# Detection of anomalies in credit card transactions.

Gazali Agboola
Applied Science and Technology

goagboola@aggies.ncat.edu

***Abstract -- In the financial sector, financial fraud is a rising problem with serious effects. The identification of credit card fraud in online transactions was greatly helped by big data analysis. Due to two main factors—first, the profiles of legitimate and fraudulent behavior change often, and second, credit card fraud data sets are extremely skewed—detection of credit card fraud, a data mining challenge, becomes difficult. The dataset sampling strategy, variable choice, and detection methods employed all have significant impacts on the effectiveness of fraud detection in credit card transactions. The performance of isolation forest (iForest), local outliers' factor(LOF), and Logistic regression (LR) on highly skewed credit card fraud data are examined in this study. The model was constructed using real data from European cardholders, and under-sampling methods were also applied. The three models were implemented in Python, and the effectiveness of the methods is assessed based on accuracy, recall, precision, and the Kappa score coefficient. The outcomes indicate that Isolation Forest has the best accuracy at 99.8% in detecting outliers.***

***Keywords—Credit card fraud, Isolated Forest, Machine learning, Outliers, Anomalies.***

## I. INTRODUCTION

Credit card frauds are very easy to carry out by perpetrators. Without much stress, a significant amount can be withdrawn within a short time from the owner's account without the owner's consent.

Credit card has become the most general mode of payment for online and regular purchases. The credit fraud rate tends to accelerate, and the number of people affected by credit card fraud almost doubled between 2019 and 2020, and in 2021, nearly 1.7 million people were impacted by credit card fraud and identity theft worldwide [1]. In 2020, $28.58 billion was lost to credit card fraud, and there are predictions that fraud will increase in 2022, so acquirers are assessing their fraud detection strategies [2]. Typically, fraud costs organizations 5% of their revenues every year, with a median loss of $125,000, according to the Association of Certified Fraud Examiners (ACFE). Because human detection is time-consuming and unreliable, financial institutions can no longer rely on it, and the introduction of big data has made these manual procedures even less successful. One of the most effective ways to identify credit card fraud is by using computational intelligence (CI)-based tools that can spot irregularities in credit card transactions. [2].

In credit card fraud, criminals make purchases or obtain cash advances using the credit card account of another person. It can occur in one of several ways, such as using another person's accounts, stealing a physical credit card or another person's account numbers and PINs, or opening new credit card accounts in another person's name. Credit card methods include the following:

1. Card theft by criminals: Taking a card from a restaurant table, bar, or wallet (or just stealing the entire wallet or purse) is a classic way to steal a credit card. When your card goes missing, or you learn that you should have received a card that never arrived, you should notify the card issuer immediately.
2. Account takeover by criminals: Criminals contact your card issuer and use your personal information to change your PIN, password, mailing address, and so on so that they control your account, and you cannot access it.
3. Cloned cards by criminals: Skimmers installed on gas pumps and retail sales terminals can take your card number when you swipe the card, then duplicate it for illicit use.
4. Card-not-present theft: In this case, thieves use a credit card account without physically possessing the card. It requires only that the thief knows the cardholder's name, account number, and the card's security code to steal from the original card owner.

The purpose of anomaly detection algorithms is to detect or identify anomalous patterns that differ from expected trends or behaviors, called outliers. The detection of events or items associated with suspicious occurrences could be very problematic to the victims [3]. Detecting anomalies can be extremely useful in detecting credit card fraud because fraudulent transactions are rare compared to authentic transactions.

Machine learning and deep learning models can recognize unusual credit card transactions and fraud. First, raw data is collected and sorted, which then is used to train a model that predicts fraud. To detect credit card fraud, machine learning offers the following solutions:

- Classifying whether credit card transactions are valid or fraudulent using techniques like logistic regression, random forests, support vector machines (SVMs), deep neural networks along with autoencoders, long short-term memory (LSTM) networks, and convolutional neural networks (CNNs)
- Using credit card profiling to identify cardholders and fraudsters using credit cards
- The use of outlier detection methods to identify transactions that are significantly different from regular credit card transactions. – anomalies in credit card transactions. Isolation forest(iForest), and Local Outliers Factor(LOF) can be used here.

## II. CHALLENGES

Building a fraud detection system is not as straightforward as it looks. The practitioner needs to ensure the right learning strategy is chosen, that is, supervised learning or unsupervised learning, and which algorithms to use (e.g., Logistic regression, decision trees, isolated forest, etc.) and which features to use, and most importantly, dealing with the class imbalance problem (fraudulent cases

are extremely minimal when compared with the legitimate cases) [4], less than 0.5% of transactions are fraudulent. The dataset to be used for this project is seriously imbalanced as shown in figure 1.

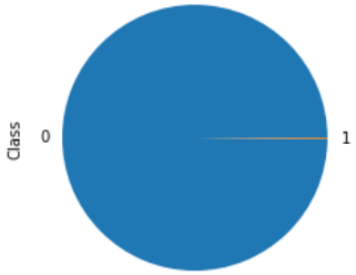



**Figure 1**

Additionally, fraudulent transactions keep evolving as existing fraud detection systems flag old methods, and fraudsters learn new ways to mimic the spending behavior of legitimate cardholders [2]. In this way, fraudulent and normal behavior profiles are continually changing. Fraudsters always try to make every fraudulent transaction looks genuine, and this makes fraud detection a challenging task to carry out.

Another challenge facing the building of credit card detection systems is that financial institutes rarely disclose customer data to the public due to confidentiality issues, and real financial datasets are very difficult to get. This is one of the biggest challenges in fraud detection research works.

## III. Literature Review

Paper [2] proposes anomaly detection methodology by comparing the performance of naïve Bayes, k-nearest neighbor, and logistic regression on 284,807 credit card transactions by European cardholders. Their comparative analysis shows that the k-nearest neighbor performance of 97.92% is better than naïve Bayes and logistic regression techniques.

Paper [5] uses the clustering method to divide the transactions of the European cardholders into different clusters (or groups) based on their transaction amount, that is, high, medium, and low using range partitioning. They use the Sliding-Window method, by aggregating the transactions into respective groups to extract some features from the window to determine the cardholder's behavioral patterns. These features include maximum and minimum transaction amounts, the average amount in the window, and the time between transactions.

Another paper [3] presents a technique that exhibits the ability to distinguish anomalies and mere inliers by creating several decision trees for every data point and the paper suggests that the Random Forest methodology provides the most reliable effects, followed by Logistic Regression and SVM after obtaining the AUC of 98.72% for Random Forest methodology.

## IV. Methodology

Detecting fraud in credit card transactions is a complex and wide-ranging area. Many techniques have been proposed over the years, mostly from the anomaly detection branch of data science. However, depending on the data available, most of these techniques can be classified into two categories:

Case 1: We have enough fraud examples in the dataset to support this hypothesis. In this case, a machine learning model can be trained, or probabilities calculated for the two classes (legitimate transactions and fraudulent transactions), and a model can be applied to new transactions to determine their legitimacy. Here we can use all supervised machine learning algorithms e.g., logistic regression, random forest, Naïve Bayes et. c.

Case 2: There are no (or very few) fraud examples in the dataset. In this case, we need to be more creative since we have no or very few examples of fraudulent transactions. We can treat a few frauds as outliers and apply outlier detection algorithms-isolation forest or downsample the dataset by picking equal samples for the fraudulent transaction and legitimate transaction.

The major aspect of this project is to develop a best-suited algorithm to find the outliers or frauds in the case of credit cards. We will implement an isolated forest

### *A. Data*

The dataset we use is obtained from Kaggle. The dataset we use contains transactions made using credit cards in September 2013 by European cardholders [7]. This dataset contains transactions that occurred in two days, and there are 492 frauds out of 284,807 transactions as shown in the data shape in figure 2. The dataset is unbalanced, there is only 0.172% (frauds) class as 1 and the rest are zeroes (legitimate class). Figure 3 shows that there are no missing data in the dataset as all the attributes from V1 to V28 show the equal length of rows as 284,807. Figure 5 shows that there is no correlation between the Class and Amount, this implies we cannot detect anomalies based on transaction amount but the spending habit of the customer. The pattern of the spending of a cardholder needs to be studied before we can conclude whether a transaction is legit or a fraud.

```
In [9]: # To import the necessary packages for data cleaning and visualization
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
        from pandas import read_csv
        import missingno as msno

In [10]: #Load the sataset
         filename = 'creditcard.csv'
         data = read_csv(filename)

In [11]: data.shape

Out[11]: (284807, 31)
```

**Figure 2**

```
In [13]: msno.bar(data)
Out[13]: <AxesSubplot:>
```

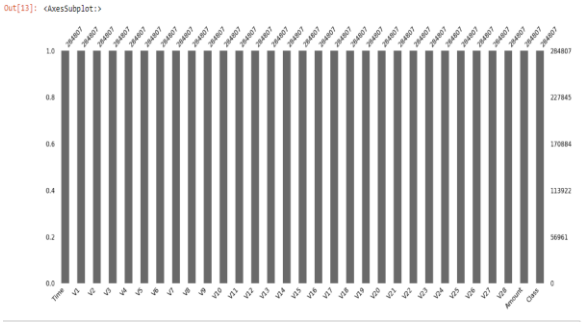


**Figure 3**

```
print(df.describe())
```

```
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  1.759061e-12 -8.251130e-13 -9.654937e-13  8.321385e-13
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01

                V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   1.649999e-13  4.248366e-13 -3.054600e-13  8.777971e-14 -1.179749e-12
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

             ...           V21           V22           V23           V24  \
count        ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean         ... -3.405756e-13 -5.723197e-13 -9.725856e-13  1.464150e-12
std          ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min          ... -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%          ... -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%          ... -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%          ...  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max          ...  2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00

                V25           V26           V27           V28         Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  284807.000000
mean  -6.987102e-13 -5.617874e-13  3.332082e-12 -3.518874e-12      88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01     250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02      22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02      77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01   25691.160000

               Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 31 columns]
```

**Figure 4**

```
In [24]: corr = data.corr()
         fig = plt.figure(figsize = (10, 9))
         sns.heatmap(corr, vmax = 1, square = True)
         plt.show()
```

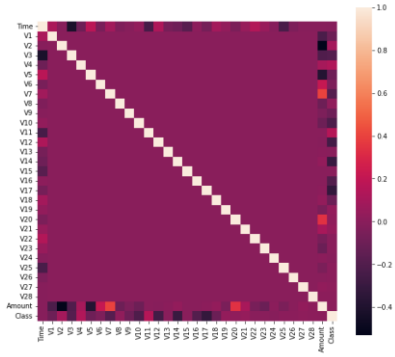


**Figure 5**

```
In [28]: corr = data[["Class", "Amount"]].corr()
         fig = plt.figure(figsize = (10, 9))
         sns.heatmap(corr, vmax = 1, square = True, annot= True)
         plt.show()
```

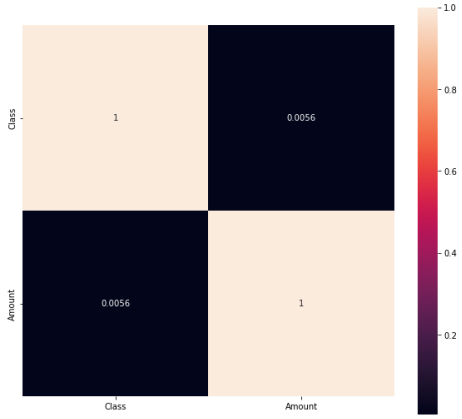



**Figure 6**

```
[5] plt.figure(figsize=(7,5))
    sns.countplot(df["Class"])
    plt.title("class count", fontsize =18)
    plt.xlabel("Record counts by class", fontsize = 15)
    plt.ylabel("count", fontsize  =15)
    plt.show()
```

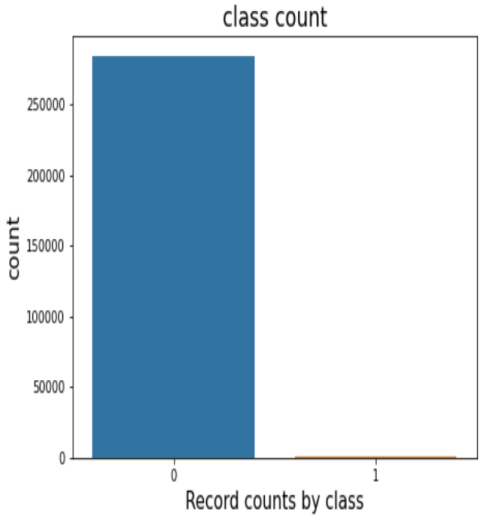
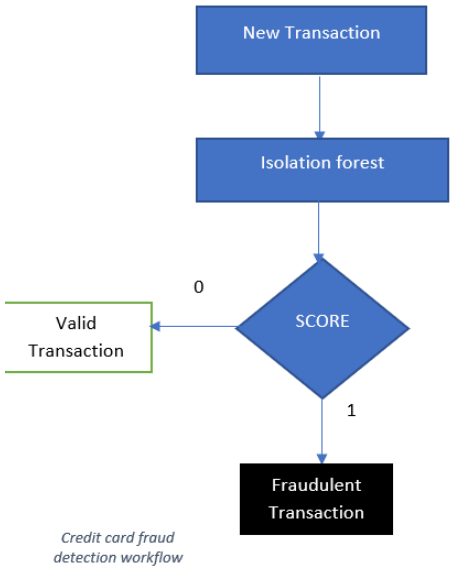



**Figure 7**

*B. Model Implementation*

Isolation Forest: The major technique used in this project is Isolation Forest because the number of fraudulent transactions in the dataset is very few. This algorithm detects anomalies by randomly partitioning not based on information gain as in a decision tree. Randomly select a feature and then create a split value between its maximum and minimum value to create partitions *[1]*. Partitions are created until all points are isolated. In most cases, we limit the number of partitions and tree heights as well. Training datasets are sampled with replacement, but trees are constructed to reduce correlations among classifiers. A random subset of features is considered in the construction of each tree rather than greedily choosing the best split point for each split. RandomForestClassifier is used to construct a Random Forest model for classification [6]. Generally, anomaly detection begins with constructing a profile of what is "normal" and then reporting anything not considered normal as anomalous. In contrast, the isolation forest algorithm does not start by defining "normal" behavior or by calculating point-based distances. Instead, Isolation Forest works by explicitly identifying anomalous points in the dataset and isolating them.




*Credit card fraud detection workflow*

**Figure 8**

Figure 8 shows the workflow of credit card fraud detection. We classify the valid transaction as 0 and the fraudulent transaction as 1. When a credit card is used for a new transaction, the isolation forest algorithm scores the observation as fraudulent or valid by assigning 1 and o respectively.

We use google collab for the model implementation because of the size of the data. I used the python library Sklearn for the model build-up.

I also compared the accuracy of Isolation Forest with the Local Outlier factor which is also an unsupervised anomaly detection method. Local Outlier Factor measures the local deviation of the density of a given sample concerning its neighbor.

```
# Determine no of fraud cases in dataset
Fraud = df[df['Class'] == 1]
Valid = df[df['Class'] == 0]

# calculate percentages for Fraud & Valid
outlier_fraction = len(Fraud) / float(len(Valid))
print(outlier_fraction)

print('Fraud Cases : {}'.format(len(Fraud)))
print('Valid Cases : {}'.format(len(Valid)))
```

```
0.0017304750013189597
Fraud Cases : 492
Valid Cases : 284315
```

**Figure 9**

```
# Setting up the model

from sklearn.metrics import classification_report, accuracy_score
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor

#defining a random state
state =5

#defining the outliers detection method
classifiers = {
    "Isolation forest": IsolationForest(max_samples =len(X_sample),
                                        contamination= outlier_sample,
                                        random_state = state ),
     "Local Outlier Factor" : LocalOutlierFactor(
         n_neighbors =20,
         contamination = outlier_sample)
}
```

**Figure 10**

Many machine learning algorithms used are affected by the bias in the training dataset, the local outlier factor completely overlooks the minority class, and the result gave a precision of 0.05. This is problematic because forecasts are often more crucial for the minority class, that is, the fraudulent class. In other to reduce this problem random Under-sampling technique was also carried out because of the imbalanced dataset.

**UNDERSAMPLING METHOD**

The class of valid transactions was randomly under-sampled by minimizing the size of the abundant class. A new dataset was obtained by keeping all the 492 samples from the fraudulent class and randomly choosing 4000 samples from the plentiful class (valid transaction class).

UNDERSAMPLING TECHNIQUE

```
Fraud_sample = Fraud.sample(n=492)
Valid_sample = Valid.sample(n=1000)
data_sample = pd.concat([Fraud_sample, Valid_sample], axis =0)
```

```
print('Fraud sample Cases : {}'.format(len(Fraud_sample)))
print('Valid sample Cases : {}'.format(len(Valid_sample)))
```

```
Fraud sample Cases : 492
Valid sample Cases : 1000
```

## V. RESULT

Anomaly detection algorithms were built using the Isolation Forest to find transactions, which are, in some sense, different from the usual observations.

The results of isolation forest algorithms and Local outlier factors are given in the figure below. An accuracy score of 99.78% was gotten for the Isolation Forest with a precision of 35% for the fraudulent transaction. The precision, AUC, and Kappa score performance improved when the undersampling was done. Isolation forest accuracy is better than local outlier which suggests that Isolation Forest has a better performance to detect anomalies in credit card transactions than local outlier factors.

Figure 10 is the confusion matrix of our model.

iForest

```
Isolation forest: 635
0.9977704199686103
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    284315
           1       0.35      0.36      0.36       492

    accuracy                           1.00    284807
   macro avg       0.68      0.68      0.68    284807
weighted avg       1.00      1.00      1.00    284807

Cohen Kappa score:  0.3542132292957356
```

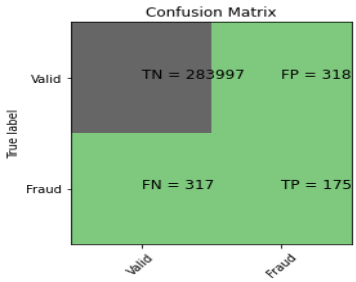
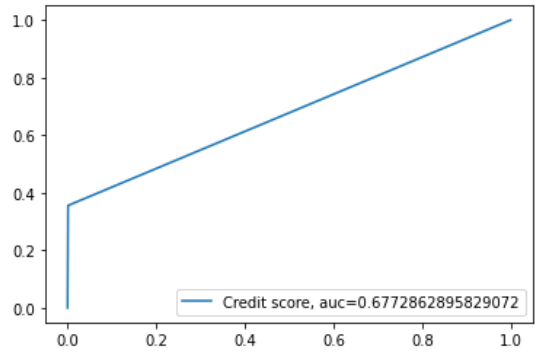



**Figure 11**




LOF

```
Local Outlier Factor: 935
0.9967170750718908
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    284315
           1       0.05      0.05      0.05       492

    accuracy                           1.00    284807
   macro avg       0.52      0.52      0.52    284807
weighted avg       1.00      1.00      1.00    284807

Cohen Kappa score:  0.04911711715198863
```

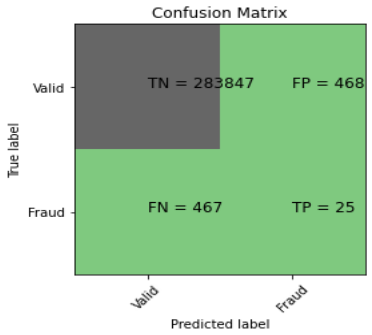
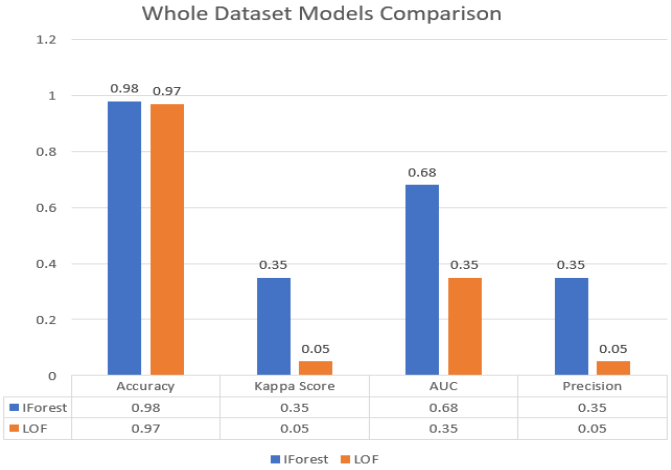


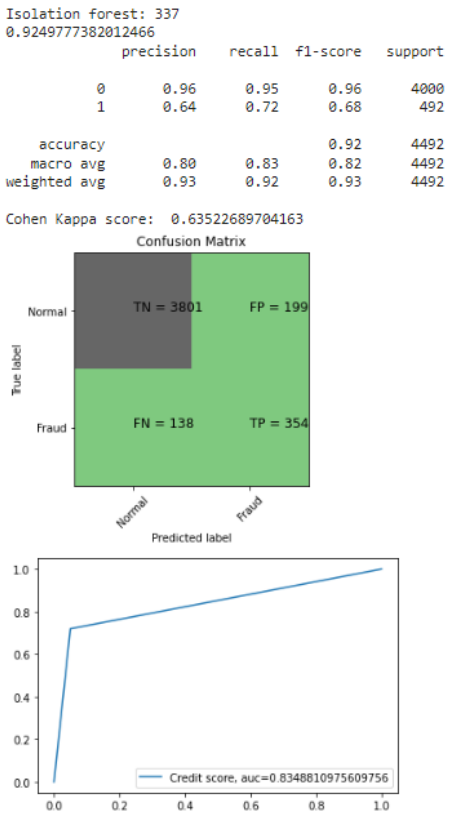

**Figure 12**



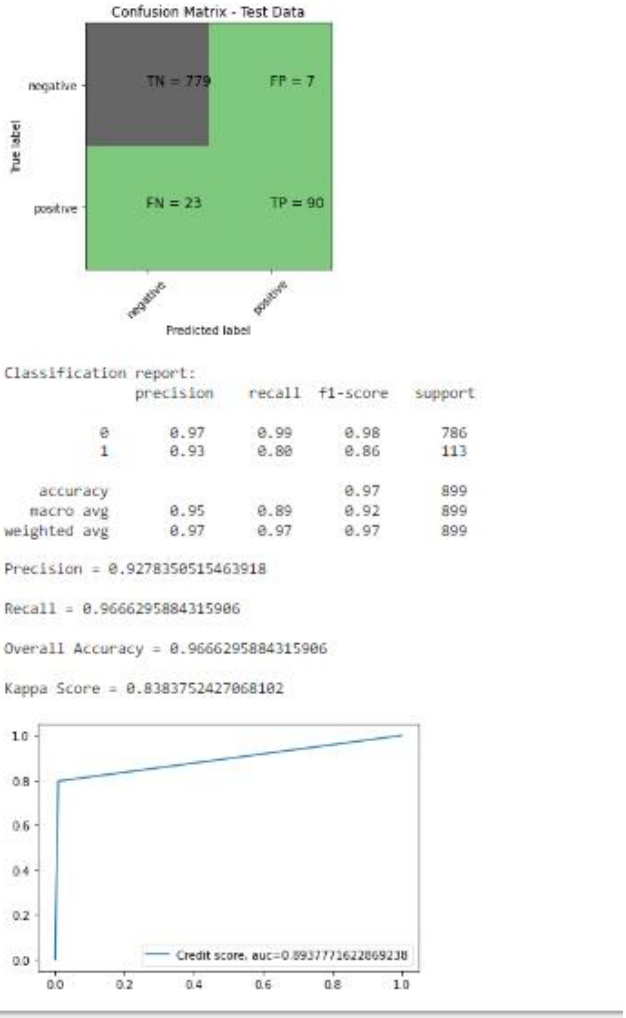

The imbalanced dataset gives accuracies of above 99% for iForest and LOF but in this case, we cannot bank on the accuracy since the precision for the fraudulent transaction is around 35% and 5% respectively. Therefore, we will discuss the result of the undersampling technique.


**Figure 13**

In figure 13, the precision of fraudulent transactions increased to 64% for Isolation Forest and 15% for the Local outliers Factor also 93% for LR classification. The undersampling technique gave better results than the unbalanced dataset.
The summary of the results is given below: The precision, recall, F1 score, and receiver operating characteristic curve (ROC) for iForest and LR.




**IForest**




**Logistic Regression**

Summary Table

| Model | Resampling Method | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Isolation Forest | Random Undersampling | 0.64 | 0.72 | 0.68 |
| Local Outlier Factor | Random Undersampling | 0.15 | 0.17 | 0.16 |
| Logistic Regression | Random Undersampling | 0.93 | 0.80 | 0.86 |

## Conclusion

Machine learning-based credit card fraud detection in the financial sector is not just a trend, but also a requirement for them to set up proactive monitoring and fraud protection procedures. These institutions are using machine learning to cut down on time-consuming manual reviews, pricey chargebacks and penalties, and denials of valid transactions. We have seen the efficacy of unsupervised learning-Isolation Forest in detecting anomalies in transactions and supervised learning namely Logistic Regression in classifying credit transactions as fraudulent or valid. We also used the undersampling method to deal with the imbalanced dataset to improve our model precision.

In future work, we will compare different algorithms, Isolation Forest, and Local Outlier factors with other algorithms using another real dataset. We will also apply SMOTE technique for oversampling to be sure which of the unsupervised or supervised learning algorithms will give the best results.

## References

[1] L. D. a. J. Caporal, "www.fool.com/ascent," The ascent, 21 Sept. 2022. [Online]. Available: https://www.fool.com/the-ascent/research/identity-theft-credit-card-fraud-statistics/.
[2] Brighteron Mastercard, 4 January 2022. [Online]. Available: https://brighterion.com/merchant-fraud-predictions-for-2022-a-pandemic-driven-increase/.
[3] A. O. A. a. S. A. O. John O. Awoyemi, "Credit card fraud detection using Machine Learning Techniques: A Comparative Analysis," in 2017 International Conference on Computing Networking and Informatics (ICCNI), 2017.
[4] S. G. S. P. S. K. G. C. Meenu, "Anomaly Detection in Credit Card Transactions using Machine Learning," International Journal of Innovative Research in Computer Science & Technology (IJIRCST), vol. 8, no. 3, pp. 1-2, May 2020.
[5] R. Shakya, "Application of Machine Learning Techniques in Credit card Fraud detection," UNLV THESES, DISSERTATIONS, PROFESSIONAL PAPERS, AND CAPSTONES, vol. 3454, p. 3, 2018.
[6] V. N. D. e. al., "Credit Card Fraud Detection using Machine Learning Algorithms," Procedia Computer Science, vol. 165, pp. 631-641, 2019.
[7] J. Brownlee, Machine Learning Mastery with Python, 2016.
[8] "Kaggle," [Online]. Available: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud.