FLIP ROBO

Micro-Credit Project

Submitted by:

Abhishek Nipane

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to my SME Mr Shubham Yadav Sir of Flip ROBO as well as Data-Trained who gave me the golden opportunity to do this wonderful project on Micro-Credit, which also helped me in doing a lot of Research and i came to know about so many new things and techniques while doing so.

Secondly i would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

I took a great deal of help from Wikipedia, Towards Data-Science, Kaggle and Stack-Overflow.

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 industries 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.

We are working with one such client that is in Telecom Industry. They are a fixed wireless telecommunications network provider. 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.

They understand the importance of communication and how it affects a person's life, thus, focusing on providing their services and products to low income families and poor customers that can help them in the need of hour.

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.

- 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 insurance.

Conceptual Background of the Domain Problem

Delinquency is a condition that arises when an activity or situation does not occur at its scheduled (or expected) date i.e., it occurs later than expected.

Use Case: Many donors, experts, and microfinance institutions (MFI) have become convinced that using mobile financial services (MFS) is more convenient and efficient, and less costly, than the traditional high-touch model for delivering microfinance services. MFS becomes especially useful when targeting the unbanked poor living in remote areas. The implementation of MFS, though, 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.

One of our Clients in Telecom collaborates with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be delinquent if he deviates from the path of paying back the loaned amount within 5 days.

Motivation for the Problem Undertaken

As I am a part of the Internship Process of RoboFlip Technologies, this project was given to me as the first assignment after getting accepted as a Data Science Intern in RoboFlip Technologies.

I took the project as a challenge for myself to see how I have improved and upgraded from the day I started learning with Data-Trained.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  As from the Problem statement we can understand that this Dataset needs Classification for Model Building.

  So what is Classification?

Classification can be performed on structured or unstructured data. Classification is a technique where we categorize data into a given number of classes. The main goal of a classification problem is to identify the category/class to which a new data will fall under.

Few of the terminologies encountered in machine learning – classification:

- **Classifier:** An algorithm that maps the input data to a specific category.

- **Classification_model:** A classification model tries to draw some conclusion from the input values given for training. It will predict the class labels/categories for the new data.

- **Feature:** A feature is an individual measurable property of a phenomenon being observed.

- **Binary Classification:** Classification task with two possible outcomes. Ex: Gender classification (Male / Female)

- **Multi-class classification:** Classification with more than two classes. In multi class classification each sample is assigned to one and only one target label. Ex: An animal can be cat or dog but not both at the same time

- **Multi-label classification:** Classification task where each sample is mapped to a set of target labels (more than one class). Ex: A news article can be about sports, a person, and location at the same time.

The following are the steps involved in building a classification model:

- **Initialize** the classifier to be used.
- **Train the classifier:** All classifiers in scikit-learn uses a fit(X, y) method to fit the model (training) for the given train data X and train label y.

- **Predict the target:** Given an unlabelled observation X, the predict(X) returns the predicted label y.
- **Evaluate** the classifier model

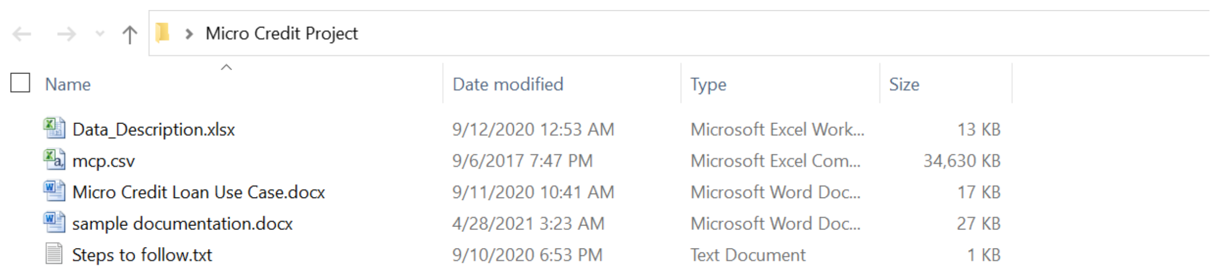
So we have somewhat understand what classification is and why we will be using this in Model building of our Dataset.

We can also see that the Dataset is highly Imbalanced which may create a biased model as our computer will pass all the minority class as noise.
So that's why we will be using SMOTE to balance out the entire dataset.

- Data Sources and their formats

The Micro-Credit project Dataset was sourced from RoboFlip Technologies as a part of their internship project.

- The Dataset is in a .csv (comma separated value).
- The general information about the dataset is given in a documentation format.
- Information about the dependent and independent variables are given in an Excel Format.



Micro Credit Project

Name	Date modified	Type	Size
Data_Description.xlsx	9/12/2020 12:53 AM	Microsoft Excel Work...	13 KB
mcp.csv	9/6/2017 7:47 PM	Microsoft Excel Com...	34,630 KB
Micro Credit Loan Use Case.docx	9/11/2020 10:41 AM	Microsoft Word Doc...	17 KB
sample documentation.docx	4/28/2021 3:23 AM	Microsoft Word Doc...	27 KB
Steps to follow.txt	9/10/2020 6:53 PM	Text Document	1 KB

Above is the screenshot of Data Sources and their Formats.

- Data Preprocessing Done

  This is the most crucial step of all in Building a Machine Learning Model.

  So first, what is Data Pre-processing?

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, lacking in certain behaviours or trends, and is likely to contain many errors.

Data pre-processing is a proven method of resolving such issues. Data pre-processing prepares raw data for further processing.

Data pre-processing is used in database-driven applications such as customer relationship management and rule-based applications (like neural networks).

In Machine Learning (ML) processes, data pre-processing is critical to encode the dataset in a form that could be interpreted and parsed by the algorithm.

Data goes through a series of steps during pre-processing:

Data Cleaning: Data is cleansed through processes such as filling in missing values or deleting rows with missing data, smoothing the noisy data, or resolving the inconsistencies in the data.

Smoothing noisy data is particularly important for ML datasets, since machines cannot make use of data they cannot interpret. Data can be cleaned by dividing it into equal size segments that are thus smoothed (binning), by fitting it to a linear or multiple regression function (regression), or by grouping it into clusters of similar data (clustering).

Data inconsistencies can occur due to human errors (the information was stored in a wrong field). Duplicated values should be removed through reduplication to avoid giving that data object an advantage (bias).

Data Integration: Data with different representations are put together and conflicts within the data are resolved.

Data Transformation: Data is normalized and generalized. Normalization is a process that ensures that no data is redundant, it is all stored in a single place, and all the dependencies are logical.

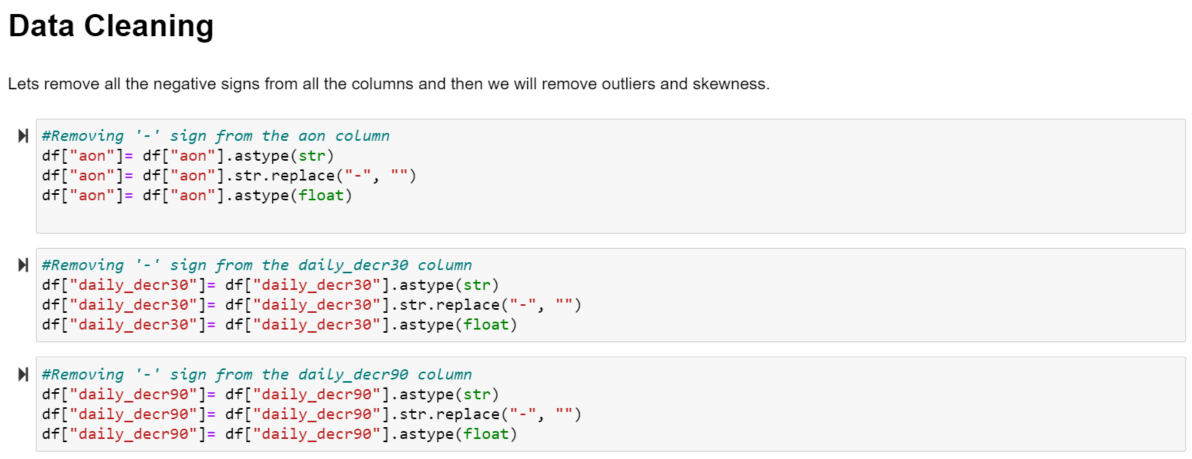
Data Reduction: When the volume of data is huge, databases can become slower, costly to access, and challenging to properly store. Data reduction step aims to present a reduced representation of the data in a data warehouse.

There are various methods to reduce data. For example, once a subset of relevant attributes is chosen for its significance, anything below a given level is discarded. Encoding mechanisms can be used to reduce the size of data as well. If all original data can be recovered after compression, the operation is labelled as lossless.

If some data is lost, then it's called a lossy reduction. Aggregation can also be used, for example, to condense countless transactions into a single weekly or monthly value, significantly reducing the number of data objects.

Data Discretization: Data could also be discretized to replace raw values with interval levels. This step involves the reduction of a number of values of a continuous attribute by dividing the range of attribute intervals.

Data Sampling: Sometimes, due to time, storage or memory constraints, a dataset is too big or too complex to be worked with. Sampling techniques can be used to select and work with just a subset of the dataset, provided that it has approximately the same properties of the original one.



Data Cleaning

Lets remove all the negative signs from all the columns and then we will remove outliers and skewness.

```
#Removing '-' sign from the aon column
df["aon"]= df["aon"].astype(str)
df["aon"]= df["aon"].str.replace("-", "")
df["aon"]= df["aon"].astype(float)
```

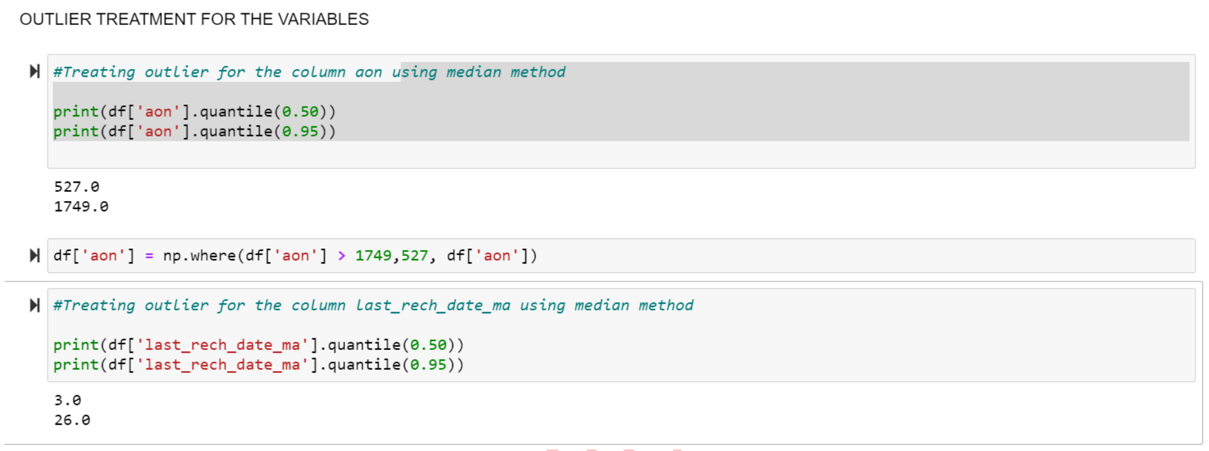
```
#Removing '-' sign from the daily_decr30 column
df["daily_decr30"]= df["daily_decr30"].astype(str)
df["daily_decr30"]= df["daily_decr30"].str.replace("-", "")
df["daily_decr30"]= df["daily_decr30"].astype(float)
```

```
#Removing '-' sign from the daily_decr90 column
df["daily_decr90"]= df["daily_decr90"].astype(str)
df["daily_decr90"]= df["daily_decr90"].str.replace("-", "")
df["daily_decr90"]= df["daily_decr90"].astype(float)
```

So now we have established what is Data pre-processing, I will let know all the steps I took to clean the Data before proceeding:

- Firstly I checked for Null Values, there were none so we have not done anything with that.

- Secondly I checked for any Nan values in the dataset which I found, so I filled all the Nan values with zero rather than removing it.
- Thirdly I found outliers and skewness in the Dataset which I dealt with by using median of the columns and replacing it with respective medians.
- I also found some ‘-‘sign in the dataset. Which I removed with the help of replace statement.



OUTLIER TREATMENT FOR THE VARIABLES

```
#Treating outlier for the column aon using median method

print(df['aon'].quantile(0.50))
print(df['aon'].quantile(0.95))
```

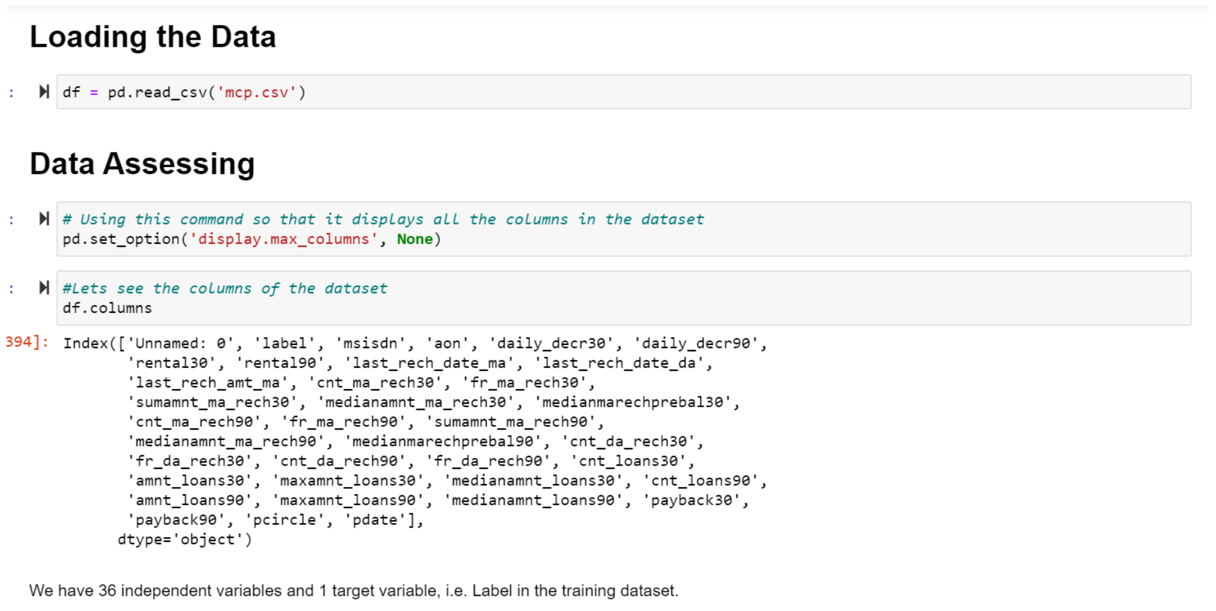
```
527.0
1749.0
```

```
df['aon'] = np.where(df['aon'] > 1749,527, df['aon'])
```

```
#Treating outlier for the column last_rech_date_ma using median method

print(df['last_rech_date_ma'].quantile(0.50))
print(df['last_rech_date_ma'].quantile(0.95))
```

```
3.0
26.0
```

- Data Inputs- Logic- Output Relationships



Loading the Data

```
: df = pd.read_csv('mcp.csv')
```

Data Assessing

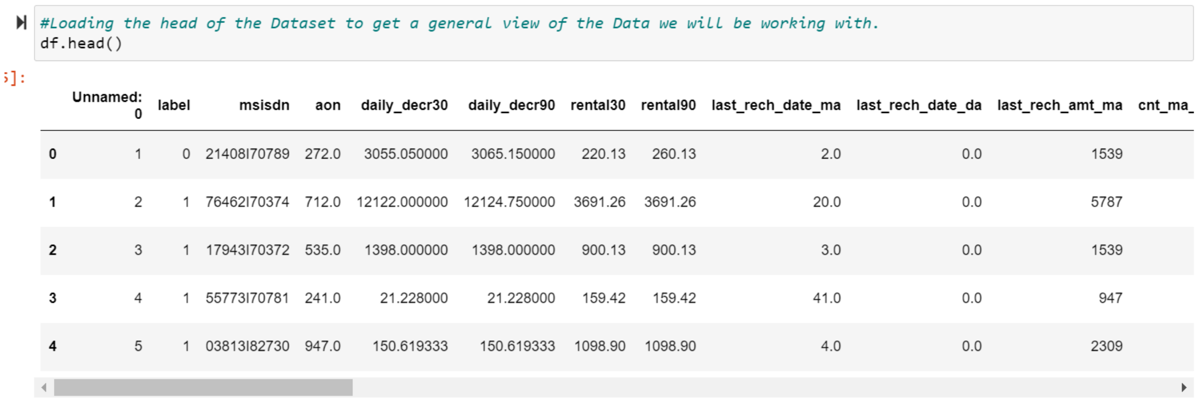
```
: # Using this command so that it displays all the columns in the dataset
  pd.set_option('display.max_columns', None)
```

```
: #Lets see the columns of the dataset
  df.columns
```

```
394]: Index(['Unnamed: 0', 'label', 'msisdn', 'aon', 'daily_decr30', 'daily_decr90',
       'rental30', 'rental90', 'last_rech_date_ma', 'last_rech_date_da',
       '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_da_rech30',
       'fr_da_rech30', 'cnt_da_rech90', 'fr_da_rech90', 'cnt_loans30',
       'amnt_loans30', 'maxamnt_loans30', 'medianamnt_loans30', 'cnt_loans90',
       'amnt_loans90', 'maxamnt_loans90', 'medianamnt_loans90', 'payback30',
       'payback90', 'pcircle', 'pdate'],
      dtype='object')
```

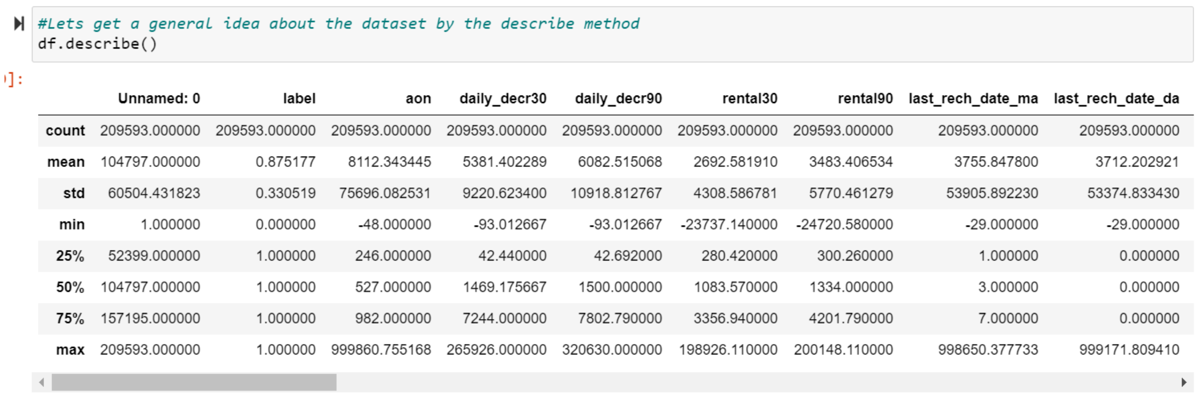
We have 36 independent variables and 1 target variable, i.e. Label in the training dataset.

From the above snapshot we can see how we load the data and get to know its dimensions.



```
#Loading the head of the Dataset to get a general view of the Data we will be working with.
df.head()
```

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma	cnt_ma_
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	1539	
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	5787	
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	1539	
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	947	
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	2309	



```
#Lets get a general idea about the dataset by the describe method
df.describe()
```

	Unnamed: 0	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000
mean	104797.000000	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800	3712.202921
std	60504.431823	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230	53374.833430
min	1.000000	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000
25%	52399.000000	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	0.000000
50%	104797.000000	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	0.000000
75%	157195.000000	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	0.000000
max	209593.000000	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410

So from both the tables above we can conclude that:

- There are no null values in the dataset that we have to deal with.
- There also seems to be outliers for some of the features.
- And some kind of skewness can also be seen from the description which we will see it more clearly later on.

Real World/Business objectives & Constraints

- No low-Latency requirement
- Interpretability is important
- Probability of a Data-Point belonging to each class is needed.

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

As from the Problem statement which states that we have to build a model which will predict default for a person. Since the target variable is not continuous we can readily presume that it is a Classification Problem for which we will be using Classification algorithms to build our model from the dataset given.

Secondly as we know Data is very expensive I tried those methods for Data cleaning which doesn't require removal of bulk of Data.

Like for example I did not use IQR method for Outliers as it would have erased huge amounts of Data. Instead I used Median method for Outlier treatment.

Thirdly we can clearly see that the Dataset is highly Imbalanced which may create a biased model as our computer will pass all the minority class as noise. So that's why we will be using SMOTE to balance out the entire dataset.

Lastly I will be using RandomSearch CV for hyper parameter tuning as it uses least amount of memory and processing power to run the algorithms as compared to that of GridSearch CV.

- Hardware and Software Requirements and Tools Used

Hardware Tools Used:

- Lenovo Ideapad Gaming 3 Laptop with 16GB of RAM.
- Inbuilt AMD Radeon Graphics and External Nvidia GTX 1650 ti graphics card.
- Ryzen 5 processor
- 512 GB SSD

Software Tools Used:

- Anaconda for calling JN.
- Jupyter Notebook for Code handling and Visualization.
- Python Shell for installing some libraries.
- Excel for calling .CSV(comma separated values)
- Microsoft Word for documentation.
- Word to pdf Converter Online.(WBA)
- Microsoft Power Point Presentation.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

So below are the few things that I faced during the project:

- There are no null values in the dataset.
- 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.
- For some features, there may be values which might not be realistic.
- I also came across outliers in some features.

- Testing of Identified Approaches (Algorithms)

So as we know it is a Classification problem we will use classification algorithms to train our Dataset. Following are some Classification Algorithms:

- Logistic Regression
- Naïve Bayes
- Stochastic Gradient Descent
- K-Nearest Neighbours
- Decision Tree
- Random Forest
- Support Vector Machine

We will be using some of the above algorithms to test our model. We may leave some Algorithms just because it takes huge amount of time to iterate the Data.

Let's understand each of the Classification algorithms briefly :

Logistic Regression

Definition: Logistic regression is a machine learning algorithm for classification. In this algorithm, the probabilities describing the possible outcomes of a single trial are modelled using a logistic function.

Advantages: Logistic regression is designed for this purpose (classification), and is most useful for understanding the influence of several independent variables on a single outcome variable.

Disadvantages: Works only when the predicted variable is binary, assumes all predictors are independent of each other and assumes data is free of missing values.

Naïve Bayes

Definition: Naive Bayes algorithm based on Bayes' theorem with the assumption of independence between every pair of features. Naive Bayes classifiers work well in many real-world situations such as document classification and spam filtering.

Advantages: This algorithm requires a small amount of training data to estimate the necessary parameters. Naive Bayes classifiers are extremely fast compared to more sophisticated methods.

Disadvantages: Naive Bayes is is known to be a bad estimator.

Stochastic gradient descent

Definition: Stochastic gradient descent is a simple and very efficient approach to fit linear models. It is particularly useful when the number of samples is very large. It supports different loss functions and penalties for classification.

Advantages: Efficiency and ease of implementation.

Disadvantages: Requires a number of hyper-parameters and it is sensitive to feature scaling.

K-Nearest Neighbours

Definition: Neighbours based classification is a type of lazy learning as it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the k nearest neighbours of each point.

Advantages: This algorithm is simple to implement, robust to noisy training data , and effective if training data is large.

Disadvantages: Need to determine the value of K and the computation cost is high as it needs to compute the distance of each instance to all the training samples.

Decision Tree

Definition: Given a data of attributes together with its classes, a decision tree produces a sequence of rules that can be used to classify the data.

Advantages: Decision Tree is simple to understand and visualise, requires little data preparation, and can handle both numerical and categorical data.

Disadvantages: Decision tree can create complex trees that do not generalise well, and decision trees can be unstable because small variations in the data might result in a completely different tree being generated.

Random Forest

Definition: Random forest classifier is a meta-estimator that fits a number of decision trees on various sub-samples of datasets and uses average to improve the predictive accuracy of the model and controls over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement.

Advantages: Reduction in over-fitting and random forest classifier is more accurate than decision trees in most cases.

Disadvantages: Slow real time prediction, difficult to implement, and complex algorithm.

Support Vector Machine

Definition: Support vector machine is a representation of the training data as points in space separated into categories by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

Advantages: Effective in high dimensional spaces and uses a subset of training points in the decision function so it is also memory efficient.

Disadvantages: The algorithm does not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

Comparison Matrix

- **Accuracy: (True Positive + True Negative) / Total Population**
    - Accuracy is a ratio of correctly predicted observation to the total observations. Accuracy is the most intuitive performance measure.
    - True Positive: The number of correct predictions that the occurrence is positive

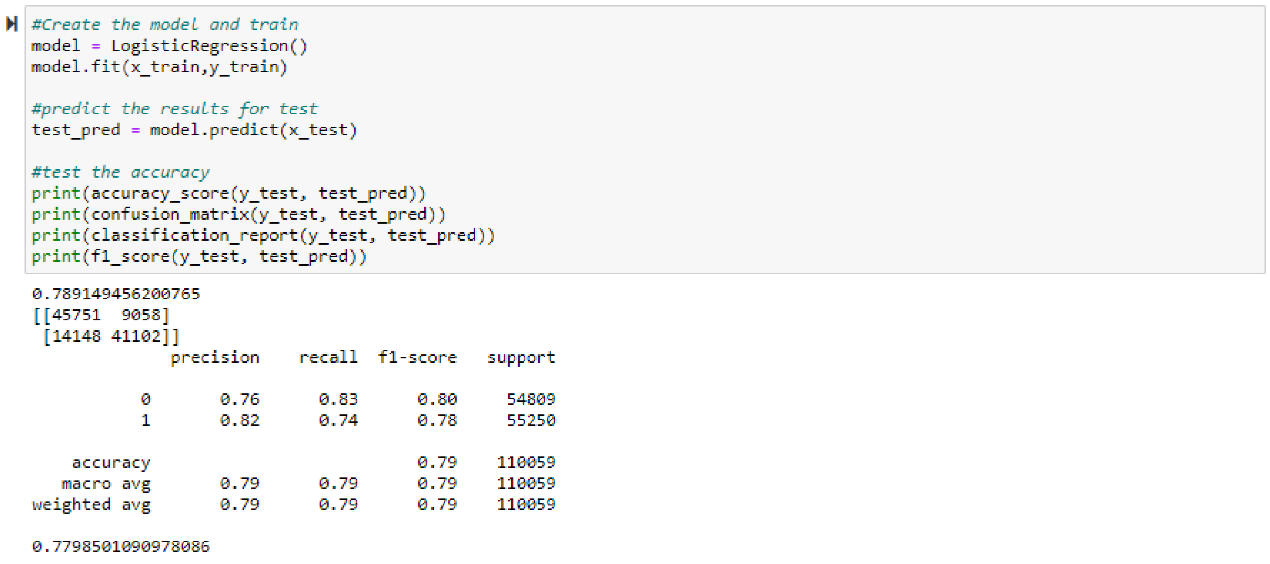
- True Negative: The number of correct predictions that the occurrence is negative

- **F1-Score: (2 x Precision x Recall) / (Precision + Recall)**
  - F1-Score is the weighted average of Precision and Recall used in all types of classification algorithms. Therefore, this score takes both false positives and false negatives into account. F1-Score is usually more useful than accuracy, especially if you have an uneven class distribution.
  - Precision: When a positive value is predicted, how often is the prediction correct?
  - Recall: When the actual value is positive, how often is the prediction correct?

Run and Evaluate selected models

So for the project Micro Credit I will be using these following algorithms which I choose by keeping in mind the time taken to iterate and what is better for this Dataset:

- So the first Model which I am going to use is Logistic Regression.

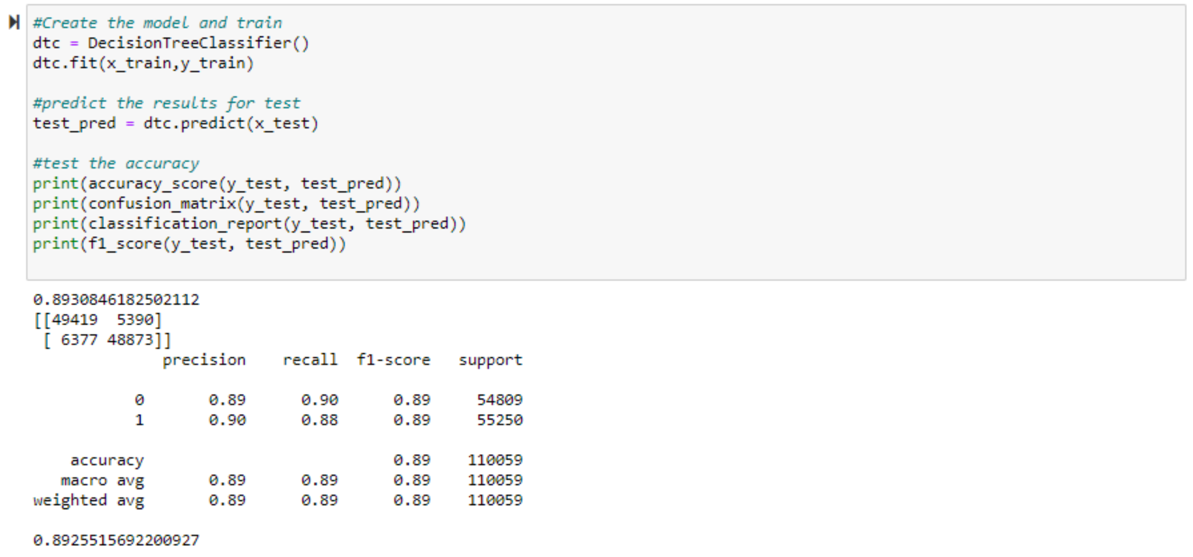


```
#Create the model and train
model = LogisticRegression()
model.fit(x_train,y_train)

#predict the results for test
test_pred = model.predict(x_test)

#test the accuracy
print(accuracy_score(y_test, test_pred))
print(confusion_matrix(y_test, test_pred))
print(classification_report(y_test, test_pred))
print(f1_score(y_test, test_pred))
```

```
0.789149456200765
[[45751  9058]
 [14148 41102]]
              precision    recall  f1-score   support

           0       0.76      0.83      0.80     54809
           1       0.82      0.74      0.78     55250

    accuracy                           0.79    110059
   macro avg       0.79      0.79      0.79    110059
weighted avg       0.79      0.79      0.79    110059

0.7798501090978086
```

From the above we can see the accuracy and F1 score is not so high, so let’s try with another algorithm to make the score better.

- So the second Model which I am going to use is Decision Tree Classifier.

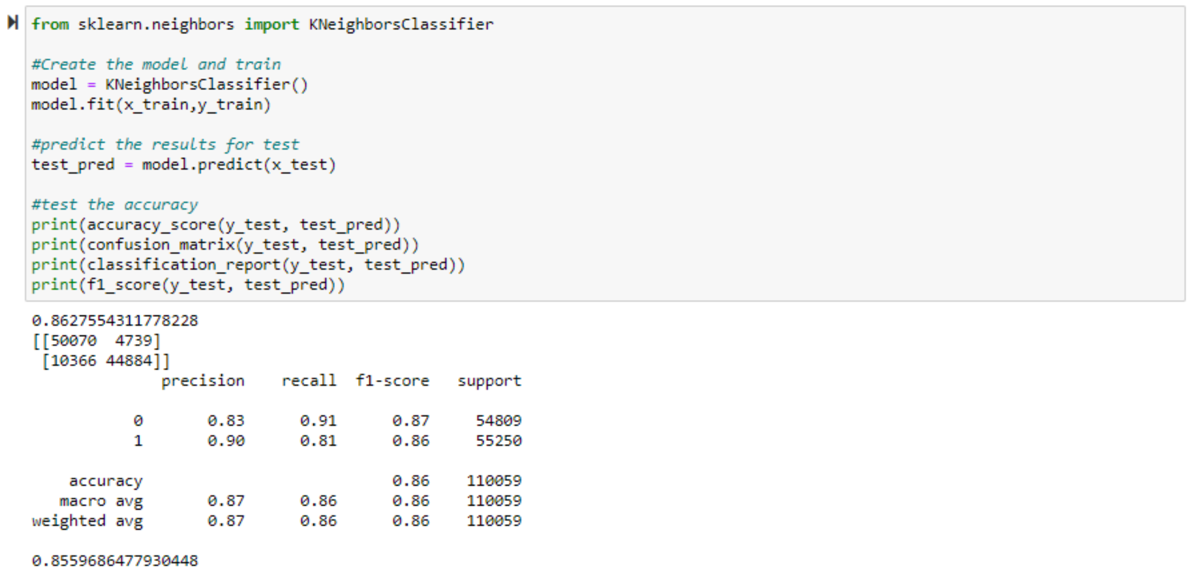


```
#Create the model and train
dtc = DecisionTreeClassifier()
dtc.fit(x_train,y_train)

#predict the results for test
test_pred = dtc.predict(x_test)

#test the accuracy
print(accuracy_score(y_test, test_pred))
print(confusion_matrix(y_test, test_pred))
print(classification_report(y_test, test_pred))
print(f1_score(y_test, test_pred))
```

```
0.8930846182502112
[[49419  5390]
 [ 6377 48873]]
              precision    recall  f1-score   support

           0       0.89      0.90      0.89     54809
           1       0.90      0.88      0.89     55250

    accuracy                           0.89    110059
   macro avg       0.89      0.89      0.89    110059
weighted avg       0.89      0.89      0.89    110059

0.8925515692200927
```

From the above we can see the accuracy and F1 score is not so high, so let's try with another algorithm to make the score better.

- So the Third Model which I am going to use is KNeighbors Classifier.

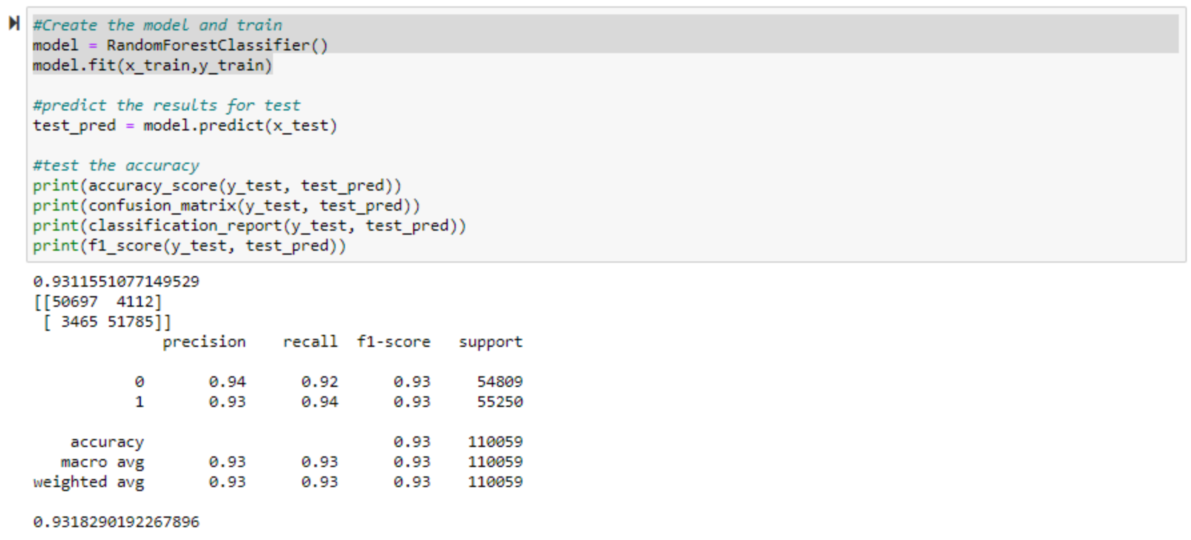


```
from sklearn.neighbors import KNeighborsClassifier

#Create the model and train
model = KNeighborsClassifier()
model.fit(x_train,y_train)

#predict the results for test
test_pred = model.predict(x_test)

#test the accuracy
print(accuracy_score(y_test, test_pred))
print(confusion_matrix(y_test, test_pred))
print(classification_report(y_test, test_pred))
print(f1_score(y_test, test_pred))
```

```
0.8627554311778228
[[50070  4739]
 [10366 44884]]
              precision    recall  f1-score   support

           0       0.83      0.91      0.87     54809
           1       0.90      0.81      0.86     55250

    accuracy                           0.86    110059
   macro avg       0.87      0.86      0.86    110059
weighted avg       0.87      0.86      0.86    110059

0.8559686477930448
```

From the above we can see the accuracy and F1 score is not so high, so let's try with another algorithm to make the score better.

- So the Fourth Model which I am going to use is Random Forest Classifier.



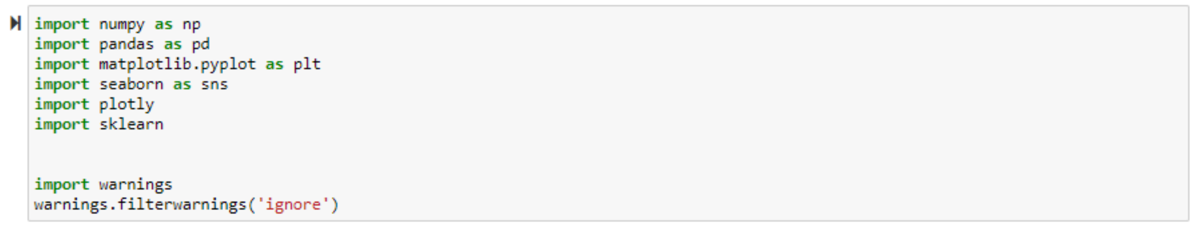
```
#Create the model and train
model = RandomForestClassifier()
model.fit(x_train,y_train)

#predict the results for test
test_pred = model.predict(x_test)

#test the accuracy
print(accuracy_score(y_test, test_pred))
print(confusion_matrix(y_test, test_pred))
print(classification_report(y_test, test_pred))
print(f1_score(y_test, test_pred))
```

```
0.9311551077149529
[[50697  4112]
 [ 3465 51785]]
              precision    recall  f1-score   support

           0       0.94      0.92      0.93     54809
           1       0.93      0.94      0.93     55250

    accuracy                           0.93    110059
   macro avg       0.93      0.93      0.93    110059
weighted avg       0.93      0.93      0.93    110059

0.9318290192267896
```

From the above we can see that we have a F1 score and Accuracy score of 0.93. We have somewhat achieved a good model. Hence we will stop our model building here as the other classification algorithms take a lot of memory and power to run the model.

Visualizations (EDA) & Interpretation of Results:

We will be using Libraries such as Seaborn, matplotlib, plotly. At first we will be importing the libraries so that we can use visualizations for our Dataset.

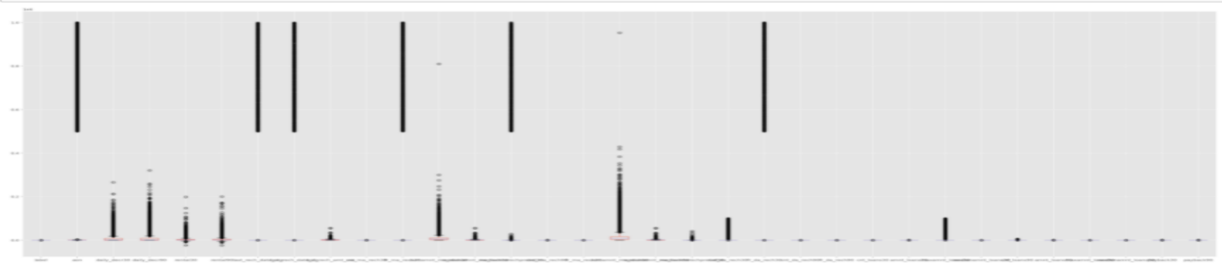


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import sklearn


import warnings
warnings.filterwarnings('ignore')
```

So we have called the Libraries, now let us use these libraries to visualize and do our EDA to understand the data better as well as to find outliers and skewness in the Data.

Visualization to find outliers in the Dataset:

```
df.plot.box(figsize=(40,15))
plt.show()
```

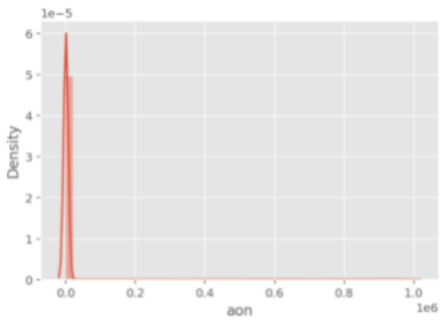


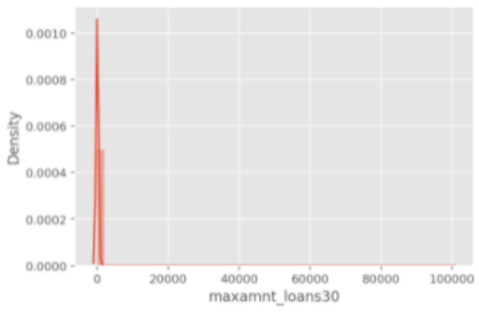
› from the above we can conclude that there are presence of Outliers in most of the features in the Dataset, which we will handle later one by one.

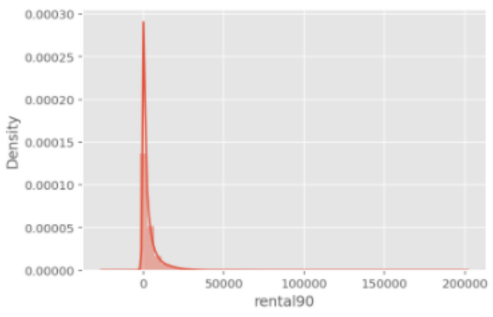
So we can see a lot of outliers in most of the variables, which we will deal later by using median method to replace the outliers.

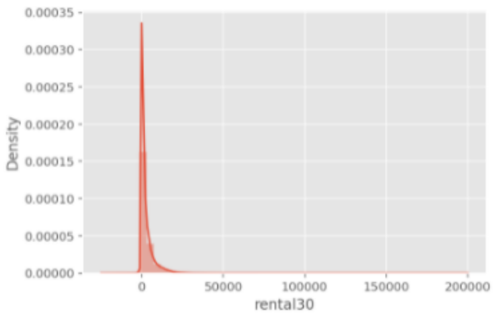
Visualisation for some Independent Variables along with their skewness:

```python
#Lets see skewness for some of the columns
columns=['aon','maxamnt_loans30','rental90','rental30']

for i in df[columns]:
    plt.figure()
    sns.distplot(df[i])
```







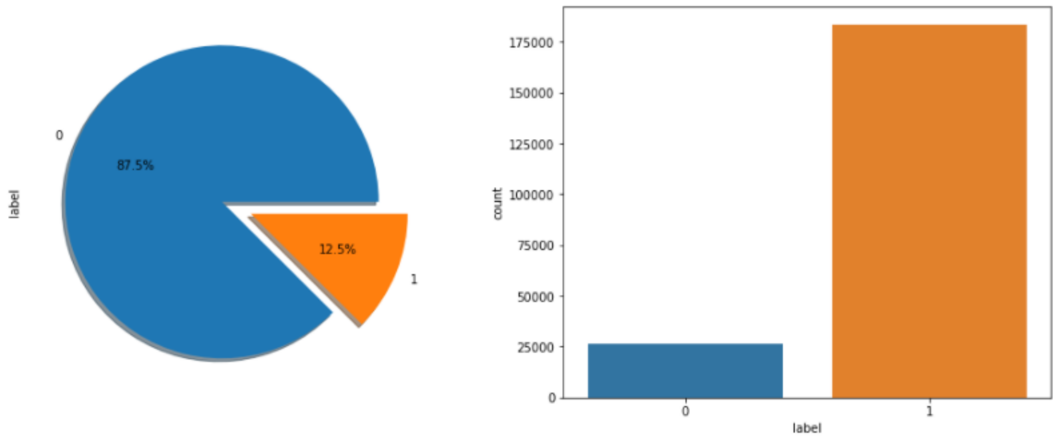


So from the above we can see that many of the variables are rightly skewed which we will fix when we are treating outliers.

Let’s do some Univariate Analysis of the Label column to understand the target variable better.

```
#Univariate Analysis

print(df['label'].value_counts())
f,ax=plt.subplots(1,2,figsize=(16,6))
labels = ['0', '1']
df['label'].value_counts().plot.pie(explode=[0,0.2],autopct='%1.1f%%',ax=ax[0],shadow=True,labels=labels,fontsize=10)
sns.countplot('label',data=df, ax=ax[1])
ax[1].set_xticklabels(['0', '1'], fontsize=10)
plt.show()
```

```
1    183431
0     26162
Name: label, dtype: int64
```

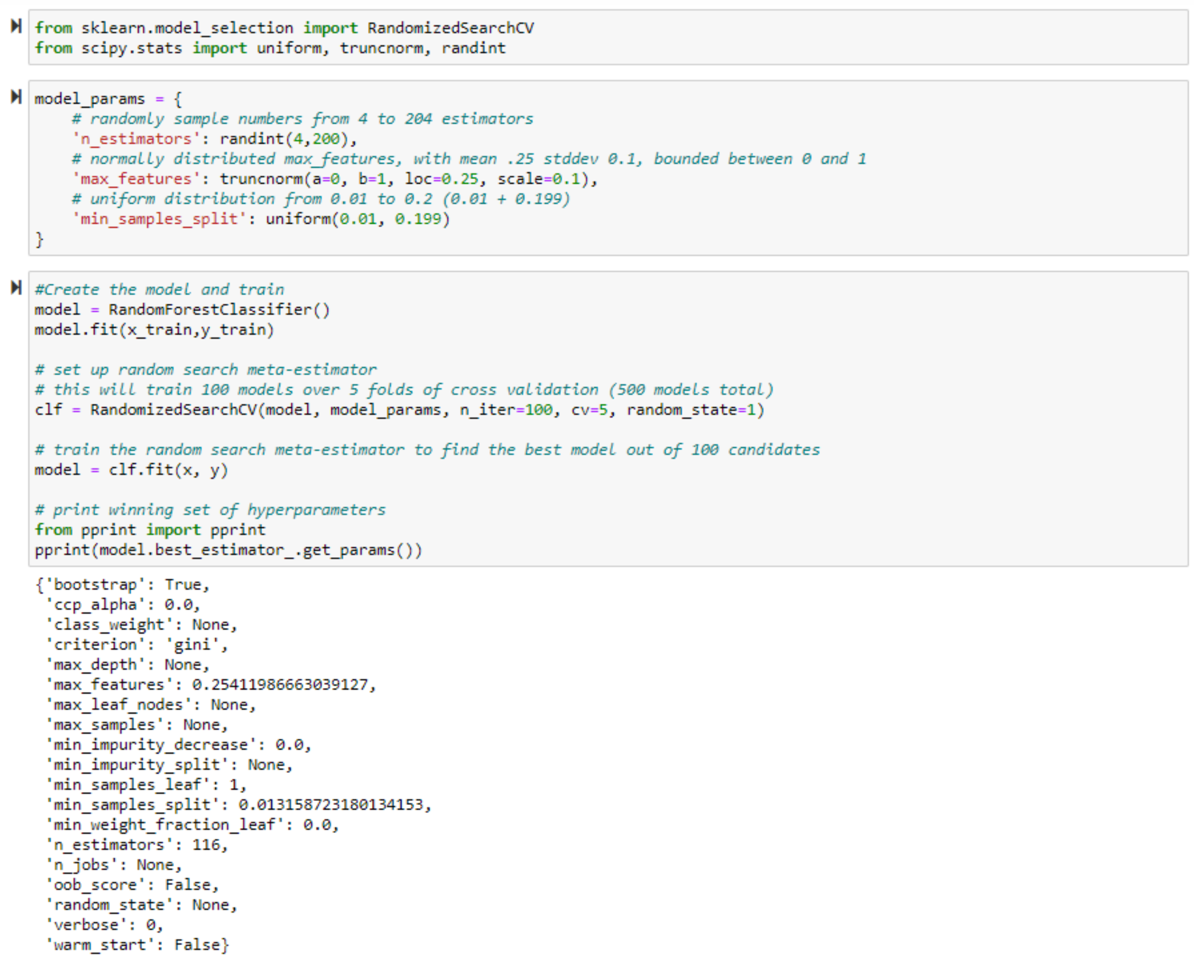


From the above visualization we can clearly see that the Micro Credit Project Dataset is highly imbalanced. Which we will balance it by using SMOTE.

HYPER-PARAMETER TUNING

As we found that Random Forest Classifier has the best accuracy and F1 score. So we will use this classifier algorithm for our Hyper parameter tuning.

We will be using RandomSearch CV instead of GridSearch CV due to memory and power problem of my laptop.



```python
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform, truncnorm, randint
```

```python
model_params = {
    # randomly sample numbers from 4 to 204 estimators
    'n_estimators': randint(4,200),
    # normally distributed max_features, with mean .25 stddev 0.1, bounded between 0 and 1
    'max_features': truncnorm(a=0, b=1, loc=0.25, scale=0.1),
    # uniform distribution from 0.01 to 0.2 (0.01 + 0.199)
    'min_samples_split': uniform(0.01, 0.199)
}
```

```python
#Create the model and train
model = RandomForestClassifier()
model.fit(x_train,y_train)

# set up random search meta-estimator
# this will train 100 models over 5 folds of cross validation (500 models total)
clf = RandomizedSearchCV(model, model_params, n_iter=100, cv=5, random_state=1)

# train the random search meta-estimator to find the best model out of 100 candidates
model = clf.fit(x, y)

# print winning set of hyperparameters
from pprint import pprint
pprint(model.best_estimator_.get_params())
```

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 0.25411986663039127,
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 0.013158723180134153,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 116,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

Above is the code and output for RandomizedSearch CV.

CONCLUSION

- Key Findings and Conclusions of the Study

  My Key Findings:

  - I found that it is a highly imbalanced set for which I used SMOTE technique to make it balanced.
  - There were many zero values in the column, I have to use if else statements in my code to remove skewness.
  - When using Machine learning algorithms I found that SVC took a lot of time as compared to other methods I used for Model building.
  - Random Forest Classifier Had the highest F1 and accuracy score in all of the models I used to predict.
  - Feature Selection was very important for these kind of Datasets as there were a multiple number of features that my laptop can handle while iterating.

- Learning Outcomes of the Study in respect of Data Science

The amount and complexity of information produced in science, engineering, business, and everyday human activity is increasing at staggering rates. Good visualizations not only present a visual interpretation of data, but do so by improving comprehension, communication, and decision making.The importance of visualization is a topic taught to almost every data scientist in an entry-level course at university but is mastered by very few individuals. It is often regarded as obvious or unimportant due to its inherently subjective nature. In this article, I hope to dispel some of those thoughts and show you that visualization is incredibly important, not just in the field of data science, but for communicating any form of information.

Visualisation through Seaborn and many other libraries helped me in gaining insight of the Dataset.

- It also helped me in gaining insight of outliers through boxplot method.
- It helped me in finding skewness of a particular variable with the help of seaborn.
- Univariate Analysis of the target variable showed us that it is a highly imbalanced Dataset.
- Multivariate Analysis showed us the relation between all the variables with each other.

Limitations of this work and Scope for Future Work

What are the limitations of this solution provided the future scope?

The most important task for any lender is to predict the probability of default for a borrower. An accurate prediction can help in balancing risk and return for the lender; charging higher rates for higher risks, or even denying the loan when required.

- There are some limitations of the solutions provided by me as I do not have access to powerful laptops, for that I have to use algorithms which consumes less power and memory.

The Future scope for this is infinite, telecom sectors can roll out options like giving loan to customer when doing recharge and also get to know who may default or not.

This also has a huge scope in Micro-Financial Banking sectors that provide loans only to poor and average households. By the above algorithms we can precisely measure whether a person will default his or her loan.