CREDT CARD FRAUD DETECTION USING MACHINE LEARNNG

By:

Surya Devathi

(N659A624)

Submitted to:

Dr. Dukka KC

CS – 697AK: Introduction to Data Science

Contents

- Abstract
- Problem Statement and Motivation
- Background and Literature Review
- Methodology
    - Description of Method
    - Description of Dataset
    - Description of your code
- Results
- Conclusion and Discussion
- References

Abstract

In everyday life, Credit cards have been a necessity for regular purchasing activities. However, the number of fraudulent transactions has also increased enormously with proportional usage of credit cards. In this, digital world with the improvements in technology the scope for theft and fraudulent transactions have also became common involving a payment card, like a debit or credit card. Credit card companies need to play a prominent in detecting the fraudulent transaction on credit card transactions, to make sure customers didn't pay for the items which they haven't bought. Here comes machine learning algorithms and data science into the picture which helps to find a solution to the problem. With the help of historical data of credit card fraudulent transactions, we develop a model such that it identifies whether transaction is valid or fraud. We can use this model to identify a new transaction is fraudulent or not. Our objective right here is to detect 100% fraudulent transactions at the same time as minimizing the wrong fraud classifications. Here we use some machine learning algorithms like Local Outlier Factor and Isolation Forest Algorithm on the transformed data to make prediction.

Keywords— Credit Card Fraud, application of machine learning, data science, isolation forest algorithm, local outlier factor, automated fraud detection.

Problem Statement and Motivation

'Fraud' in credit card transactions is unauthorized and undesirable utilization of an account through a person other than the owner of that account. Vital prevention measures can be taken to stop this abuse and the behavior of such fraudulent practices are studied to limit it and guard in opposition to comparable occurrences within the future. In different words, credit Card Fraud may be described as a case where someone uses a person else's credit card for private motives whilst the proprietor and the card issuing government are blind to the reality that the card is getting used.

Fraud detection entails tracking the usage of populations of customers in order to estimate, or keep away from objectionable behavior, which consist of fraud, intrusion, and defaulting.

There is totally applicable trouble that demands the eye of groups which include machine learning and data science where the solution to this hassle can be automated.

This problem is specially hard from the angle of gaining knowledge of, as its miles characterized via various factors such as class imbalance. The variety of legitimate transactions a way outnumbers fraudulent ones. Additionally, the transaction styles frequently trade their statistical attributes over the path of time.

However, the implementation of a real-world fraud detection system is the big challenge. But, in real world examples, the large stream of payment requests is quickly scanned by automatic tools that determine which transactions to authorize.

Machine learning algorithms are used to analyze all the authorized transactions and report the suspicious ones. To confirm if the transaction was genuine or fraudulent, these reports are investigated by professionals who contact the cardholders.

The feedback provided by the investigators to automated system is used to train and update the algorithm. So, that it may increase the performance of fraud-detection time to time.

Fraud detection techniques are constantly advanced to shield criminals in adapting to their fraudulent techniques. These frauds are classified as:

- Credit Card Frauds: Online and Offline
- Card Theft
- Account Bankruptcy
- Device Intrusion
- Application Fraud
- Counterfeit Card
- Telecommunication Fraud

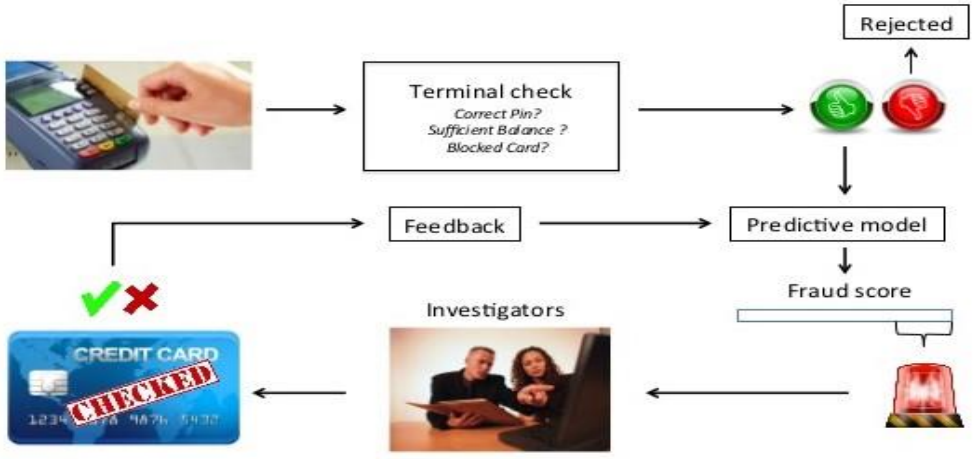


Fig: Credit Card Transaction Process

At present we have different approaches used for fraud detection. Some of them are:

- Artificial Neural Network
- Fuzzy Logic
- Genetic Algorithm
- Logistic Regression
- Decision tree
- Support Vector Machines
- Bayesian Networks
- Hidden Markov Model
- K-Nearest Neighbor

Background and Literature Review

Fraud is the unlawful or criminal deception intended to result in financial or personal benefit. It is a deliberate act that is against the law, rule or policy with an aim to gain unauthorized financial benefit.

Huge number of literatures pertaining to anomaly or fraud detection in this domain have been published already and are available for public usage. A survey conducted by Clifton Phua and his associates have revealed that techniques used in this domain include data mining applications, automated fraud detection, adversarial detection.

In another paper, Suman, Research Scholar, GJUS&T at Hisar HCE presented techniques like Supervised and Unsupervised Learning for credit card fraud detection. These methods and algorithms fetched an unexpected success in some areas, but they failed to provide a permanent and consistent solution in identifying fraud detection.

A similar research was presented by Wen-Fang YU and Na Wang in the same domain where they used Outlier mining, Outlier detection mining and Distance sum algorithms to accurately so that it's easy predict fraudulent transaction in an emulation experiment of credit card transaction data set of one certain commercial bank.

Outlier mining is a field of data mining which is used in monetary and internet fields. It deals with detecting objects that are detached from the main system i.e. the transactions that aren't valid. They studied the customer's behavior and identified the attributes based on the trends using those attributes they've calculated that distance between the observed value of that attribute and its predetermined value.

Unconventional strategies such as hybrid data mining/complicated community category set of rules is capable to understand unlawful instances in an actual card transaction records set, based on network reconstruction set of rules that allows growing representations of the deviation of one example from a reference organization have proved efficient usually on medium sized on-line transaction.

Artificial Genetic Algorithm is one of the approaches that shed new light in this domain, countered fraud from a different direction. It proved accurate in finding out the fraudulent transactions and reducing the number of false alerts. Even though, it was accompanied by classification problem with variable misclassification costs.

Methodology

A. Description of Dataset

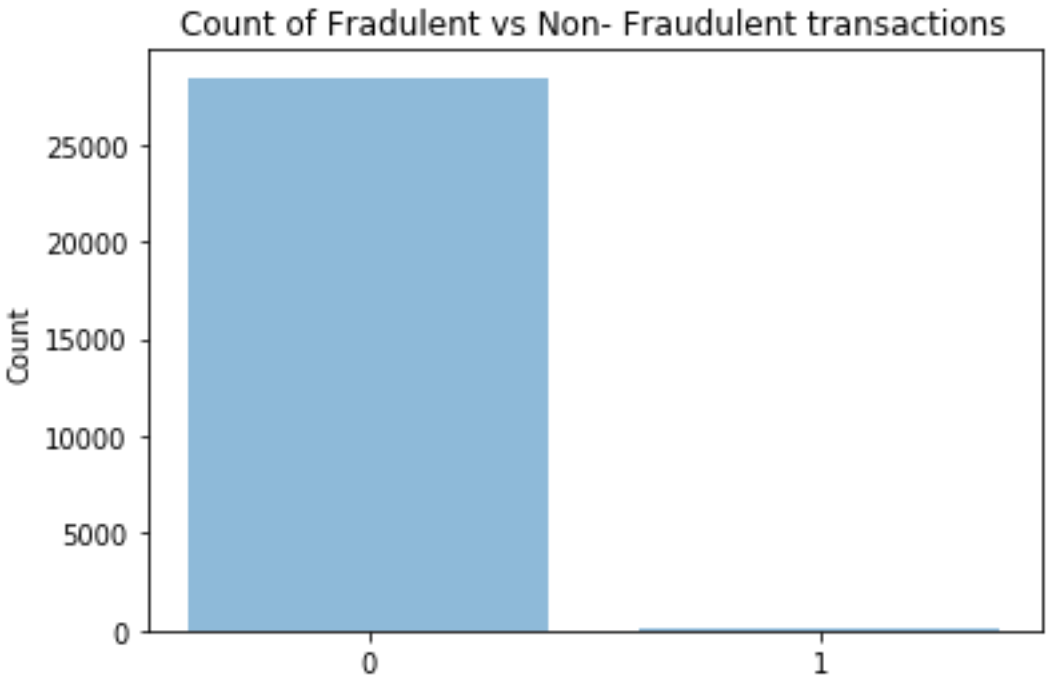
First of all, the dataset is obtained from Kaggle, a data analysis website which provides datasets.

The dataset consists of 31 columns out of which 28 are named as v1-v28 to protect sensitive data banks hide the name of attributes. Remaining three columns represent Time, Amount and

Class. Time shows the time gap between the first transaction and the following one. Amount is the amount of money transacted.

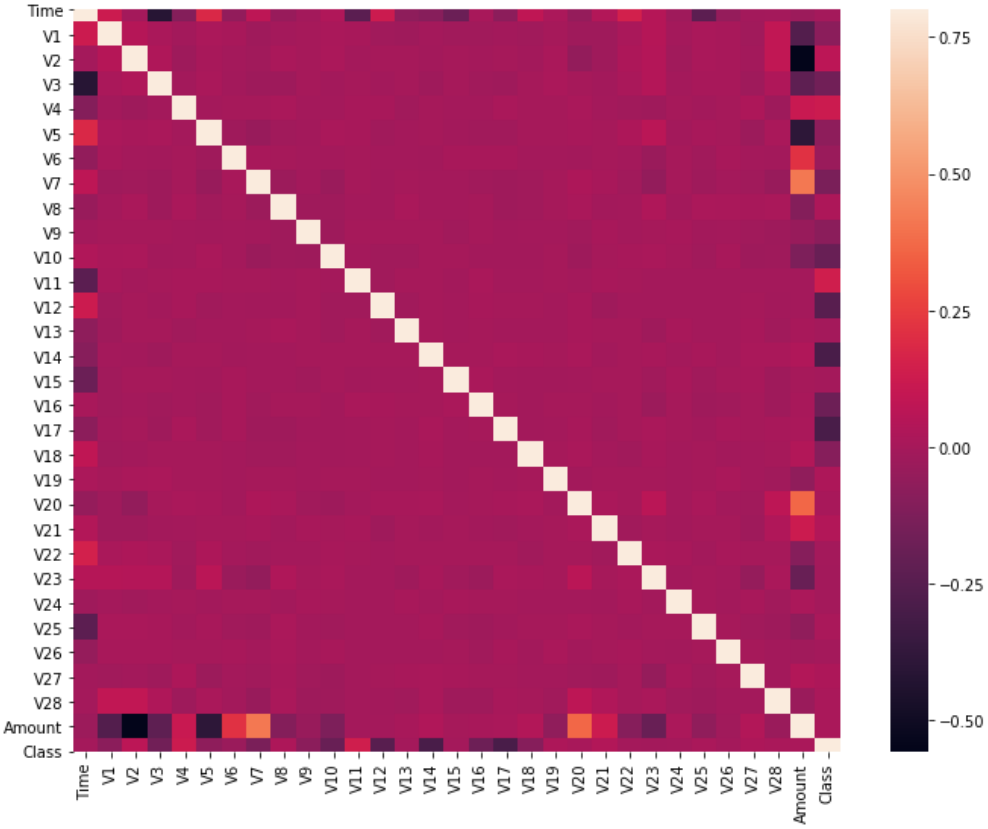
Class 0 represents a valid transaction and 1 represents a fraudulent transaction.

We plot different graphs to check for inconsistencies in the dataset and to visually comprehend it.



After checking this dataset, we plot a histogram for every column. This is done to get a graphical representation of the dataset which can be used to verify that there are no missing any values in the dataset. This is done to ensure that we don't require any missing value imputation and the machine learning algorithms can process the dataset smoothly.

After this analysis, we plot a heatmap to get a colored representation of the data and to study the correlation between out predicting variables and the class variable. This heatmap is shown below:



B. Description Method

The approach we propose, we need to identify the anomalies for which we use some machine learning algorithms to detect such anomalous activities, called outliers.

The dataset is now formatted and processed. We standardized the time and amount column then Class column is removed to ensure fairness of evaluation. The data is processed by a set of algorithms from sklearn modules. The following module diagram explains how these algorithms work together: This data is fit into a model and the following outlier detection modules are applied on it:

- Local Outlier Factor
- Isolation Forest Algorithm

These algorithms are a part of sklearn. The ensemble module in the sklearn package includes ensemble-based methods and functions for the classification, regression and outlier detection.

This free and open-source Python library is constructed using NumPy, SciPy and matplotlib modules which provides a number of easy and green equipment which may be used for records analysis and system studying. It features diverse category, clustering and regression algorithms and is designed to interoperate with the numerical and scientific libraries.

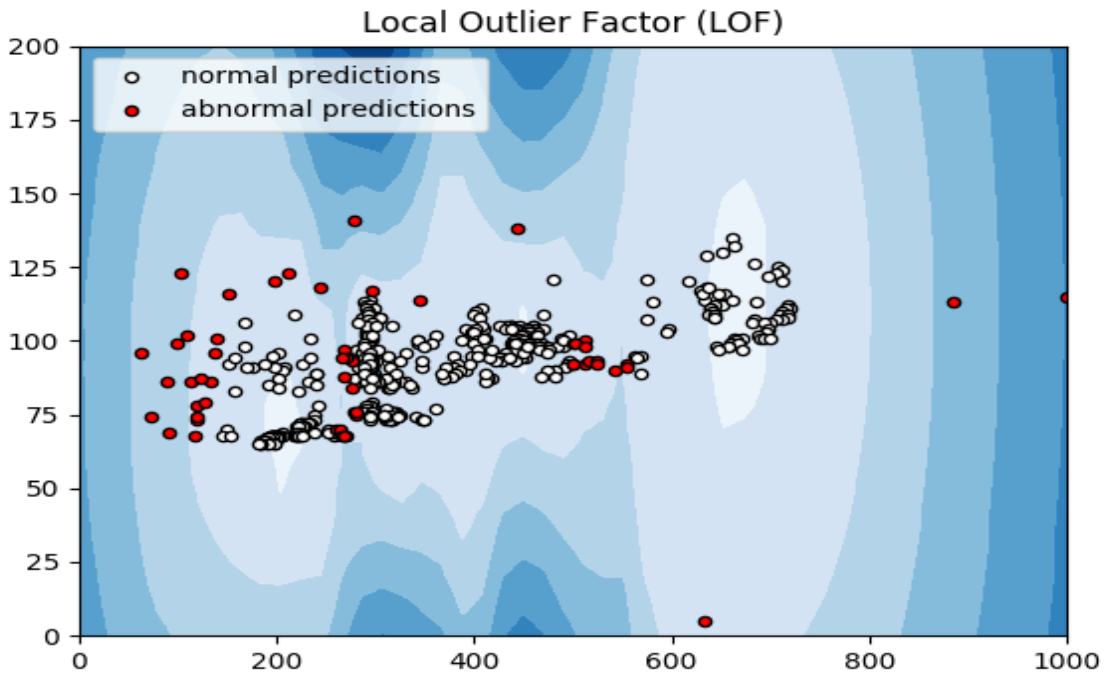
I have used Jupyter Notebook platform to make a program in Python to demonstrate the approach that we suggest. This program can also be executed on using various tools such as the cloud using Google Collab platform which supports all python notebook files.

Local Outlier Factor

It is an Unsupervised Outlier Detection algorithm. 'Local Outlier Factor' refers to the anomaly score of each sample. It measures the local deviation of the sample data with respect to its neighbors.

More precisely, locality is given by k-nearest neighbours, whose distance is used to estimate the local data.

On plotting the results of Local Outlier Factor algorithm, we get the following figure:



By comparing the local values of a sample to that of its neighbours, one can identify samples that are substantially lower than their neighbours. These values are quite amanous and they are considered as outliers.

As the dataset is very large, we used only a fraction of it in out tests to reduce processing times.

The final result with the complete dataset processed is also determined and is given in the results section of this paper.

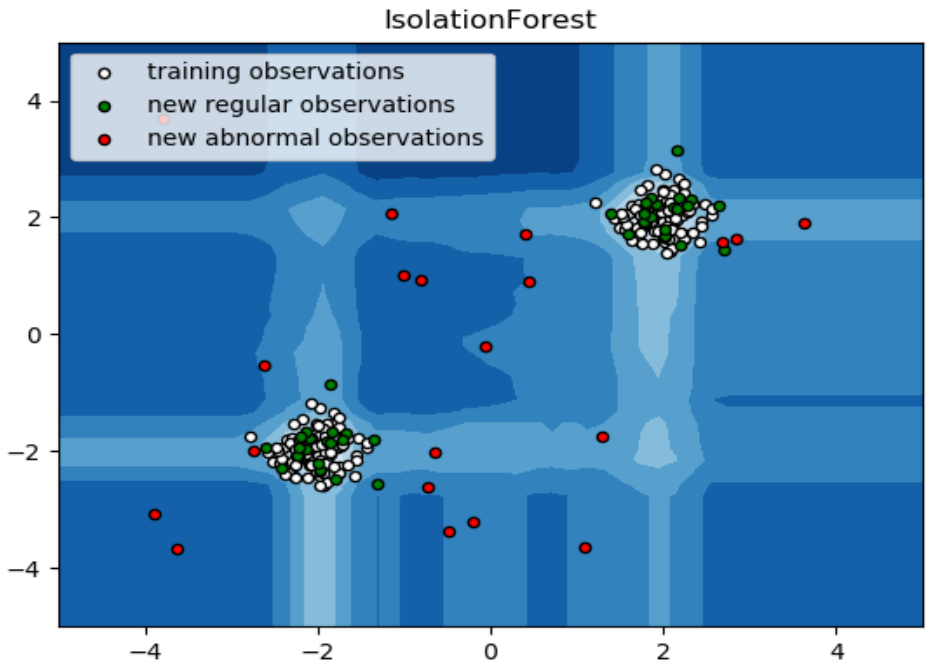
Isolation Forest Algorithm

The Isolation Forest isolates observations by arbitrarily selecting a feature and then randomly selecting a split value between the maximum and minimum values of the designated feature.

Recursive partitioning can be represented by a tree, the number of splits required to isolate a sample is equivalent to the path length root node to terminating node.

The average of this path length gives a measure of normality and the decision function which we use.

On plotting the results of Isolation Forest algorithm, we get the following figure:



Partitioning them randomly produces shorter paths for anomalies. When a forest of random trees mutually produces shorter path lengths for specific samples, they are extremely likely to be anomalies.

Once the anomalies are detected, the system can be used to report them to the concerned authorities. For testing purposes, we are comparing the outputs of these algorithms to determine their accuracy and precision.

C. Description of Code

1. Initially we need to import all the libraries required

```
# import the necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import gridspec
```

2. Load the data from csv file using pandas

```
# copy the path for the csv file
path ="credit_card.csv"
data = pd.read_csv(path)
```

3. Understand the Data

```
# Grab a peek at the data
data.head()
```

4. Describing the dataset

```
# Print the shape of the data
# data = data.sample(frac = 0.1, random_state = 48)
print(data.shape)
print(data.describe())
```

5. Identify the imbalance present in the dataset

```
# Determine number of fraud cases in dataset
fraud = data[data['Class'] == 1]
valid = data[data['Class'] == 0]
outlierFraction = len(fraud)/float(len(valid))
print(outlierFraction)
print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))
print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))
```

We have only 0.17% of all the transactions are fraudulent transaction which clearly shows that data is imbalanced. So, let's move on and perform model on the imbalanced dataset. If the accuracy is not appropriate, we can find another way to balance the dataset. So, we will start without balancing and perform balancing if only its accuracy is not upto the mark.

6. Code to get the details of fraudulent transactions

```
print("Amount details of the fraudulent transaction")
fraud.Amount.describe()
```

7. Code to get the details for valid transactions

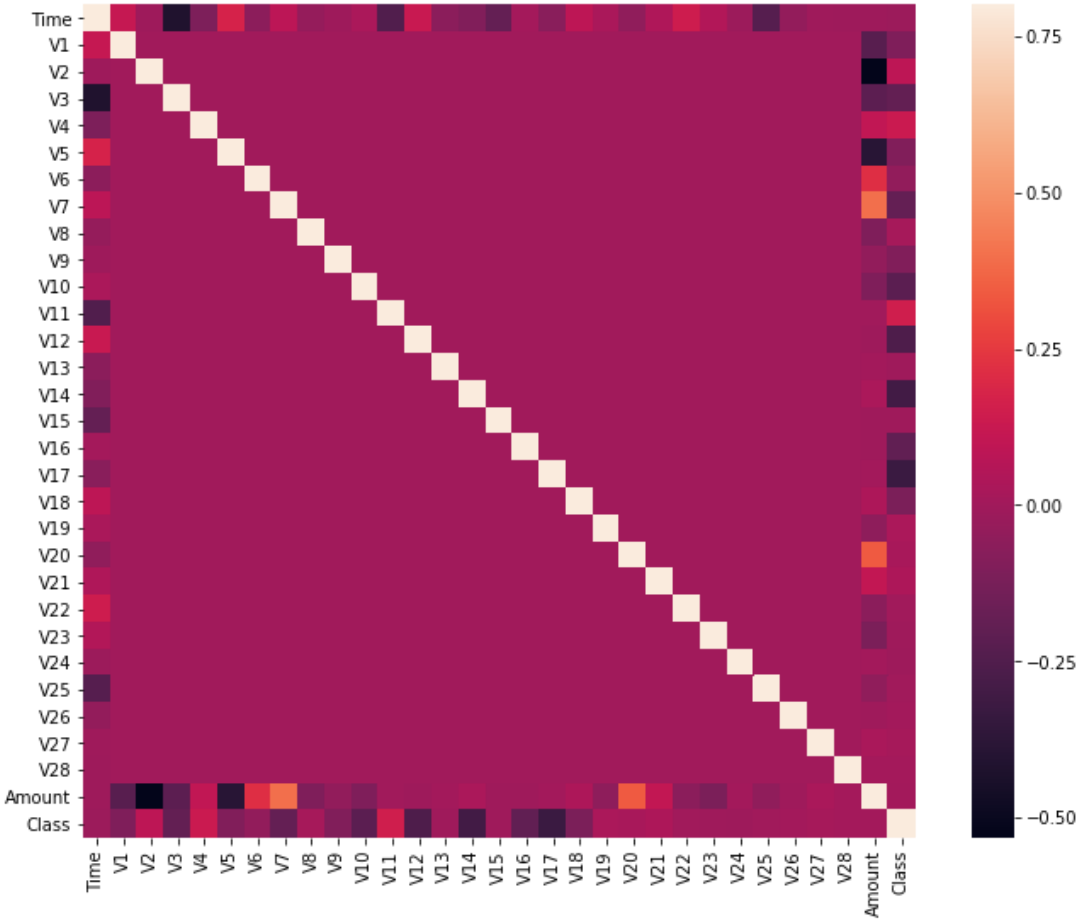
```
print("details of valid transaction")
valid.Amount.describe()
```

From above two steps we observe that average fraudulent transaction amount is more than the valid transaction amount. We can estimate how important to deal with such cases.

8. Plotting the Correlation matrix

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()
fig = plt.figure(figsize = (12, 9))
sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
```



From this matrix map we can identify which attributes plays a role in identifying the fraudulent transaction.

9. We split the data into inputs and outputs using X and Y values.

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

```
Y = data["Class"]
print(X.shape)
print(Y.shape)
# getting just the values for the sake of processing
# (it's a numpy array with no columns)
xData = X.values
yData = Y.values
```

10. Divide the training and the testing data

    Here we spilt the data into two parts. One part is used for testing the model and other is used for training the model.

```
# 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
xTrain, xTest, yTrain, yTest = train_test_split( xData, yData, test_size = 0.2, random_state = 42)
```

11. Constructing the Random Forest Algorithm

```
# Building the Random Forest Classifier (RANDOM FOREST)
from sklearn.ensemble import RandomForestClassifier
# random forest model creation
rfc = RandomForestClassifier()
rfc.fit(xTrain, yTrain)
# predictions
yPred = rfc.predict(xTest)
```

12. Finally, construct all types of evaluating parameters.

```
# Evaluating the classifier
# printing every score of the classifier
# scoring in anything
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix

n_outliers = len(fraud)
n_errors = (yPred != yTest).sum()
print("The model used is Random Forest classifier")

acc = accuracy_score(yTest, yPred)
print("The accuracy is {}".format(acc))

prec = precision_score(yTest, yPred)
print("The precision is {}".format(prec))
```

```
rec = recall_score(yTest, yPred)
print("The recall is {}".format(rec))

f1 = f1_score(yTest, yPred)
print("The F1-Score is {}".format(f1))

MCC = matthews_corrcoef(yTest, yPred)
print("The Matthews correlation coefficient is{}".format(MCC))
```

13. Draw the confusion matrix

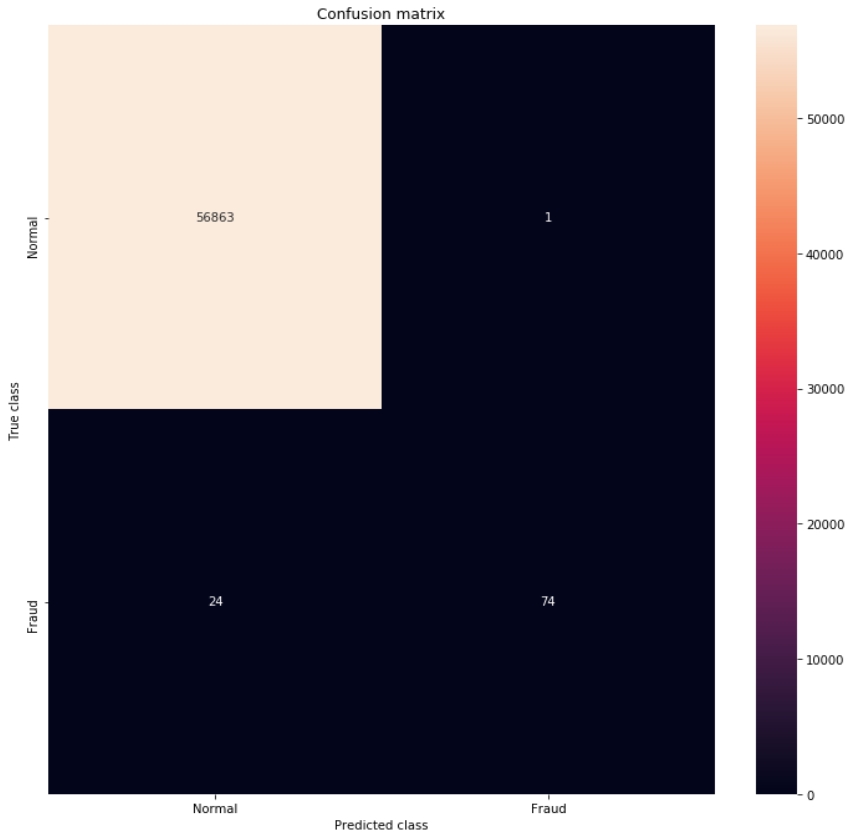
```
# Evaluating the classifier
# printing every score of the classifier
# scoring in anything
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix

n_outliers = len(fraud)
n_errors = (yPred != yTest).sum()
print("The model used is Random Forest classifier")

acc = accuracy_score(yTest, yPred)
print("The accuracy is {}".format(acc))

prec = precision_score(yTest, yPred)
print("The precision is {}".format(prec))

rec = recall_score(yTest, yPred)
print("The recall is {}".format(rec))
f1 = f1_score(yTest, yPred)
print("The F1-Score is {}".format(f1))
MCC = matthews_corrcoef(yTest, yPred)
print("The Matthews correlation coefficient is{}".format(MCC))
```



Results

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. We used 10% of entire dataset for making test faster.

```
Isolation Forest: 71
0.99750711000316
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28432
           1       0.28      0.29      0.28        49

    accuracy                           1.00     28481
   macro avg       0.64      0.64      0.64     28481
weighted avg       1.00      1.00      1.00     28481

Local Outlier Factor: 97
0.9965942207085425
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28432
           1       0.02      0.02      0.02        49

    accuracy                           1.00     28481
   macro avg       0.51      0.51      0.51     28481
weighted avg       1.00      1.00      1.00     28481
```

Those outcomes alongside with the class record for each set of rules is given in the output as follows, wherein class 0 approach the transaction changed into decided to be valid and 1 way it became decided as a fraud transaction. This result matched in opposition to the class values to test for false positives.

Conclusion

Credit card fraud is an act of dishonesty without a doubt. I have indexed out the most common techniques of fraud alongside with their detection methods and reviewed recent findings in this discipline. It also explained in element, how system gaining knowledge of can be applied to get higher consequences in fraud detection along side the set of rules, pseudocode, explanation its implementation and experimentation outcomes.

At the same time as the set of rules does attain over 99.6% accuracy, its precision stays only at 28% while a tenth of the statistics set is taken into consideration. However, whilst the whole dataset is fed into the algorithm, the precision rises to 33%. This excessive percentage of accuracy is to be predicted due to the huge imbalance between the range of legitimate and variety of authentic transactions.

Future Enhancements

While we couldn’t attain out goal of a 100% accuracy in fraud detection, we did stop up creating a system that can, with sufficient time and records, get very close to that purpose. As with this type of task, there's a few room for development right here.

The very nature of this project allows for multiple algorithms to be incorporated collectively as modules and their results can be combined to boom the accuracy of the final result.

This version can in addition be advanced with the addition of extra algorithms into it. But, the output of these algorithms wishes to be in the identical format as the others. as soon as that condition is satisfied, the modules are easy to add as executed within the code. This gives a excellent diploma of modularity and versatility to the project.

More room for improvement can be discovered inside the dataset. As proven before, the precision of the algorithms will increase while the scale of dataset is elevated. Consequently, greater facts will truely make the model extra correct in detecting frauds and decrease the quantity of false positives. But, this calls for official guide from the banks themselves.

References

- "Credit Card Fraud Detection Based on Transaction Behaviour -by IEEE Region 10 Conference, Malaysia.
- "Credit Card Fraud Detection: A Realistic Modeling and a Novel Neural Network and Learning systems Vol 29.
- David J.Wetson,David J.Hand,M Adams,Whitrow and Piotr Jusczak Issue 2008.
- Dataset from Kaggle : https://www.kaggle.com/mlg-ulb/creditcardfraud
- https://pythonspot.com/matplotlib-bar-chart/
- https://scikitlearn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html