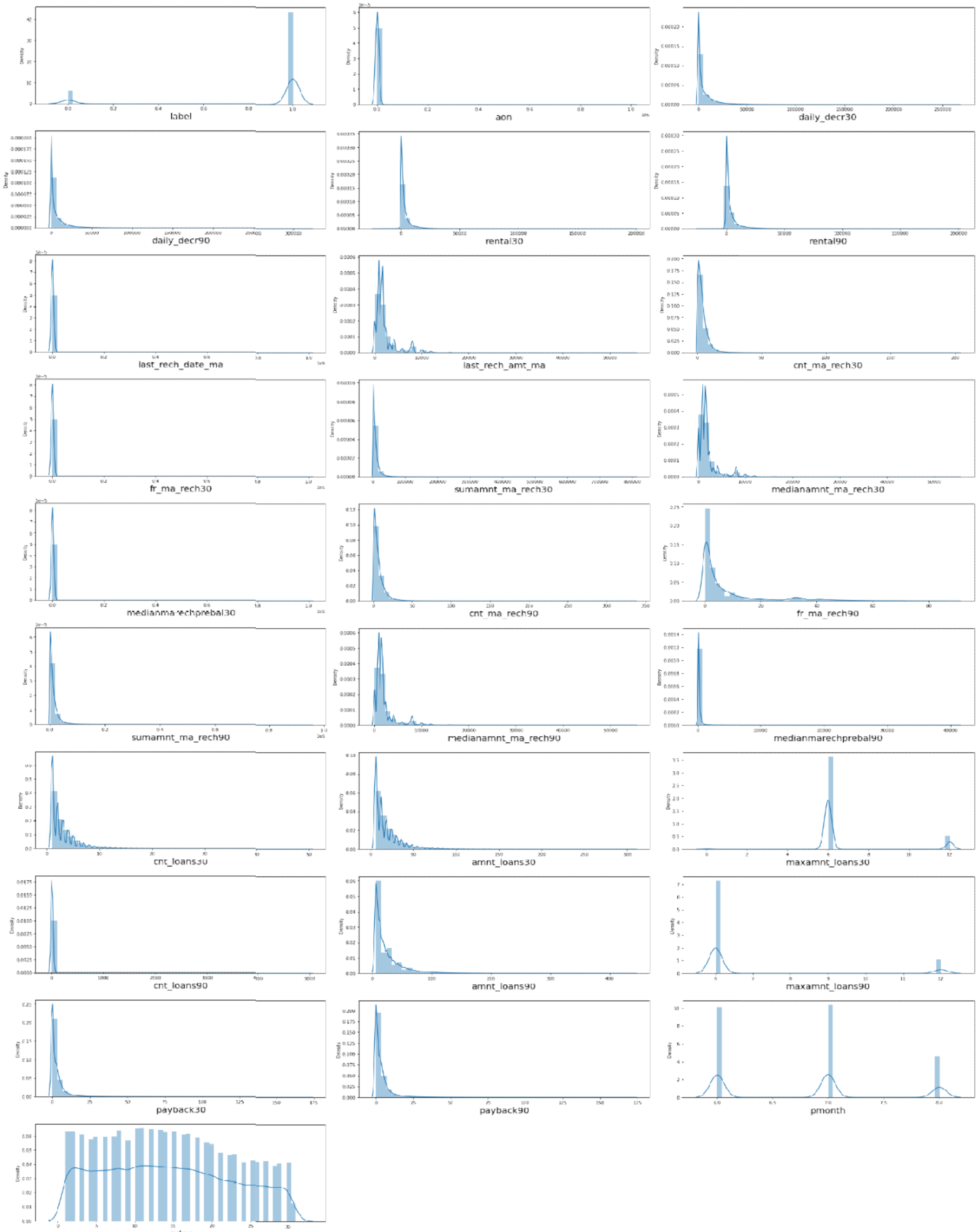


Plotting the Histogram



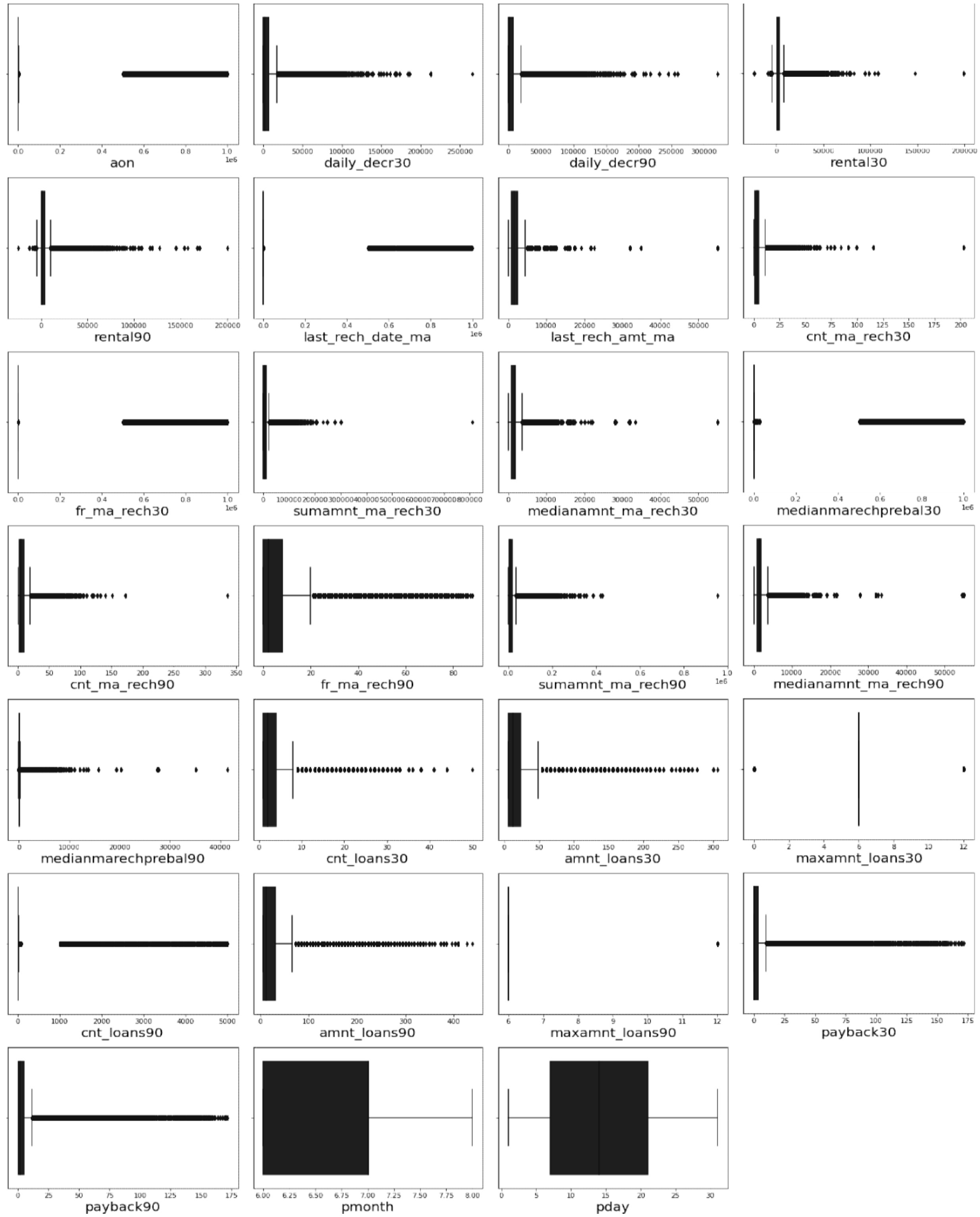
- To remove outliers I have used percentile method. And to remove skewness I have used yeo-johnson method. We have dropped all the unnecessary columns in the dataset according to our understanding. Use of Pearson’s correlation coefficient to check the correlation between dependent and independen features. Also I have used Normalization to scale the data. After scaling we have to balance the target column using oversampling. Then followed by model building with all Classification algorithms. I have used oversampling (SMOTETomek) to get rid of data unbalancing.

Bar plots to see the relation of numerical feature with target and 2 types of plots for numerical columns like distribution plot for univariate and bar plot for bivariate analysis.

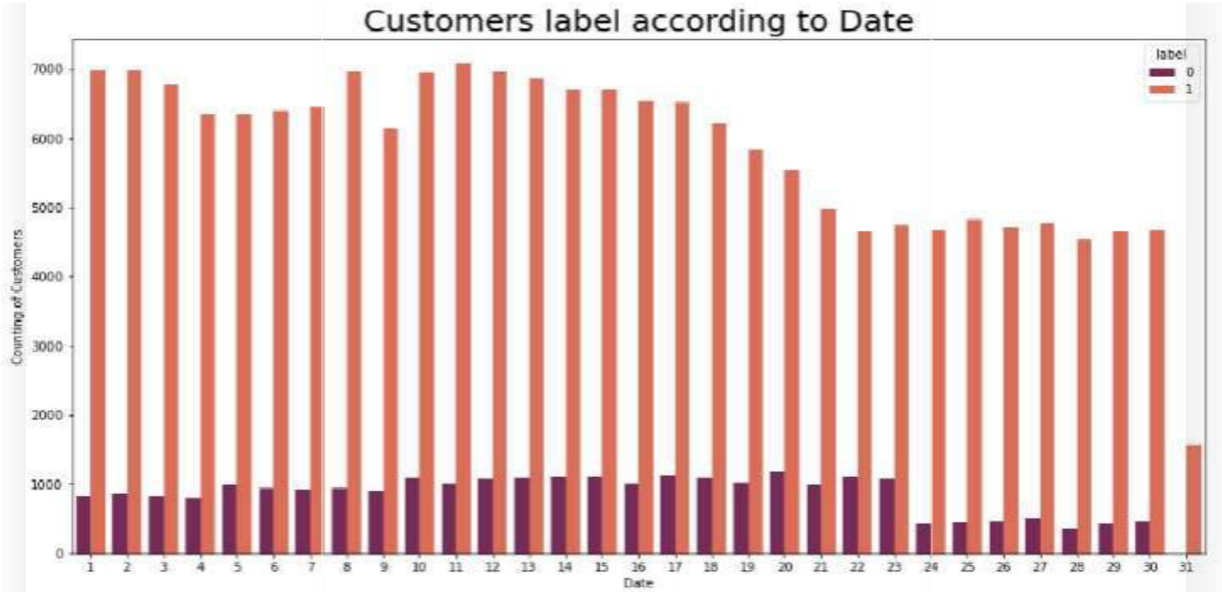


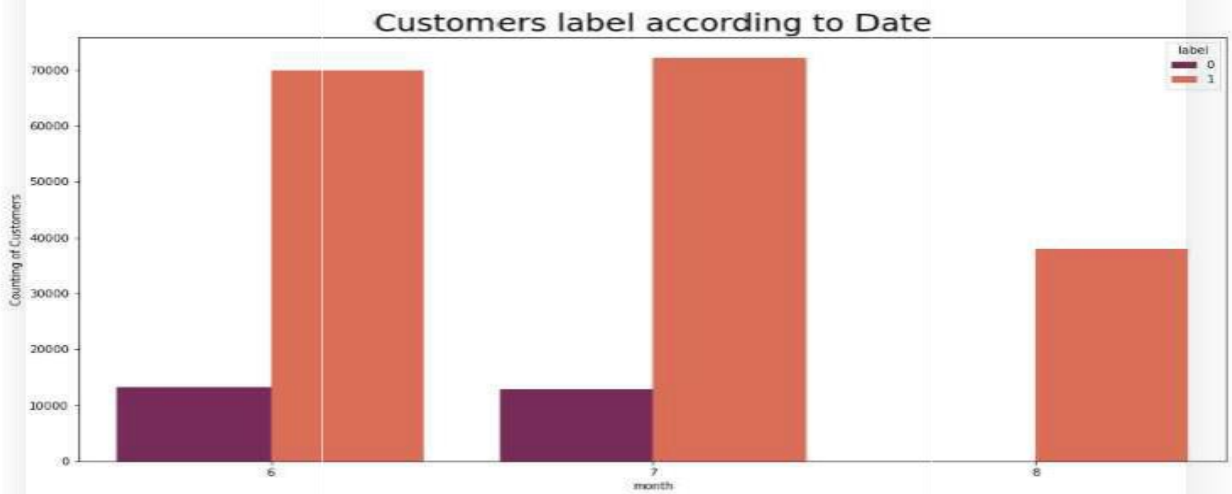
Outliers in most of the columns so we have to treat them using suitable methods.

Skewness seen in most of the columns so we have to treat them using suitable methods.



Loan count as per date and month (Non-defaulter and Defaulter):





Hyperparameter Tuning

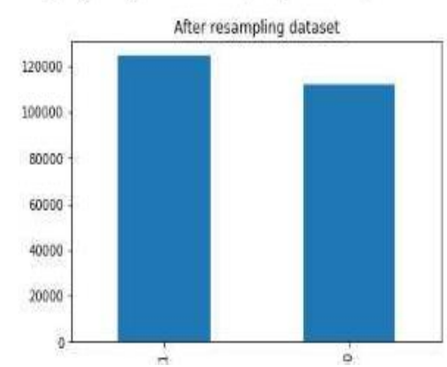
```
n_estimators=[int(x) for x in np.linspace(start=0,stop=100,num=10)]
max_features=['auto','sqrt','log2']
criterion=['gini','entropy']
max_depth=[2,4]
min_samples_split=[1,2]
bootstrap=[True,False]
```

```
para_grid={'n_estimators':n_estimators,
          'max_features':max_features,
          'criterion':criterion,
          'max_depth':max_depth,
          'min_samples_split':min_samples_split,
          'bootstrap':bootstrap}
```

```
grid=GridSearchCV(RandomForestClassifier(),para_grid,n_jobs=-1)
grid.fit(X_train_sm,Y_train_sm)
grid.best_params_
```

```
Y_train_sm.value_counts().plot(kind='bar')
plt.title('After resampling dataset')
```

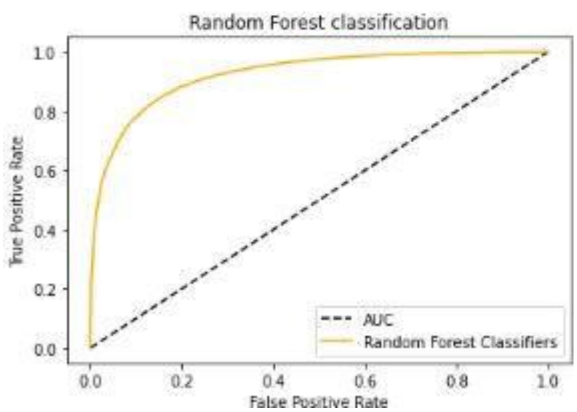
```
Text(0.5, 1.0, 'After resampling dataset')
```



Roc & Auc

- Present the receiver operating characteristic (ROC) curves and their respective areas under the curve (AUCs). ROC curves and AUCs are used to measure the quality of a classifier's output; thus, they measure how correctly a classifier has been tuned. Movement along the ROC curve is typically a trade-off between the classifier's sensitivity (true positive rate (TPR)) and specificity (TNR), and the steeper the curve, the better. For the ROC curve, sensitivity increases as we move up, and specificity decreases as we move right. The ROC curve along a 45_ angle

```
ROC AUC SCORE:  78.77763646889835
```



Saving best model

```
# Saving best model
import joblib
joblib.dump(rf,'Microcredit.plk')
```

```
['Microcredit.plk']
```

```
# Loading the saved model
model=joblib.load("Microcredit.plk")
```

- # Interpretation of the Results
  - In this research, two experiments were performed, the first experiment was validating and filtering data using all the variables available in the dataset after pre-processing, while the second experiment was conducted using most important variables and the goal of this is to be able to improve the model's performance using fewer variables.
  - Requirement of train and test and building of many models to get accuracy of the model.
  - There are multiple of matric which decide the best fit model like as : R-squared ,RMSE value, VIF, CDF & PDF Z-score , Roc & Auc and etc.
  - Database helped in making perfect model and will help in understanding Indonesian micro finance services (MFS) And use multiple metrics like F1_score, precision, recall and accuracy_score which will help to decide the best model.
  - Random forest Classifier as the best model with 91.18% accuracy_score..
  - Lastly predicted wheather the loan is paid back or not using saved model. It was good!! that was able to get the predictions near to actual values.

CONCLUSION

- Key Findings and Conclusions of the Study

  This research evaluated individuals' credit risk performance in a micro-finance environment using machine learning and deep learning techniques. While traditional methods utilizing models such as linear regression are commonly adopted to estimate reasonable accuracy nowadays, these models have been succeeded by extensive employment of machine and deep learning models that have been broadly applied and produce prediction outcomes with greater precision. Using real data, we compared the various machine learning algorithms' accuracy by performing detailed experimental analysis while classifying individuals' requesting a loan into three classes, namely, good, average, and poor.

  In this project report, we have used machine learning algorithms to predict the micro credit defaulters. We have mentioned the step by step procedure to analyze the dataset and finding the correlation between the features. Thus we can select the features which are correlated to each other and are independent in nature. These feature set were then given as an input to four algorithms and a hyper parameter tuning was done to the best model and the accuracy has been improved.

  Calculated the performance of each model using different performance metrics and compared them based on these metrics. Then we have also saved the best fit model and predicted the label. This is interesting that predicted and actual values were almost same.

Learning Outcomes of the Study in respect of Data Science

- Dataset is imbalanced. Label '1' has approximately 87.5% records, while, label '0' has approximately 12.5% records, and defaulter are higher.
- This model will be a good way for the management to understand whether the customer will be paying back the loaned amount within 5 days of insurance of loan. The relationship between predicting defaulter and the economy is an important motivating factor for predicting micro credit defaulter

• Limitations of this work and Scope for Future Work

- The length of the dataset it is very huge and hard to handle.
- Number of outliers and skewness these two will reduce our model accuracy.
- Also, we have tried best to deal with outliers, skewness and zero values. So it looks quite good that we have achieved a accuracy of 91.32% even after dealing all these drawbacks.
- This study will not cover all Classification algorithms instead, it is focused on the chosen algorithm, starting from the basic assembling techniques to the advanced ones.

Thank You