MINOR PROJECT 2 - REPORT

CREDIT CARD FRAUD DETECTION

Submitted By

Name	Roll No	Branch	Sap ID
Dhruv Maheshwari	R200219018	BCA-BFSI	500074514
Dhruv Kundu	R200219017	BCA-IOT	500074515
Vishal Singh	R200219051	BCA-BFSI	500074506

Under the guidance of

DR. SATENDRA KUMAR

School of Computer Science.

UPES

UNIVERSITY WITH A PURPOSE

SCHOOL OF COMPUTER SCIENCE

UNIVERSITY OF PETROLEUM & ENERGY STUDIES

Bidholi Campus, Energy Acres, Dehradun – 248007.

April - 2022

School of Computer Science

University of Petroleum & Energy Studies, Dehradun

Minor II

PROJECT TITLE: CREDIT CARD FRAUD DETECTION

Index

Abstract

In today's digital era credit cards have become very common. Reason being credit cards can be valuable tools for earning rewards, traveling, handling emergencies or unplanned expenses, and building credit. Many companies such as **Slice, CRED, Uni-card** have introduced many credit card schemes to lure consumers into using credit cards. Because of the increase in credit card users, the number of transactions have also increased at a rapid rate. Due to which it has been very difficult to detect frauds in the transactions. With the advancement in Machine Learning and data mining techniques, we can now train a model with the available data to perform future detection of frauds in credit card transactions.

Introduction

In today's digital era credit cards have become very common. Reason being credit cards can be valuable tools for earning rewards, traveling, handling emergencies or unplanned expenses, and building credit. Many companies such as **Slice, CRED, Uni-card** have introduced many credit card schemes to lure consumers into using credit cards. Because of the increase in credit card users, the number of transactions have also increased at a rapid rate. For any bank or financial organization, credit card fraud detection is of utmost importance. Various frauds occurs due to -

- Firstly and most ostensibly when your card details are overseen by some other person.
- When your card is lost or stolen and the person possessing it knows how to get things done.
- Fake phone call convincing you to share the details.
- And lastly and most importantly, a high-level hacking of the bank account details.

The aim is, therefore, to create a classifier that indicates whether a requested transaction is a fraud or not. We overcome the problem by creating a binary classifier and experimenting with various machine learning techniques to see which fits better.

Problem Statement

The recent lockdown caused by the Covid outbreak, witnessed a sudden increase in online transactions. Due to extensive use of online payments, use of credit cards have increased at a rapid rate.

- This means there is more possibility of fraudulent transactions which eventually leads to heavy financial losses.
- Due to the large amount of data being processed every day, the model build is not fast enough to respond to scam in time.
- Imbalanced Data i.e most of the transactions(99.8%) are not fraudulent which makes it really hard for detecting the fraudulent ones.
- Due to advancement in technology, scammers are also using new methods to fraud people.

Therefore, banks and other financial institutions support the progress of credit card fraud detection applications.

Objectives

The main objective of the project is to detect fraud in credit card transactions. Other objectives includes -

- The model build must be simple and fast enough to detect anomaly and classify it as fraud as soon as possible.
- The model must reduce false positives as much as possible.
- The model should reduce the number of verification measures and process data in real time.
- For protecting the privacy of the user the dimensionality of the data can be reduced.
- A more trustworthy source must be taken which double-checks the data, at least for training the model.
- We can make the model simple and interpretable so that when the scammer adapts to it with just some tweaks we can have a new model up and running to deploy.

Credit Card Fraud Dataset

About the data: The data we are going to use is the Kaggle Credit Card Fraud Detection dataset. This dataset has 492 frauds out of 284,807 total transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. It contains features V1 to V28 which are the principal components obtained by PCA. We are going to neglect the 'Time' feature which is of no use to build the models. The remaining features are the 'Amount' feature that contains the total amount of money being transacted and the 'Class' feature that contains whether the transaction is a fraud case or not. 'Class' feature is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Dataset - https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Methodology & Steps involved -

Importing dependencies -

```
# import the necessary packages

import numpy as np # working with arrays
import pandas as pd # data processing
import matplotlib.pyplot as plt # visualization
import seaborn as sns
import sklearn as skl
from matplotlib import gridspec
```

Importing and displaying the data -

```
data = pd.read_csv("data\creditcard.csv") # loading csv data file
data.drop('Time', axis = 1, inplace = True)

print(data.head()) # displaying data from the csv
```

Output -

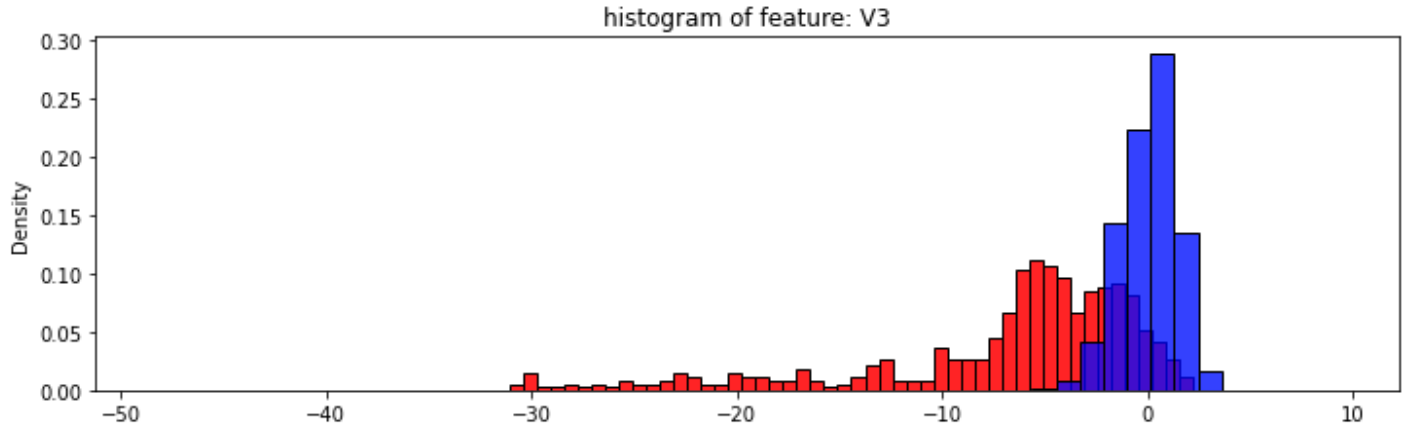
```
         V1        V2        V3        V4        V5        V6        V7  \
0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9       V10  ...       V21       V22       V23       V24  \
0  0.098698  0.363787  0.090794  ... -0.018307  0.277838 -0.110474  0.066928
1  0.085102 -0.255425 -0.166974  ... -0.225775 -0.638672  0.101288 -0.339846
2  0.247676 -1.514654  0.207643  ...  0.247998  0.771679  0.909412 -0.689281
3  0.377436 -1.387024 -0.054952  ... -0.108300  0.005274 -0.190321 -1.175575
4 -0.270533  0.817739  0.753074  ... -0.009431  0.798278 -0.137458  0.141267

        V25       V26       V27       V28  Amount  Class
0  0.128539 -0.189115  0.133558 -0.021053  149.62      0
1  0.167170  0.125895 -0.008983  0.014724    2.69      0
2 -0.327642 -0.139097 -0.055353 -0.059752  378.66      0
3  0.647376 -0.221929  0.062723  0.061458  123.50      0
4 -0.206010  0.502292  0.219422  0.215153   69.99      0

[5 rows x 30 columns]
```

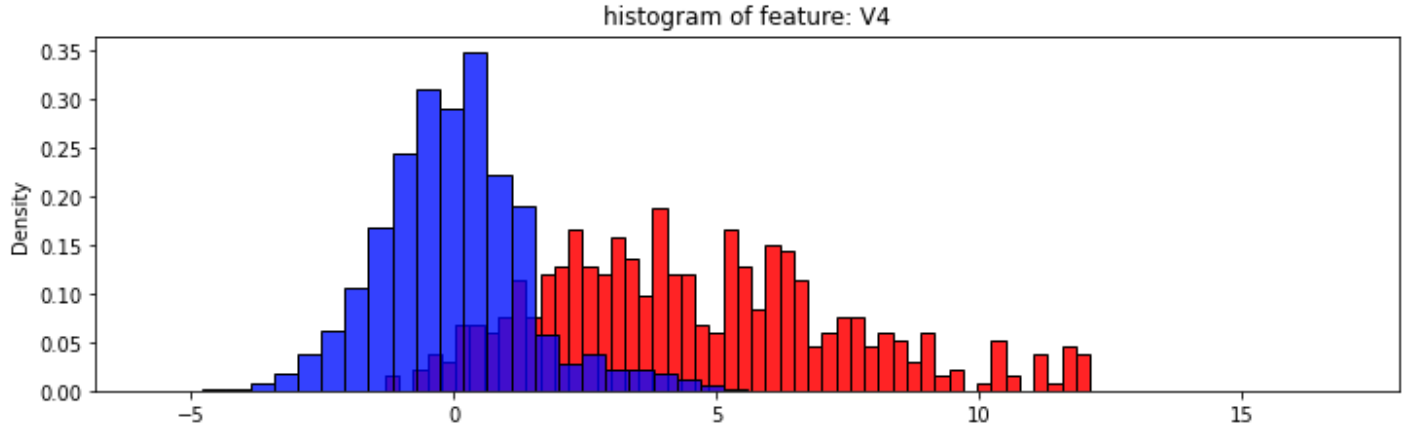
Histogram of the features -

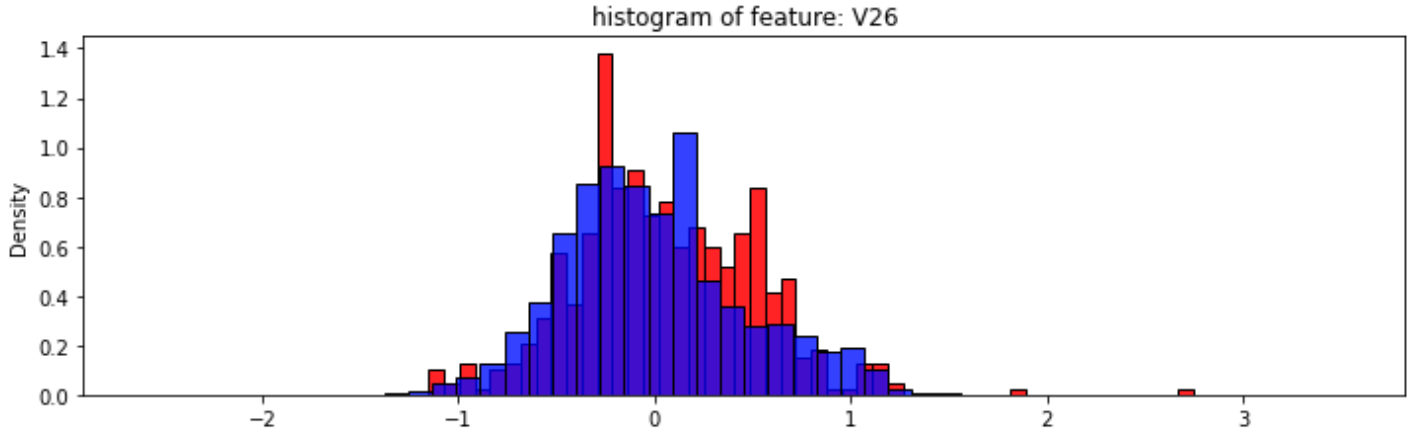
Now we will be visualizing all the features from the dataset on graphs.

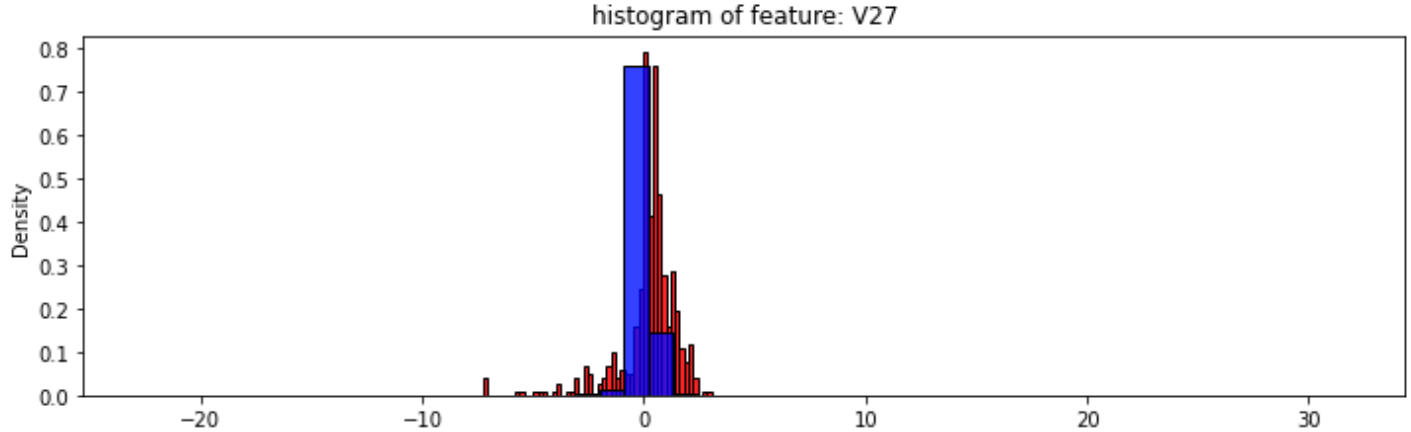
```
# distribution of anomalous features

features = data.iloc[:,0:28].columns
plt.figure(figsize=(12,28*4))
gs = gridspec.GridSpec(28, 1)

for i, c in enumerate(data[features]):
    ax = plt.subplot(gs[i])

    # check documentation of sns.histogram to update the graphs
    sns.histplot(data[c][data.Class == 1], bins=50, color="red", stat="density")
    sns.histplot(data[c][data.Class == 0], bins=50, color="blue", stat="density")
    ax.set_xlabel("")
    ax.set_title("histogram of feature: " + str(c))

plt.show()
```

Output -


histogram of feature: V3
Density
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−50
−40
−30
−20
−10
0
10


histogram of feature: V4
Density
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−5
0
5
10
15


histogram of feature: V26
Density
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−2
−1
0
1
2
3


histogram of feature: V27
Density
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−20
−10
0
10
20
30

Data Processing and EDA -

Let's have a look at how many fraud cases and valid cases are there in our dataset. Along with that, let's also compute the percentage of fraud cases in the overall recorded transactions.

```
fraud_cases = data[data["Class"] == 1]
valid_cases = data[data["Class"] == 0]

total_cases = len(data)
total_fraud_cases = len(fraud_cases)
total_valid_cases = len(valid_cases)
fraud_percentage = total_fraud_cases / float(total_valid_cases)

print("Case Count")
print("----------------------------------------")
print("Total number of cases - {}".format(total_cases))
print("Number of Fraud cases - {}".format(total_fraud_cases))
print("Number of Valid cases - {}".format(total_valid_cases))
print("Percentage of fraud cases - {}".format(fraud_percentage))
print("----------------------------------------")
```

Output -

```
Case Count
----------------------------------------
Total number of cases - 284807
Number of Fraud cases - 492
Number of Valid cases - 284315
Percentage of fraud cases - 0.0017304750013189597
----------------------------------------
```

We can see that out of 284,807 samples, there are only 492 fraud cases which is only 0.17 percent of the total samples. So, we can say that the data we are dealing with is highly imbalanced data and needs to be handled carefully when modeling and evaluating. Next, we are going to get a statistical view of both fraud and valid transaction amount data using the 'describe'.

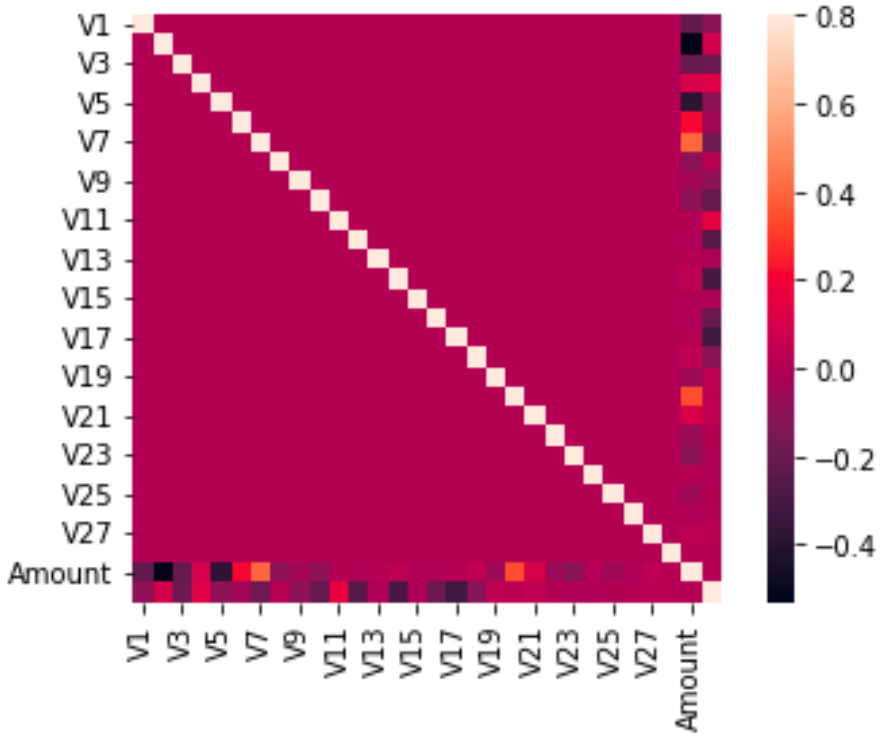
```
print('Case Amount Statistics')
print('-------------------------------------------')
print('Fraud Case Amount Stats')
print(fraud_cases.Amount.describe())
print('-------------------------------------------')
print('Valid Case Amount Stats')
print(valid_cases.Amount.describe())
print('-------------------------------------------')
```

Output -

```
Case Amount Statistics
-----------------------------------------------
Fraud Case Amount Stats
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
-----------------------------------------------
Valid Case Amount Stats
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
-----------------------------------------------
```

As we can clearly notice from this, the average Money transaction for the fraudulent ones are more. This makes this problem crucial to deal with.
Correlation matrix graphically gives us an idea of how features correlate with each other and can help us predict what are the features that are most relevant for the prediction.

```
# Correlation matrix
corrmat = data.corr()
sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
```



In the HeatMap we can clearly see that most of the features do not correlate to other features but there are some features that either have a positive or a negative correlation with each other. For example “V2” and “V5” are highly negatively correlated with the feature called “Amount”. We also see some correlation with “V20” and “Amount”.

Feature Selection & Data Split -

In this process, we are going to define the independent (X) and the dependent variables (Y). Using the defined variables, we will split the data into a training set and testing set which is further used for modeling and evaluating. We can split the data easily using the ‘train_test_split’ algorithm.

```
from sklearn.model_selection import train_test_split # data split

#dividing the X and the Y from the dataset
X=data.drop(["Class"], axis=1)
Y=data["Class"]

#getting just the values for the sake of processing (its a numpy array with no columns)
X_data=X.values
Y_data=Y.values

# Using Skicit-learn to split data into training and testing sets from sklearn.model_selection import train_test_split
# Split the data into training and testing sets

X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data, test_size = 0.2, random_state = 42)
```

Modeling -

Building the Decision Tree & Random Forest Algorithm -

In this step, we will be building two different types of classification models namely **Decision Tree** & **Random Forest**. Even though there are many more models like K-Nearest Neighbors (KNN), Logistic Regression, Support Vector Machine (SVM), and XGBoost, which we can use, but for simplicity we used the above two for solving these classification problems. Both these models can be built feasibly using the algorithms provided by the scikit-learn package.

```
# Importing libraries for Algorithms
from sklearn.tree import DecisionTreeClassifier # Decision tree algorithm
from sklearn.ensemble import RandomForestClassifier # Random forest tree algorithm

# Building the Decision Tree Classifier
decision_tree_model = DecisionTreeClassifier(max_depth = 4, criterion = 'entropy')
decision_tree_model.fit(X_train, Y_train)

# Building the Random Forest Classifier
random_forest_model = RandomForestClassifier(max_depth = 4)
random_forest_model.fit(X_train, Y_train)

# Predictions
decision_tree_predict = decision_tree_model.predict(X_test)
random_forest_predict = random_forest_model.predict(X_test)
```

Decision Tree algorithm is a supervised machine learning algorithm used for classification and regression tasks. Inside the algorithm, we have mentioned the 'max_depth' to be '4' which means we are allowing the tree to split four times and the 'criterion' to be 'entropy' which is most similar to the 'max_depth' but determines when to stop splitting the tree. Finally, we have fitted and stored the predicted values into the 'decision_tree_predict' variable.

Random forest is a supervised machine learning algorithm. It creates a "forest" out of an ensemble of "decision trees", which are normally trained using the "bagging" technique. The bagging method's basic principle is that combining different learning models improves the outcome.
We built the classifier using the 'RandomForestClassifier' algorithm and we mentioned the 'max_depth' to be 4 just like how we did to build the decision tree model. Finally, fitting and storing the values into the 'random_forest_predict'.

Remember that the main difference between the decision tree and the random forest is that, the decision tree uses the entire dataset to construct a single model whereas, the random forest uses randomly selected features to construct multiple models. That's the reason why the random forest model is used versus a decision tree.

Evaluation -

In this process we are going to evaluate our built models using the evaluation metrics provided by the scikit-learn package. Our main objective in this process is to find the best model for our given case. The evaluation metrics we are going to use are the accuracy score metric, precision score metric, recall score matric, f1 score metric, and matthews correlation coefficient metric.

```
#Evaluating the classifier
#printing every score of the classifier
#scoring in any thing

from sklearn.metrics import classification_report, accuracy_score,precision_score,recall_score,f1_score,matthews_corrcoef
from sklearn.metrics import confusion_matrix

n_outliers = len(fraud_cases)
n_errors = (decision_tree_predict != Y_test).sum()

decision_tree_accuracy = accuracy_score(Y_test,decision_tree_predict)
random_forest_accuracy = accuracy_score(Y_test,random_forest_predict)

decision_tree_precision = precision_score(Y_test,decision_tree_predict)
random_forest_precision = precision_score(Y_test,random_forest_predict)

decision_tree_recall = recall_score(Y_test,decision_tree_predict)
random_forest_recall = recall_score(Y_test,random_forest_predict)

decision_tree_f1 = f1_score(Y_test,decision_tree_predict)
random_forest_f1 = f1_score(Y_test,random_forest_predict)

decision_tree_mcc=matthews_corrcoef(Y_test,decision_tree_predict)
random_forest_mcc=matthews_corrcoef(Y_test,random_forest_predict)
```

	Parameters / Model	Decision Tree	Random Forest
0	Accuracy	0.999438	0.999298
1	Precision	0.851064	0.914286
2	Recall Score	0.816327	0.653061
3	F1-Score	0.833333	0.761905
4	Matthews Correlation Coefficient	0.833234	0.772397

Confusion Matrix -

Typically, a confusion matrix is a visualization of a classification model that shows how well the model has predicted the outcomes when compared to the original ones. Usually, the predicted outcomes are stored in a variable that is then converted into a correlation table. Using the correlation table, the confusion matrix is plotted in the form of a heatmap. Even though there are several built-in methods to visualize a confusion matrix, we are going to define and visualize it from scratch for better understanding.

```
#printing the confusion matrix
LABELS = ["Valid Cases", "Fraud Cases"]

decision_tree_conf_matrix = confusion_matrix(Y_test, decision_tree_predict)
random_forest_conf_matrix = confusion_matrix(Y_test, random_forest_predict)

# plt.figure(figsize=(12, 12))
```

✓ 0.1s

```
sns.heatmap(decision_tree_conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d")

plt.title("Confusion matrix")
plt.ylabel("True class")
plt.xlabel("Predicted class")
plt.show()
```

✓ 0.4s

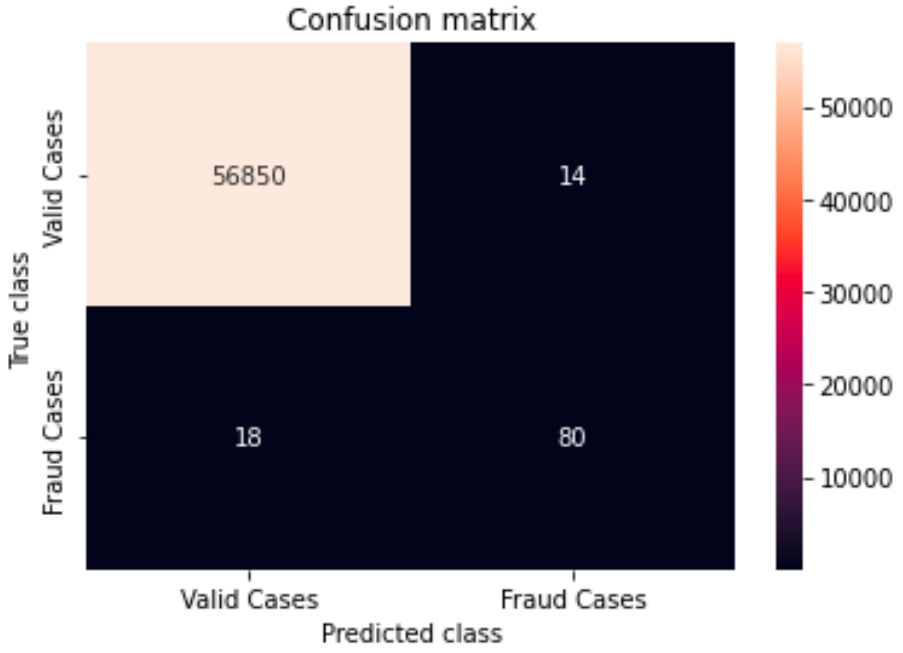
Outputs are collapsed ···

+ Code + Markdown

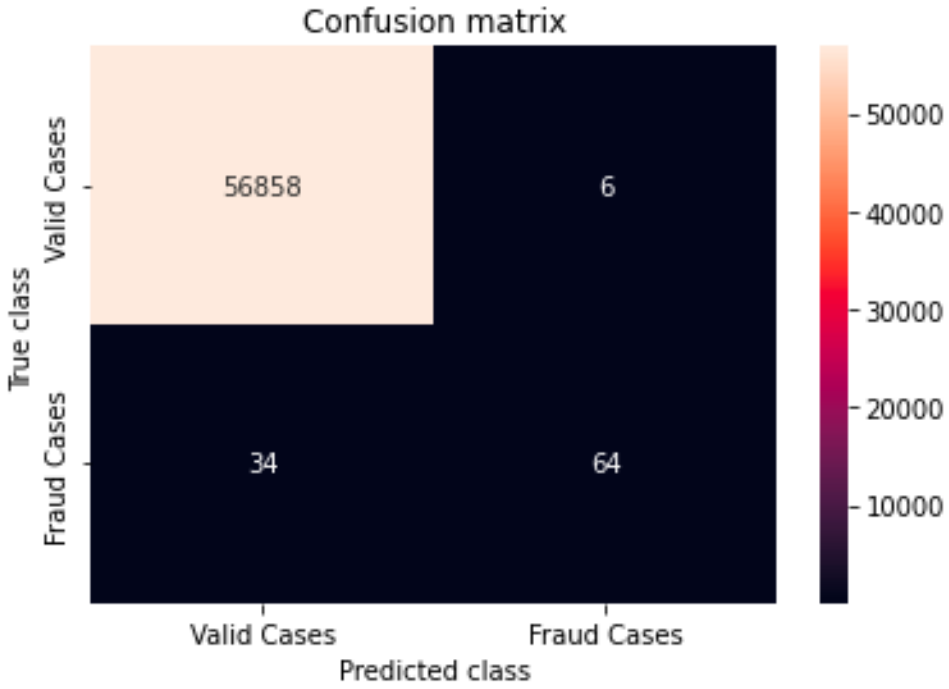
```
sns.heatmap(random_forest_conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d")

plt.title("Confusion matrix")
plt.ylabel("True class")
plt.xlabel("Predicted class")
plt.show()
```

Output -

Decision Tree Confusion Matrix



Random Forest Confusion Matrix



Conclusion

We have successfully built two different types of classification models: decision tree and random forest. After that, we evaluated both of the models using the evaluation metrics and chose which model is most suitable for the given case which is the decision tree. In this project, we have limited our model count to two but there are many more models to explore. Also, we have built the models feasibly in python but, there is more and more math and statistics behind each of the models. The model is fast, it is definitely simple and most importantly easily interpretable. The privacy of the user is still intact, as the data used had its dimensionality reduced in the beginning. Through this project, we understood and learned machine learning techniques and achieved an accuracy of more than 99%.

References

1. Detecting Credit Card Fraud with Machine Learning, Aaron Rosenbaum, Stanford University, Stanford, CA, 94305, USA
2. https://www.kaggle.com/mlg-ulb/creditcardfraud
3. Credit Card Fraud Detection using Machine Learning Algorithms, Vaishnavi Nath Dornadulaa , Geetha Sa, VIT, Chennai, India

Draft verified by

Project Guide
Dr. Satendra Kumar

Head Of Department
(Dept. Of Cybernetics)
Prof. (Dr.) Monit Kapoor