MICRO CREDIT DEFAULTER PROJECT

Submitted by:

Rupali Rane

ACKNOWLEDGMENT

I would like to express my gratitude to everyone who contributed to the successful completion of this project as well as the institute Flip Robo Technologies through which I got an opportunity to work on this project.

Then I'd like to express my gratitude to SME Mr. Mohd Kashif for providing me guidance on this project. The recommendations and guidance have aided in the completion of the project.

INTRODUCTION

Business Problem Framing

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

Many microfinance institutions (MFI), experts, and donors support the idea of using mobile financial services (MFS), which they believe are more convenient, efficient, and cost-effective than the traditional high-touch model that has been used for many years to deliver microfinance services. Though the MFI industry is primarily focused on low-income families and is extremely beneficial in these areas, MFS implementation has been uneven, with significant challenges and successes.

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

We are now working with a client in the telecom industry. They are a provider of fixed wireless telecommunications networks. They've released a number of products and built their business and organization around the budget operator model, which entails providing better products at lower prices to all value-conscious clients via a disruptive innovation strategy that focuses on the subscriber.

They've teamed up with a microfinance institution to offer micro-credit on mobile balances that must be paid back in five days. If the Consumer deviates from the course of repaying the loaned amount within the time period of 5 days, he is considered a defaulter. The payback amount for a loan of 5 (in Indonesian Rupiah) should be 6 (in Indonesian Rupiah), whereas the payback amount for a loan of 10 (in Indonesian Rupiah) should be 12. (in Indonesian Rupiah).

Build a model that can be used to predict if a client will pay back the lent amount within 5 days of loan insurance in terms of a probability for each loan transaction. Label '1' shows that the loan has been paid, indicating that it is a non-defaulter, whereas Label '0' indicates that the loan has not been paid, indicating that it is a defaulter.

Conceptual Background of the Domain Problem

Micro Credit is a technology and service package given in collaboration with telecom providers at the final stage of product delivery.

Some of the significant features of microfinance are as follows:

1. The borrowers are generally from low-income backgrounds
2. Loans availed under microfinance are usually of small amount, i.e., micro loans
3. The loan tenure is short
4. Microfinance loans do not require any collateral
5. These loans are usually repaid at higher frequencies
6. The purpose of most microfinance loans is income generation

Review of Literature

With the rapid growth of technology and increased competition, telecom firms are looking for ways to improve the quality of their service and, as a result, the health of their revenue.

Miniature credit arrangement furnishes administrators and specialist organizations with the capacity to stretch out their support of their clients through a little, transient credit office. At the point when we go through the dataset given, we should look at deliberately every one of the characteristics given, arrange the clients between defaulters and non-defaulters, and lessen the possibility of deceitfulness in miniature credit advances by clients.

Motivation for the Problem Undertaken

In this task, I need to construct an AI model which makes expectations to find deceitful clients in light of their past exercises which makes it simpler for specialist organizations and telecoms to give this office to their wholesalers, affiliates, and endorsers by limiting the credit risk for them. This takes the weight off the shoulders of the administrator who can zero in on working on the nature of administration being given to the clients.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

In this venture, we have utilized different inbuilt python techniques to check the insights of the information. To comprehend the different datatypes of the qualities.

I have utilized 'dtype' technique, to check in the event that there are any invalid qualities present in the dataset, I have utilized the 'IsNull().sum()' method. {It is likewise given in the dataset that there are no invalid qualities.

I have utilized the 'portray()' technique to comprehend the generally factual perspective on the information, The portray() technique returns a depiction of the information in the DataFrame. On the off chance that the DataFrame contains mathematical information, the portrayal contains this data for every section: count - The quantity of not-unfilled qualities. mean -The normal (mean) esteem. sexually transmitted disease - The standard deviation. min-The base esteem. The max-The greatest worth of property.

The data() technique prints data about the DataFrame. The data contains the number of segments, section marks, section information types, memory utilization, range file, and the number of cells in every section (non-invalid qualities).

Note: the info() technique really prints the info.

```
df = pd.read_csv('Data file.csv')
df
```

	Unnamed: 0	label	msisdn	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_cate_ma	last_rech_date_da	...	maxamnt_loans30
0	1	0	21408I70789	272.0	3055.050000	3065.150000	220.13	260.13	2.0	0.0	...	6.0
1	2	1	76462I70374	712.0	12122.000000	12124.750000	3691.26	3691.26	20.0	0.0	...	12.0
2	3	1	17943I70372	535.0	1398.000000	1398.000000	900.13	900.13	3.0	0.0	...	6.0
3	4	1	55773I70781	241.0	21.228000	21.228000	159.42	159.42	41.0	0.0	...	6.0
4	5	1	03813I82730	947.0	150.619333	150.619333	1098.90	1098.90	4.0	0.0	...	6.0
...	...	...	...	...	...	...	...	...	...	...	...	...
209588	209589	1	22758I85348	404.0	151.872333	151.872333	1089.19	1089.19	1.0	0.0	...	6.0

```
#Let us check the statistical summary of the dataframe.
df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	209593.0	104797.000000	60504.431823	1.000000	52399.000	104797.000000	157195.00	209593.000000
label	209593.0	0.875177	0.330519	0.000000	1.000	1.000000	1.00	1.000000
aon	209593.0	8112.343445	75696.082531	-48.000000	246.000	527.000000	982.00	999860.755168
daily_decr30	209593.0	5381.402289	9220.623400	-93.012667	42.440	1469.175667	7244.00	265926.000000
daily_decr90	209593.0	6082.515068	10918.812767	-93.012667	42.692	1500.000000	7802.79	320630.000000
rental30	209593.0	2692.581910	4308.586781	-23737.140000	280.420	1083.570000	3356.94	198926.110000
rental90	209593.0	3483.406534	5770.461279	-24720.580000	300.260	1334.000000	4201.79	200148.110000
last_rech_date_ma	209593.0	3755.847800	53905.892230	-29.000000	1.000	3.000000	7.00	998650.377733
last_rech_date_da	209593.0	3712.202921	53374.833430	-29.000000	0.000	0.000000	0.00	999171.809410
last_rech_amt_ma	209593.0	2064.452797	2370.786034	0.000000	770.000	1539.000000	2309.00	55000.000000
cnt_ma_rech30	209593.0	3.978057	4.256090	0.000000	1.000	3.000000	5.00	203.000000
fr_ma_rech30	209593.0	3737.365121	53643.625172	0.000000	0.000	2.000000	6.00	999606.368132

Data Sources and their formats

The example information is given to Flip Robo from their client data set. It is thus given to me for this activity. The information given is in the structure CSV record, I'll be changing it into DataFrame completely to perform the essential procedure on lines/segments like choosing, erasing, adding, and renaming. To work on the choice of clients for the credit, the client needs a few expectations that could end up being useful to them in additional venture and improvement in the choice of clients.

There are 209593 rows and 37 columns in the dataset provided. All the attributes are numerical datatypes except ‘pCircle’ and ‘pdate’.

```
#checking the different coloumn names available in the dataset.
df.columns
```

```
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')
```

```
#checking the overview information of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209593 entries, 0 to 209592
Data columns (total 37 columns):
 #    Column                Non-Null Count    Dtype
---   ------                --------------    -----
 0    Unnamed: 0            209593 non-null   int64
 1    label                 209593 non-null   int64
 2    msisdn                209593 non-null   object
 3    aon                   209593 non-null   float64
 4    daily_decr30          209593 non-null   float64
 5    daily_decr90          209593 non-null   float64
 6    rental30              209593 non-null   float64
 7    rental90              209593 non-null   float64
 8    last_rech_date_ma     209593 non-null   float64
 9    last_rech_date_da     209593 non-null   float64
 10   last_rech_amt_ma      209593 non-null   int64
 11   cnt_ma_rech30         209593 non-null   int64
 12   fr_ma_rech30          209593 non-null   float64
 13   sumamnt_ma_rech30     209593 non-null   float64
```

Data Pre-processing Done

There are outliers present in the dataset. We can use the IQR method of identifying outliers to set up a “fence” outside of Q1 and Q3. Any values that fall outside of this fence are considered outliers. ... Any observations that are more than 1.5 IQR below Q1 or more than 1.5 IQR above Q3 are considered outliers.

IQR = Q3-Q1

high = Q3+(1.5*IQR)
low = Q1-(1.5*IQR)

The following attributes in the list are having extreme outliers, let us treat them with the below technique. We will replace the higher outlier values with upper boundaries, and lower outlier values with lower boundaries.

We can see that there is reduction in outliers. We can see some of features like 'fr_da_rech30','fr_da_rech90', 'last_rech_date_da', 'medianmarechprebal30’ have nearly zero correlation with target variable.
‘cnt_da_rech30’,’cnt_loans90’,’fr_da_rech90’,’medianmarechprebal90’ also have very high skewness in the data.

From the heatmap we observe that, 'amnt_loans30'&'cnt_loans90', 'daily_decr30 & daily_decr90' have strong correlation. We can remove one of the attributes to reduce multicollinearity.

We can understand that these values have less importance towards the model inference. We will proceed by dropping these columns from the dataset

Data Inputs- Logic- Output Relationships

There are 87.5% of non-defaulters and 12.5% of defaulter customers, the data is unbalanced, the target variable should be balanced before feeding the data to model.
After the removal of outliers, there is not much skewness present in the data, as we can see all the values of skewness are less than 10. There is only 1 value 'maxamnt_loans30' with a higher value, we can proceed by dropping this attribute.
The correlation graph shows the attributes are having positive and negative correlations. The attributes with values near zero are not contributing to the model prediction. We can drop them using the feature engineering technique.

The following images show skewness and correlation of the data after the outlier removal.

```
#cheking the skewness present in the dataset
df.skew()
```

```
label                    -2.270254
aon                       0.859469
daily_decr30              1.128563
daily_decr90              1.132084
rental30                  1.077084
rental90                  1.078299
last_rech_date_ma         0.944700
last_rech_date_da         0.000000
last_rech_amt_ma          0.881053
cnt_ma_rech30             0.774796
fr_ma_rech30              1.133385
sumamnt_ma_rech30         0.946429
medianamnt_ma_rech30      0.569947
medianmarechprebal30      0.899169
cnt_ma_rech90             0.810947
fr_ma_rech90              1.070414
sumamnt_ma_rech90         0.997038
medianamnt_ma_rech90      0.604827
medianmarechprebal90      0.864082
cnt_da_rech30             0.000000
fr_da_rech30              0.000000
cnt_da_rech90             0.000000
fr_da_rech90              0.000000
cnt_loans30               1.211929
amnt_loans30              1.102369
maxamnt_loans30          17.658052
medianamnt_loans30        4.551043
cnt_loans90               1.154202
amnt_loans90              1.091910
maxamnt_loans90           1.678304
medianamnt_loans90        4.895720
payback30                 1.064092
payback90                 1.040428
dtype: float64
```

Set of assumptions related to the problem under consideration

1. We assume that the dataset provided is a completely random sample that is representative of the population.
2. There is minimal or no multicollinearity among the independent variables.

Hardware and Software Requirements and Tools Used

Following are the recommended hardware required to build and run Machine-learning model.

i. 7th generation (Intel Core i7 processor)
ii. 8GB RAM / 16 GB RAM (recommended)

We have used the following software and tools for the machine learning model.

iii. ANACONDA

Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.

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

%matplotlib inline

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

```
from imblearn.over_sampling import RandomOverSampler
from scipy.stats import zscore

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import cross_val_score,GridSearchCV, train_test_split

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC

from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,plot_confusion_matrix, plot_roc_curve
```

The figure shows the important libraries I have imported to execute the project. I have used built-in Data science libraries like pandas, NumPy, and Visualization libraries like matplotlib and seaborn. Jupyter Notebook is a shareable notebook that combines live code, visualizations, and text.

Machine learning libraries like scikit-learn for data pre-processing, model selection, model evaluation, and SciPy for standardizing& normalizing the data.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

The dataset provided has a huge volume of the data which did not have any null values, but there were outliers present in the dataset, unless the outlier treatment there is a possibility of our machine learning model overfitting the data or increasing the variability in the data. Keeping the data loss into concern, Inter-
quartile range method is implemented to reduce the outliers.
The attributes which are having less correlation with the target variable have been dropped and removed based on the inference learned from heatmaps and bar plots.
I have treated the target variable with a random over-sampling technique to equalize the class variables into independent variables.

Testing of Identified Approaches (Algorithms)

To feed the dataset to the model, the independent and dependent variables are to be split and the independent attributes are standardized using 'StandardScaler' library.
Now the data obtained is clean and is having least multicollinearity, independent and balanced data. 'train_test_split' function in model selection is used for splitting data arrays into two subsets: for training data and for testing data. We train the model using the training set and then apply the model to the test set.

Let us seperate the independent and dependent variables.

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

```
scalar = StandardScaler()
X_scaler = scalar.fit_transform(X)
```

```
The max accuracy obtained is 0.7761023158445637 for the Random State 52
```

```
#splitting the dataset

X_train, X_test, y_train, y_test = train_test_split(X_scaler, y, random_state=52, test_size=0.25)
```

I have chosen 5 classification machine learning algorithms which is suitable for the pre-processed and cleaned data to train and test the dataset.

A. Logistic Regression

Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables.

B. Random Forest Classifier

The random forest classifier is a versatile classification tool that makes an aggregated prediction using a group of decision trees trained using the bootstrap method with extra randomness while growing trees by searching for the best features among a randomly selected feature subset.

C. Decision Tree Classifier

A decision tree is a class discriminator that recursively partitions the training set until each partition consists entirely or dominantly of examples from one class.

D. XGBoost Classifier

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance that is dominative competitive machine learning.

E. KNN Classifier

K Nearest Neighbors(KNN) is a very simple, easy-to-understand, versatile, and one of the topmost machine learning algorithms. KNN is used in a variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition

Run and Evaluate selected models

1. Logistic Regression

14]:

```
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
y_pred_train = log_reg.predict(X_train)
y_pred = log_reg.predict(X_test)

print("******************RESULTS******************")
print("The accuracy score of train is :", accuracy_score(y_train, y_pred_train)*100)
print("The accuracy score test is :", accuracy_score(y_test, y_pred)*100)
cv_score = cross_val_score(log_reg,X_train, y_train,cv=5)
print("The cross validation score is :", cv_score.mean()*100)

print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
print("Classification \n", classification_report(y_test, y_pred))
print("***********************************************")

plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,fmt = "d",linecolor="r",linewidths=1)
plt.show()
```

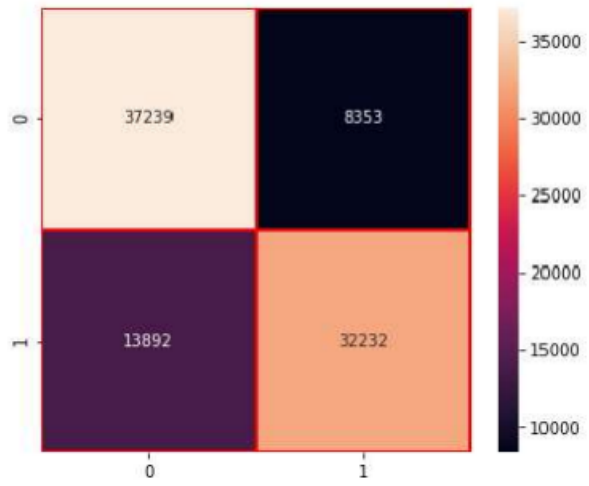
Results and confusion matrix graph

```
******************RESULTS******************
The accuracy score of train is : 75.61403763819936
The accuracy score test is : 75.74578045270182
The cross validation score is : 75.59986399651228
Confusion Matrix:
 [[37239  8353]
 [13892 32232]]
Classification
              precision    recall  f1-score   support

           0       0.73      0.82      0.77     45592
           1       0.79      0.70      0.74     46124

    accuracy                           0.76     91716
   macro avg       0.76      0.76      0.76     91716
weighted avg       0.76      0.76      0.76     91716

***********************************************
```



2. Random Forest Classifier

n [15]:

```
ran_clf = RandomForestClassifier()
ran_clf.fit(X_train, y_train)
y_pred_train = ran_clf.predict(X_train)
y_pred = ran_clf.predict(X_test)

print("******************RESULTS******************")
print("The accuracy score of train is :", accuracy_score(y_train, y_pred_train)*100)
print("The accuracy score test is :", accuracy_score(y_test, y_pred)*100)
cv_score = cross_val_score(ran_clf,X_train, y_train,cv=5)
print("The cross validation score is :", cv_score.mean()*100)

print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
print("Classification\n ", classification_report(y_test, y_pred))
print("***********************************************")

plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,fmt = "d",linecolor="r",linewidths=1)

plt.show()
```

Results and confusion matrix graph

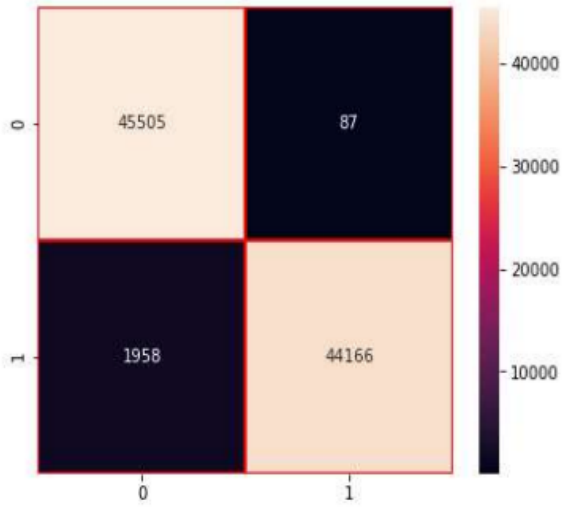
Results and Confusion matrix graph:

```
******************RESULTS******************
The accuracy score of train is : 99.97419551801589
The accuracy score test is : 97.77029089798944
The cross validation score is : 96.99868459492029
Confusion Matrix:
 [[45505    87]
 [ 1958 44166]]
Classification
                precision    recall  f1-score   support

           0       0.96      1.00      0.98     45592
           1       1.00      0.96      0.98     46124

    accuracy                           0.98     91716
   macro avg       0.98      0.98      0.98     91716
weighted avg       0.98      0.98      0.98     91716

***********************************************
```



3. Decision Tree Classifier

```
dec_clf = DecisionTreeClassifier()
dec_clf.fit(X_train, y_train)
y_pred_train = dec_clf.predict(X_train)
y_pred = dec_clf.predict(X_test)

print("********************RESULTS********************")
print("The accuracy score of train is :", accuracy_score(y_train, y_pred_train)*100)
print("The accuracy score test  is :", accuracy_score(y_test, y_pred)*100)
cv_score = cross_val_score(dec_clf,X_train, y_train,cv=5)
print("The cross validation score is :", cv_score.mean()*100)

print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
print("Classification \n", classification_report(y_test, y_pred))
print("***********************************************")

plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,fmt = "d",linecolor="r",linewidths=1)
plt.show()
```

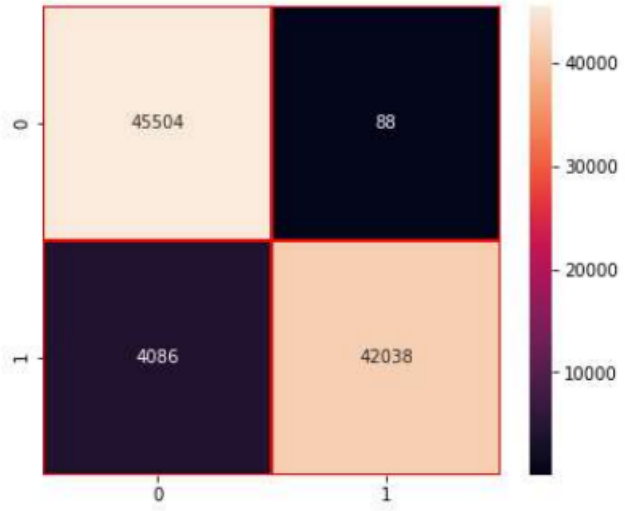
Results and confusion matrix graph

```
********************RESULTS********************
The accuracy score of train is : 99.97492240483234
The accuracy score test  is : 95.44899472284007
The cross validation score is : 94.31974294082005
Confusion Matrix:
 [[45504    88]
 [ 4086 42038]]
Classification
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     45592
           1       1.00      0.91      0.95     46124

    accuracy                           0.95     91716
   macro avg       0.96      0.95      0.95     91716
weighted avg       0.96      0.95      0.95     91716

***********************************************
```



4. XGBoost Classifier

```
]: xgb_clf = XGBClassifier()
   xgb_clf.fit(X_train, y_train)
   y_pred_train = xgb_clf.predict(X_train)
   y_pred = xgb_clf.predict(X_test)

   print("********************RESULTS********************")
   print("The accuracy score of train is :", accuracy_score(y_train, y_pred_train)*100)
   print("The accuracy score test is :", accuracy_score(y_test, y_pred)*100)
   cv_score = cross_val_score(xgb_clf,X_train, y_train,cv=5)
   print("The cross validation score is :", cv_score.mean()*100)

   print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
   print("Classification \n ", classification_report(y_test, y_pred))
   print("************************************************")

   plt.figure(figsize=(6,5))
   sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,fmt = "d",linecolor="r",linewidths=1)
   plt.show()
```

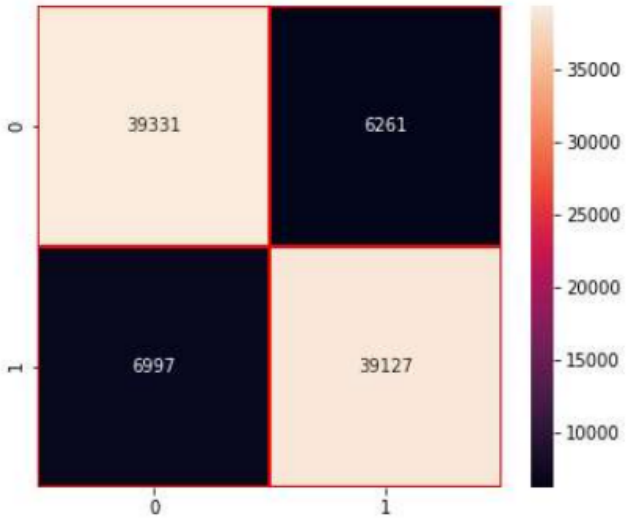
Results and confusion matrix graph

```
********************RESULTS********************
The accuracy score of train is : 86.11464458869109
The accuracy score test is : 85.54450695625627
The cross validation score is : 85.22711541913208
Confusion Matrix:
 [[39331  6261]
 [ 6997 39127]]
Classification
                precision    recall  f1-score   support

           0       0.85      0.86      0.86     45592
           1       0.86      0.85      0.86     46124

    accuracy                           0.86     91716
   macro avg       0.86      0.86      0.86     91716
weighted avg       0.86      0.86      0.86     91716

************************************************
```



5. K-Neighbours Classifier

```
knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)
y_pred_train = knn_clf.predict(X_train)
y_pred = knn_clf.predict(X_test)

print("********************RESULTS********************")
print("The accuracy score of train is :", accuracy_score(y_train, y_pred_train)*100)
print("The accuracy score test is :", accuracy_score(y_test, y_pred)*100)
cv_score = cross_val_score(knn_clf,X_train, y_train,cv=5)
print("The cross validation score is :", cv_score.mean()*100)

print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
print("Classification ", classification_report(y_test, y_pred))
print("***********************************************")

plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,fmt = "d",linecolor="r",linewidths=1)
plt.show()
```

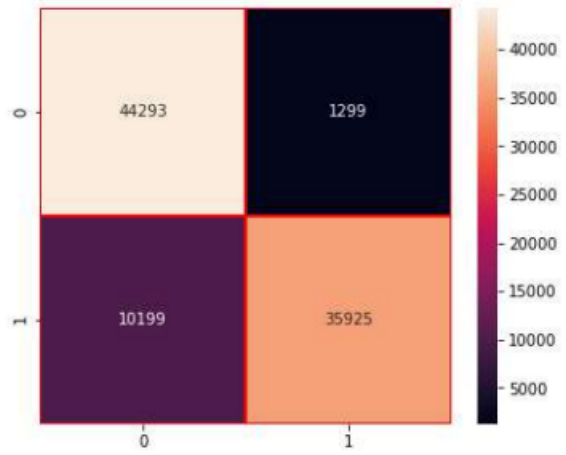
Results and confusion matrix graph

```
********************RESULTS********************
The accuracy score of train is : 91.09236550776679
The accuracy score test is : 87.4634742029744
The cross validation score is : 85.1700548146046
Confusion Matrix:
 [[44293  1299]
 [10199 35925]]
Classification                precision    recall  f1-score   support

           0       0.81      0.97      0.89     45592
           1       0.97      0.78      0.86     46124

    accuracy                           0.87     91716
   macro avg       0.89      0.88      0.87     91716
weighted avg       0.89      0.87      0.87     91716

***********************************************
```



Key Metrics for success in the solving problem under consideration

A key/evaluation metric quantifies the performance of a predictive model This typically. Following are the metrics I have used to evaluate the model performance.

1. Confusion Matrix:
   The confusion matrix provides a more insightful picture based on the counts of test records correctly and incorrectly predicted by the model, and what type of errors are being made. The confusion matrix is useful for measuring recall

(also known as Sensitivity), Precision, Specificity, Accuracy, and, most importantly, the AUC-ROC Curve.

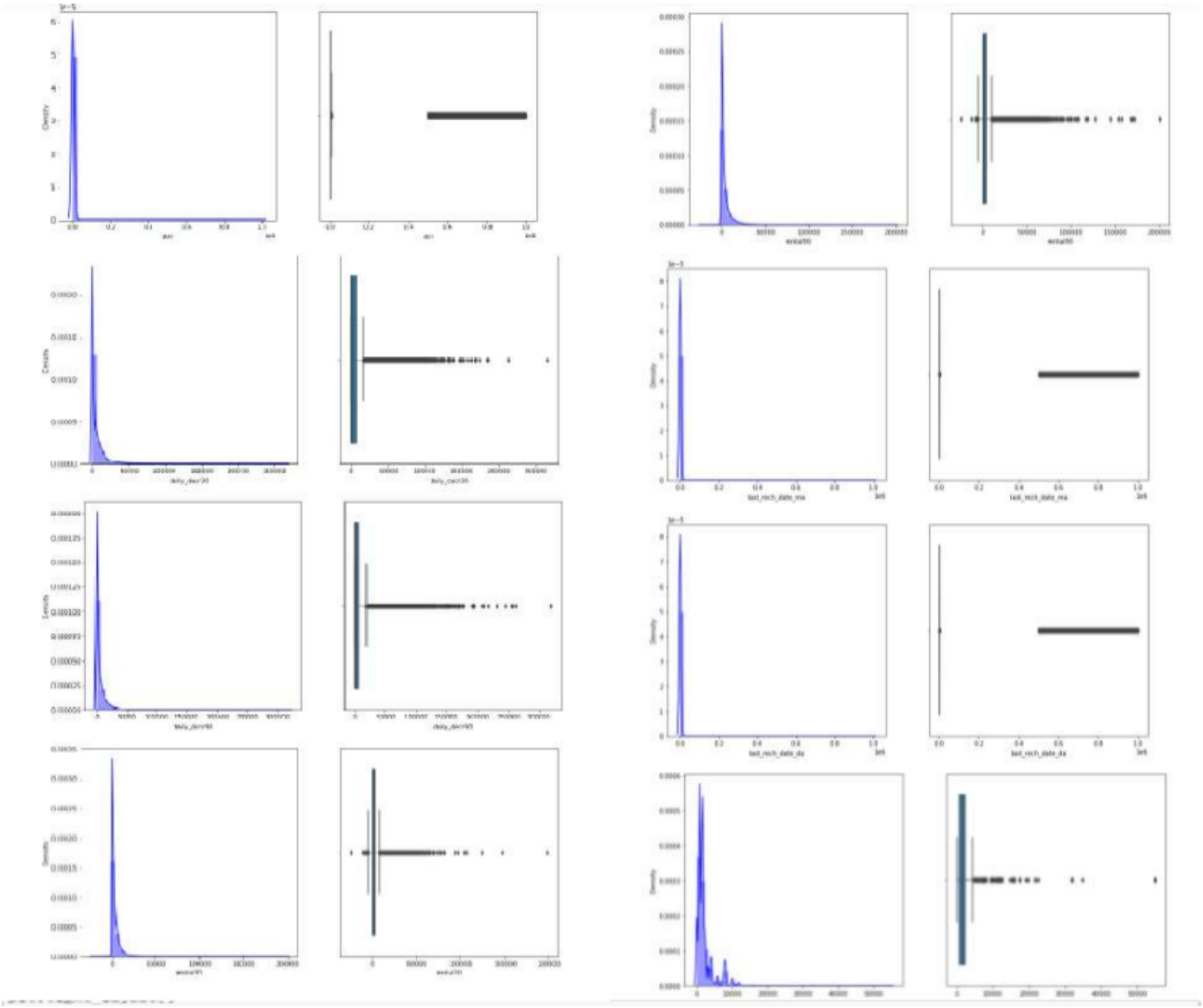
2. Sensitivity:
   It measures how many observations out of all positive observations have we classified as positive. It tells us how many fraudulent transactions we recalled from all fraudulent transactions.
3. Precision:
   It measures how many observations predicted as positive are in fact positive. Taking our fraud detection example, tells us what ratio of transactions is correctly classified as fraudulent.
4. Accuracy:
   It measures how many observations, both positive and negative, were correctly
   classified.
5. F1 Score:
   A good F1 score means that we have low false positives and low false negatives, so we're correctly identifying real threats, and we are not disturbed by false alarms.
6. Cross-Validation Score:
   It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

Visualizations

We have used matplotlib and seaborn to interpret the relationship, we have plotted the graph using the histogram to know how the data is distributed and the box plot is used to check the outliers present and how is variance spread around the mean of the data.

We can see from the below graph that there are huge number of outliers present in the data set which impacts the performance of the data.

We have tried to treat outliers with some imputation methods which results in



Interpretation of the Results

We have trained several models above for the dataset we had prepared, and we got different results for the different algorithms.

The logistic regression model gave us 75.7% of accuracy and a cross validation score of 75.6 % for the test model, **Random Forest classifier** model gave us 97.7% accuracy and 96.99%. **The Decision tree classifier model** gave us accuracy and cv scores of 95.5% and 94.3 %. **XGboost classifier** has given us 85.9% and 85.3% of accuracy and cv score for the test dataset. **KNN classifier** has given us

87.4% and 85.17% of accuracy and cv score for the test dataset.

The code prints out the number of false positives it detected and compares it with the actual values. This is used to calculate the accuracy score and precision of the algorithms. These results along with the classification report for each algorithm are given in the output as follows, where class 0 means the transaction was determined to be valid and 1 means it was determined as a fraud transaction. This result matched the class values to check for false positives.

All the above models have given us the best results for the model prepared, letus's check if we can improvise our model performance to 100% accuracy.

CONCLUSION

Key Findings and Conclusions of the Study

We see that the Random Forest classifier model has given the highest AUC in the graph, an accuracy score of 97%, and the CV score of 96% which is the highest among all the models tested also, we see that evaluation metrics are high for this model. Hence, we will be saving this model.

Learning Outcomes of the Study in respect of Data Science

1. Data Exploration and Cleaning, on data exploration, I found that the dataset was imbalanced for the target feature (87.5% for non-defaulters and 12.5% for Defaulters). Also, I found that the data had some very unrealistic values such as 999860 days which is not possible. Also, there were negative values for variables that must not have one (for example, frequency, amount of recharge, etc). All these unrealistic values were imputed which caused data to stabilize.
2. Feature Selection: there were 36 features, many of which I suspected were redundant because of the data duplication. It was imperative to select only the most significant of them to make ML models more efficient and cost-effective. The method used was 'Univariate Selection' using a chi-square test. I selected the top 20 features which were highly significant.

3. Data Visualization: On visualizing data, there were two important insights I gathered.
   a. Imbalance of data

b. Distribution was not normal

4. Data Standardization: Since the data was not normal, all the features except the target variable which was dichotomous (Values ‘1’ and ‘0’).

5. Oversampling of Minority Class: The data was expensive, I did not want to lose out on data by under-sampling the majority class. Instead, I decided to oversample the minority class using Random Oversampling.

6. Build Models: It was a supervised classification problem, I built 5 models to evaluate the performance of each of them:

a. Logistic Regression

b. Random Forest Classifier

c. Decision Tree Classifier

d. XGBoost Classifier

e. KNN Classifier

The data was imbalanced, accuracy was not the correct performance metric. Instead, I focused on other metrics like precision, recall and ROC-AUC Curve.

Limitations of this work and Scope for Future Work

The data set consists of a large number of outliers which hinders the performance of machine learning models. Unless we solve the outlier problems, we are not reaching the best model accuracy. One can focus on the collection of real time customer-oriented data which can be useful for EDA. And more inferences can be provided based on the analysis.