BANK FRAUD DETECTION REPORT

Author: Kolawole Joseph. E

Introduction

This project work helps to detect Bank fraud by using given data. Supervised Machine learning algorithms was used to detect fraudulent behavior based upon past fraud as indicated in the "label" column, and the use of unsupervised Machine learning algorithms was employed to discover new types of fraud activities. Fraudulent transactions are rare compared to the norm. Hence, imbalanced datasets was properly classified. The project work provides technical insights and demonstrates the implementation of fraud detection models.

Data Description

The dataset consists of 37 columns and 190001 rows with the "label" column as its target column.

COLUMN	DESCRIPTION
label	Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan {1:success, 0:failure}
msisdn	mobile number of user
aon	age on cellular network in days
daily_decr30	Daily amount spent from main account, averaged over last 30 days
daily_decr90	Daily amount spent from main account, averaged over last 90 days
rental30	Average main account balance over last 30 days
rental90	Average main account balance over last 90 days
last_rech_date_ma	Number of days till last recharge of main account
last_rech_date_da	Number of days till last recharge of data account
last_rech_amt_ma	Amount of last recharge of main account
cnt_ma_rech30	Number of times main account got recharged in last 30 days
fr_ma_rech30	Frequency of main account recharged in last 30 days
sumamnt_ma_rech30	Total amount of recharge in main account over last 30 days
medianamnt_ma_rech30	Median of amount of recharges done in main account over last 30 days at user level
medianmarechprebal30	Median of main account balance just before recharge in last 30 days at user level
cnt_ma_rech90	Number of times main account got recharged in last 90 days
fr_ma_rech90	Frequency of main account recharged in last 90 days
sumamnt_ma_rech90	Total amount of recharge in main account over last 90 days
medianamnt_ma_rech90	Median of amount of recharges done in main account over last 90 days at user level
medianmarechprebal90	Median of main account balance just before recharge in last 90 days at user level
cnt_da_rech30	Number of times data account got recharged in last 30 days
fr_da_rech30	Frequency of data account recharged in last 30 days
cnt_da_rech90	Number of times data account got recharged in last 90 days
fr_da_rech90	Frequency of data account recharged in last 90 days
cnt_loans30	Number of loans taken by user in last 30 days
amnt_loans30	Total amount of loans taken by user in last 30 days
maxamnt_loans30	maximum amount of loan taken by the user in last 30 days There are only two options: 5 & 10 INR, for which the user needs to pay back 6 & 12 INR respectively
medianamnt_loans30	Median of amounts of loan taken by the user in last 30 days

cnt_loans90	Number of loans taken by user in last 90 days
amnt_loans90	Total amount of loans taken by user in last 90 days
maxamnt_loans90	maximum amount of loan taken by the user in last 90 days
medianamnt_loans90	Median of amounts of loan taken by the user in last 90 days
payback30	Average payback time in days over last 30 days
payback90	Average payback time in days over last 90 days
pcircle	telecom circle
pdate	date

Approach

For this project work, all necessary Python libraries, modules and dataset were first imported, and then followed by data visualization. Preprocessing and EDA was carried out before the models were built. Evaluation metrics as well such as precision, recall, and f1 score for all the models were also done. The whole code was written in both Jupyter Notebook and Google Colab

Code and Visualization

The complete code can be found in the link below. However, some of the code snippets and visuals are shown below:
https://drive.google.com/file/d/1OVEnmF_YMMmFbgoMZA9Fb_TUERfCzqIk/view?usp=sharing

- The Code:

```
# IMPORTING NECESSARY LIBRARIES AND MODULES

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

```
# Determine number of fraud cases in dataset
Fraud = df[df['label'] == 0]
Valid = df[df['label'] == 1]
outlier_fraction = len(Fraud)/float(len(Valid))
print('Outlier frction: {}'.format(outlier_fraction))
print('Fraud Cases: {}'.format(len(df[df['label'] == 1])))
print('Valid Transactions: {}'.format(len(df[df['label'] == 0])))
```

```
#separating training dataset from test dataset with a test size of 20%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
ate=None)
# Printing the shape of the splitted dataset
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

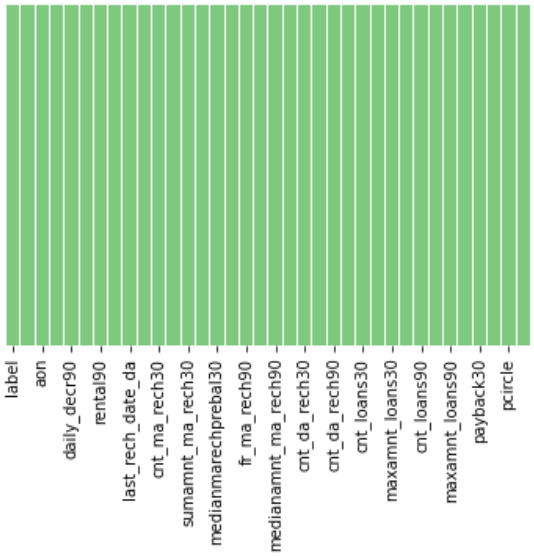
```
# Define the model as the random forest
rf_model = RandomForestClassifier(random_state=5, n_estimators=100)
# Fit the model to the training set
rf_model.fit(X_train, y_train)
print(rf_model.score(X_train, y_train))
rf_model_pred = rf_model.predict(X_test)
print(cl('ACCURACY SCORE', attrs = ['bold']))
print(cl('------------------------------------------------------------------------
', attrs = ['bold']))
print(cl('Accuracy score of the Random Forest Tree model is: {}'.format(accuracy_s
core(y_test, rf_model_pred)), attrs = ['bold'], color = 'red'))
print(cl('F1 SCORE', attrs = ['bold']))
print(cl('------------------------------------------------------------------------
', attrs = ['bold']))
print(cl('F1 score of the Random Forest Tree model is: {}'.format(f1_score(y_test,
 rf_model_pred)), attrs = ['bold'], color = 'blue'))
print(cl('R^2 SCORE', attrs = ['bold']))
print(cl('------------------------------------------------------------------------
', attrs = ['bold']))
print(cl('R^2 score of the Random Forest Tree model is: {}'.format(r2_score(y_test
, rf_model_pred)), attrs = ['bold'], color = 'green'))
```

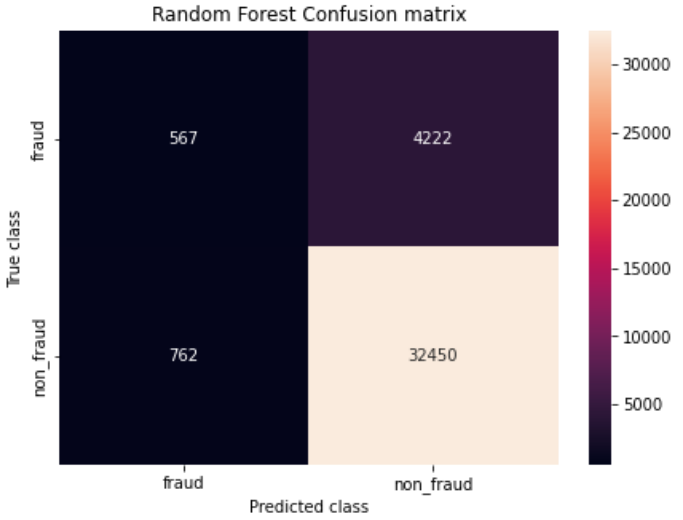
```
#Visualizing the Random Forest confusion matrix
LABELS = ['fraud', 'non_fraud']
plt.figure(figsize=(7, 5))
sns.heatmap(rf_conf_mat, xticklabels=LABELS,
 yticklabels=LABELS, annot=True, fmt="d");
plt.title("Random Forest Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
```

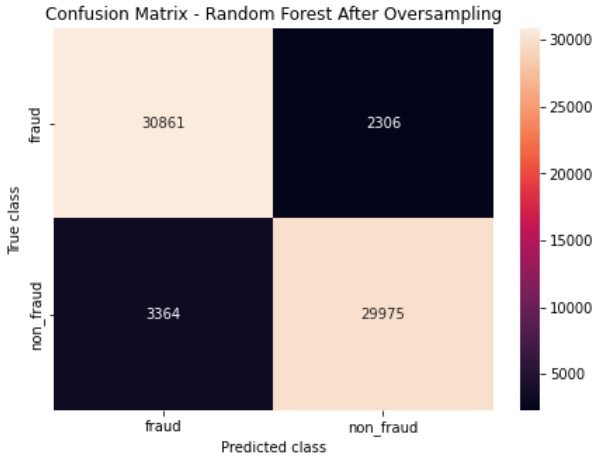
```
# Applying SMOTE to oversample the minor dataset
X_resampled, y_resampled = SMOTE().fit_resample(X, y)
print("Resampled shape of X: ", X_resampled.shape)
print("Resampled shape of Y: ", y_resampled.shape)
value_counts = Counter(y_resampled)
print(value_counts)
(train_X, test_X, train_y, test_y) = train_test_split(X_resampled, y_resampled, te
st_size= 0.2, random_state= 42)
print("Resampled shape of X_train: ", train_X.shape)
print("Resampled shape of X_test: ", test_X.shape)
print("Resampled shape of y_train: ", train_y.shape)
print("Resampled shape of y_test: ", test_X.shape)
```

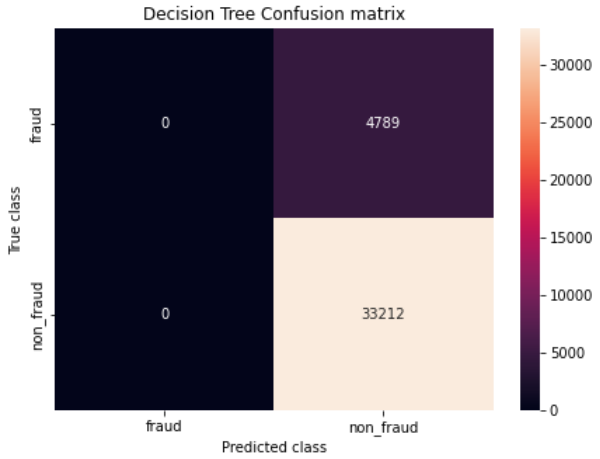
➢ Visualization

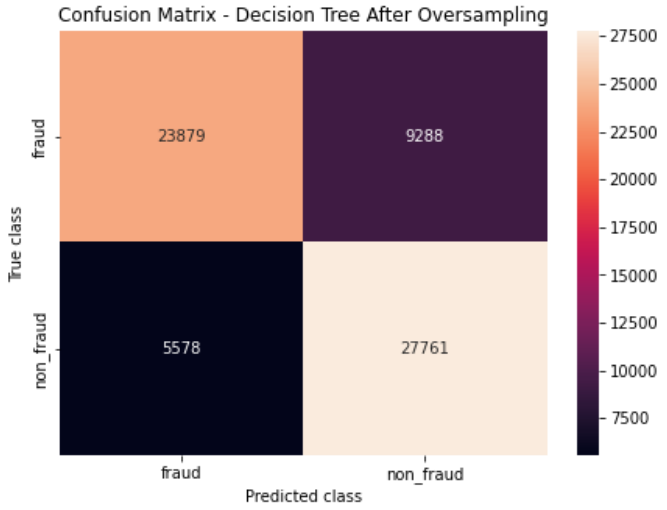
Heatmap showing null values

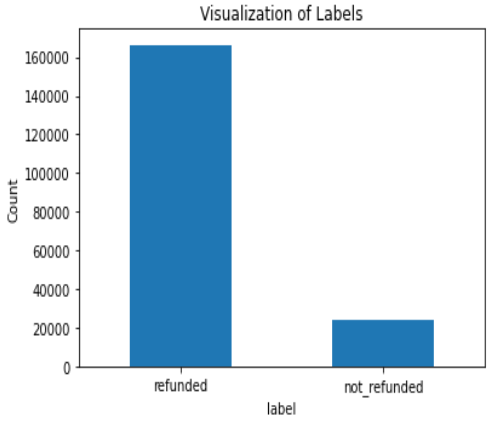












Algorithms

As I mentioned earlier, I have used various classification models on this dataset and they have different accuracy and other performance measures. The following machine learning algorithms are used on the dataset:

1. Random Forest
2. Decision Tree
3. Logistic Regression
4. K-Nearest Neighbor
5. SVM
6. XGBoost

Evaluation

The built models were evaluated using the evaluation metrics provided by the scikit-learn package. The main objective in this process is to find the best model for the given case. The evaluation metrics used are the accuracy score metric, f1 score metric, and finally the confusion matrix.

Result and discussion

The Random Forest classifier has slightly an edge over the Decision Tree classifier. But, taking a closer look at the dataset, it can be observed that the suffers a serious problem of class imbalance. The genuine (not fraud) transactions are more than 88% with the fraud transactions constituting 12%. Hence, the need to take care of the imbalance issues, since the models predict the label with higher importance given to genuine transactions (as there is more data about them) and hence obtains more accuracy. The class imbalance problem was solved by SMOTE technique. This improves the Random Forest Model accuracy from the initial 87% to 92% accuracy. This makes the Random Forest model perform better than Decision Trees and the other machine learning algorithms used.

Conclusion

After resampling, visualizations and machine learning algorithms had been done, it is clearly evident that the model performed much better than the previous Random Forest classifier without oversampling.

Difficulty faced

One of the difficulties faced was the inability to convert Object dtype to Float or Int type. However, after several trials and errors, Alternative method was taken by dropping the affected columns.

Future Work

Given more time and resources, other Machine learning models would be built and tested on the dataset with proper dash boarding and deployment using Streamlit, Flask/Django, and AWS /Azure.

References

1. SKillVertex – *Bank Fraud detection Dataset and other Data Science study materials*
2. *https://trenton3983.github.io/files/projects/2019-07-19_fraud_detection_python/2019-07-19_fraud_detection_python.html#Undersampling*
3. *https://medium.com/codex/credit-card-fraud-detection-with-machine-learning-in-python-ac7281991d87*
4. https://data-flair.training/blogs/credit-card-fraud-detection-python-machine-learning/