Financial Engineering and Risk Management (2024)
Clausius Scientific Press, Canada

DOI: 10.23977/ferm.2024.070610
ISSN 2523-2576 Vol. 7 Num. 6

Credit Card Fraud Detection

Yitong Liu[1,*]

[1]*Statistic Department, Columbia University, 116th and Broadway, New York, U.S.A.*
[*]*Corresponding author: yl5549@columbia.edu*

Keywords: Data visualization; credit card fraud; PCA

Abstract: Nowadays personal credit history is very important, it could affect opening account at bank, renting apartments, etc. It is crucial for the bank to use machine learning techniques to recognize unusual behaviors to avoid unnecessary losses. In China, there are large amount of people who hold their own credit card, but only a few of them would use their card in daily life due to uncertainty of the safety of the credit card payment [ref1]. This project focuses on detecting credit card fraud using a combination of supervised and unsupervised learning techniques. The dataset, sourced from Kaggle, contains over 594,000 credit card transactions from 4,112 unique customers, with key features such as transaction amount, merchant, and customer details. We process the data by cleaning and dealing with its nature of imbalance. By applying different method, we have concluded that that while supervised learning models provided excellent recall and accuracy, there is still room for improvement in reducing false positives, particularly in unsupervised methods. Future work includes oversampling to further balance the dataset and testing the models on larger datasets to enhance generalization.

1. Introduction

As digital transactions proliferate, credit card fraud becomes an increasingly serious threat to both consumers and financial institutions. It is essential to develop the ability to detect fraudulent activities swiftly and accurately. Traditional methods of fraud detection, which rely heavily on manual analysis or rule-based systems, are no longer sufficient due to the sheer volume of transactions and the evolving nature of fraud tactics. On the other hand, machine learning offers a promising solution by automating the detection of fraudulent transactions through data-driven models. However, due to the nature of imbalanced data for credit card fraud detection, it is hard for the models to strike a balance between precision and recall. In this project, we use supervised learning and unsupervised learning to address these challenges. Our goal is to develop a system that accurately identifies fraud while minimizing false positives, which is critical in real-world applications where incorrect fraud detection can inconvenience legitimate customers.

2. Data processing

2.1. Data description

We extract data from Kaggle[ref2]. The dataset is cross-sectional, includes 594643 observations of credit card payment action from 4112 unique customers, and 10 variables (numerical and

categorical, including response variable). Below are some key variables:

• **Customer**: An encryption and unique identifier for each credit card owner, formatted as "C" + numbers.

• **Merchant**: An encryption and unique identifier for each merchant, formatted as "M" + numbers.

• **Age**: The number of years that the credit card has been held, ranging from 0-6, with 7 or "U" indicating unknown.

• **Gender**: The gender of each credit card owner.

• **Amount**: The amount of currency (in dollars) for each credit card transaction.

• **Category**: The category of action (e.g., Transportation, Food) for each credit card transaction.

• **Fraud (Response Variable)**: A Boolean value indicating whether the credit card transaction is classified as fraud.

2.2. Data Visualization

As shown in Figure 1, after pulling in the data, we explore the data by visualize it and it has revealed several important insights into the distribution and behavior of variables, particularly in relation to fraudulent transactions: We discover an overwhelming proportion of non-fraudulent transactions. This is typical in fraud detection problems, where legitimate transactions vastly outnumber fraudulent ones. This problem could lead to the result of if we let the system approve all the credit card, the false rate would be still very low. Another problem is most of the transactions in the dataset involved relatively low amounts, whereas fraudulent transactions were generally associated with significantly higher amounts. We also discover some minor problems that need to utilize data cleaning to solve it in the next part.

2.3. Data Cleaning

Based on the analysis of the dataset, we focus on the key variable: amount. After some research, we decide to transform the amount into 7 levels to avoid the data being too skewed. [ref3] On the other hand, since even though the Dataset doesn't contain NAs, but variables like gender, age include "U" as unknown. Since the quantity is low, we removed those observations. Then we create new predictors by recording the total number of transitions for each credit card holder, merchant, and transition category, and apply those numeric variables to each observation. After this, since PCA and k-means only work inherently work with numerical data. We map all the predictors into numeric. We have selected the predictor using one-hot-encoding. We dropped irrelevant variables and variables that are hard convert to numeric such as the the zip code of Merchants, the zip code of card holders and step. We also realize that the information in categorical data like "category" will lose after transforming them into numeric data. So, we applied one-hot encoding to those categorical variables ('category', 'gender'), which turns them into numerical variables with 0-1 values. Through these steps of data wrangling, we got our final data set. Our final dataset is cross-sectional with 594128 observations, 10 predictors and no missing data (without all the one-hot-encoding variable). This dataset contains only numeric variables, except the response variable "fraud", which is appropriate for dimension reduction and cluster algorithms like PCA and K-mean. Response Variable 'fraud' contain only two unique values: "F" means this transition is fraud, "NF" means not fraud.

3. Modelling

3.1. Dimension Reduction-PCA

Principal Component Analysis, also known as PCA, is a statistical technique used to simplify the

complexity in high-dimensional while maintaining trends and patterns. Here, we choose this to achieve the goal of dimension reduction. The primary benefit of PCA is that it becomes an essential process in calculating the number of clusters as well as providing a conceptual mathematical model to model the structure of the sets once we have identified these principal components from the data. [ref4] Here are the steps we generate PCA on our data: **Steps to generate PCA:**

1) **Data preparation:** Standardize the data if they are not on the same scale.

2) **Covariance matrix:** Calculate the covariance matrix to understand how each variable relates to another.

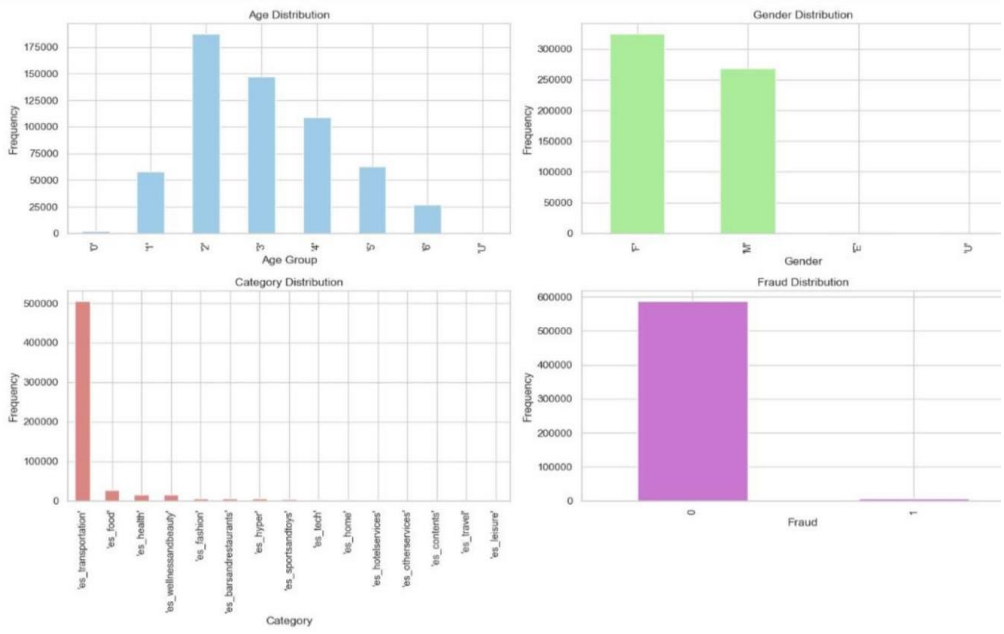

Figure 1: Data visualization

3) **Eigenvalue decomposition:** Perform eigenvalue decomposition of the covariance matrix to identify principal components.

4) **Select Components:** Choose the principal components to retain based on explained variance.

5) **Data transformation:** Transform the data into a new subspace using the selected principal components.

6) **Outlier detection:** Detect and manage outliers in the transformed data.

7) **Visualization:** Visualize the results of the PCA using plots to observe patterns and relationships.

In Figure 2 and Figure 3, from these steps, we have generated both 2D and 3D PCA. Firstly, we standardized the data using Standard Scaler () and performed PCA on it. A data frame was created with the PCA results, and the fraud (target column) was appended into the PCA data frame. We then visualized the PCA results using a scatter plot.

From the 2D PCA scatter plot, a clear pattern emerges where the blue 'x' marks, representing fraud cases, are clustered on the right side of the plot. This indicates that there is a region where fraud cases are more prevalent. The red circles, representing non-fraud cases, are spread throughout the plot but are particularly dense on the left side, suggesting that non-fraud cases are more common in that region.

However, there is significant overlap between the two classes, especially in the middle of the plot along the PC1 axis. This overlap indicates that while the first two principal components provide some level of separation between fraud and non-fraud cases, they may not be sufficient to create a perfect classifier. Additionally, the spread along the PC1 axis is greater than along the PC2 axis, which is expected since PC1 accounts for the most variance in the data. This suggests that PC1 is more significant in differentiating between fraud and non-fraud cases than PC2.

The results imply that additional features or alternative methods may be required to adequately separate the two classes.

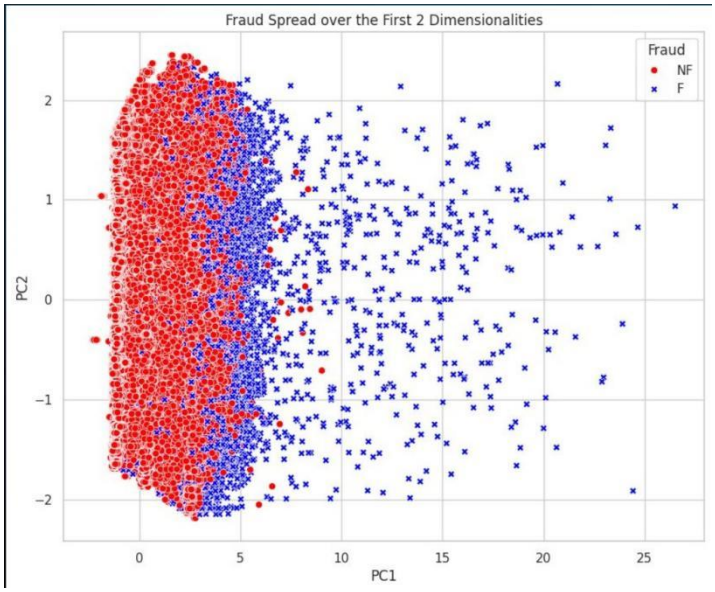

Figure 2: 2D PCA

Thus, we create 3D PCA to accommodate the previous model's flaw through similar steps for 2D PCA. From Figure 3, we could conclude that there is some clustering (red points are more concentrated in certain regions), there is also a significant overlap between red and blue points, suggesting that simply using the first three principal components might not be sufficient to cleanly separate fraudulent from non-fraudulent cases. Besides, higher density of non-fraud cases (blue points) spread out across the space, with fraud cases (red points) more sparsely distributed. This could reflect the common reality in fraud detection where fraudulent cases are less frequent than legitimate ones.

To compare 2D and 3D PCA, we calculated the explained variance for both. From the results, we observe that the explanation rate for 2D PCA is lower than that of 3D PCA. However, 2D PCA is easier for non-professionals to understand, both in terms of the graphical representation and the underlying algorithms. As shown in Figure 4, we prefer the 3D PCA in this case because it has a higher rate of explained variance (72.45%) compared to the 2D PCA (53.29%).

3.2. Feature Imbalance Weight

Recall that the dataset has the problem of extremely imbalanced weights for fraud and non-fraud cases. We chose under-sampling to address this issue. This approach involves keeping all available fraud transactions (since the number of fraud cases is much fewer than non-fraud), while under-sampling the non-fraud transactions to approximately the same number. After under-sampling, we obtained 7,200 fraud data points and 9,743 non-fraud data points.

Model: 75% of the data was used for training and 25% for testing, with the 75% training data also used for cross-validation. As shown in Figure 5, the models employed include Flexible Discriminant Analysis (FDA), extreme Gradient Boosting (XGBoost), and Linear Discriminant Analysis (LDA).

Results:

• Recall score for FDA model on test data: 0.9954853273137697

• As shown in Figure 6, Recall score for XGBoost on test data: 0.9811165845648604

• Recall score for LDA model on test data: 0.9830699774266366 Since XGBoost allows for easy calculation of feature importance, it helps identify the most influential variables for fraud detection. Visualization of these features can provide insights into the model's decision-making process,

highlighting which factors are most strongly associated with fraudulent transactions. In addition, from figure 6 we can see that XGBoost model performs excellent work in identifying fraud cases with high precision and recall, while also correctly identify non-fraudulent cases with minor error. We also visualize the feature importance from XGBoost to understand model behavior and evaluate model performance. From the results in figure 7, we observe that the most important feature

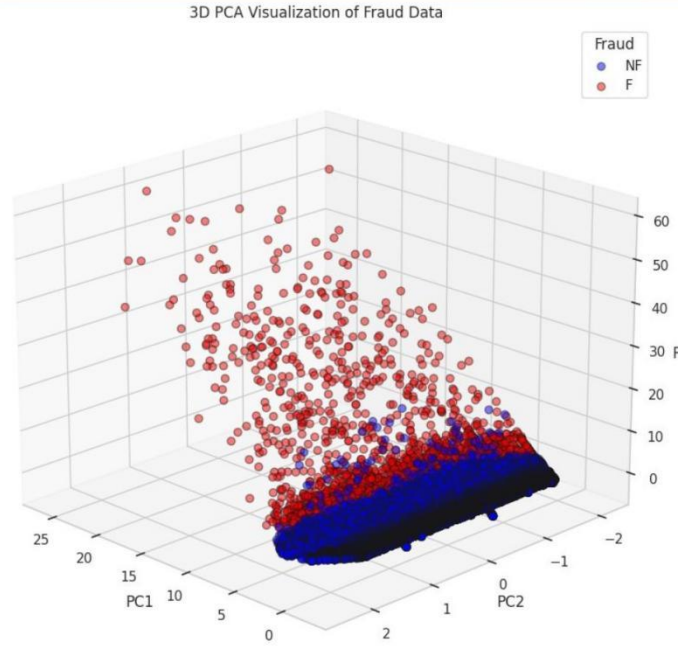

Figure 3: 3D PCA

```
# Calculate and print the explained variance
explained_variance_2d = pca_2d.explained_variance_ratio_.sum() * 100
explained_variance_3d = pca_3d.explained_variance_ratio_.sum() * 100

print("Total Variance Explained by First 2 Principal Components: {:.2f}%".format(explained_variance_2d))
print("Total Variance Explained by First 3 Principal Components: {:.2f}%".format(explained_variance_3d))
```

```
Total Variance Explained by First 2 Principal Components: 53.29%
Total Variance Explained by First 3 Principal Components: 72.45%
```

Figure 4: Result for PCA

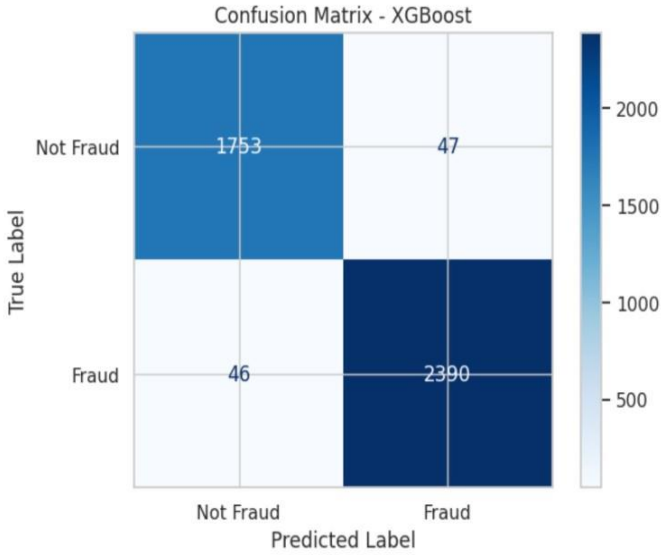

Figure 5: XG-Boost Confusion Matrix

Precision: 0.974
Precision: 0.981
Recall: 0.974
Recall: 0.981
F1-score: 0.974
F1-score: 0.981
Accuracy: 0.978

Figure 6: Model output

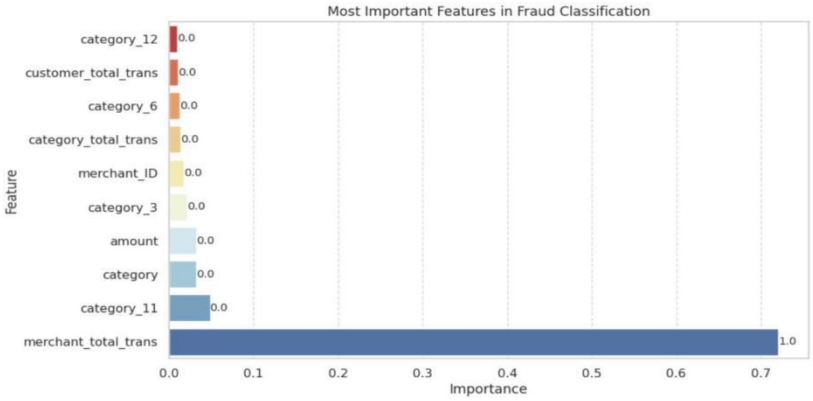

Figure 7: Feature Importance for XGBoost

It is merchant _total trans, followed by category 11 (fashion), and then category and amount. This is reasonable, as fraud is often detected by the total number of merchant transactions in real-world scenarios.

3.3. K-Means

To explore this method more deeply, it is essential to understand the definition of K-Means. K-Means is an unsupervised learning algorithm for clustering that partitions data into *K* distinct groups based on feature similarity. The aim is to leverage the strengths of both clustering and classification to enhance the accuracy and efficiency of fraud detection. [ref5] Below are the steps we applied to our data for K-Means:

1) Determine the optimal *K* value to perform K-Means clustering using the elbow method.

2) Apply K-Means to partition our data into *K* clusters, where each data point is assigned to the nearest cluster center.

3) Isolate outliers by calculating the squared Euclidean distance post-clustering.

4) Assume the clusters contain the true labels of fraud behavior, and the outliers represent the suspect behaviors. Then, plot a confusion matrix to evaluate the performance of K-Means. 5. Analyze cluster compositions to identify patterns and categorize transactions.

Firstly, to determining the best number of clusters, we use WCSS, Within-Cluster Sum of Squares.

$$WCSS = \sum_{i=1}^{K} \sum_{x \epsilon c_i} \|x - \mu_i\|^2 \quad (1)$$

By visualizing this, we have Figure 8. We can see that the rate of decrease of WCSS significantly slows down after $k = 4$, so we choose $k = 4$ as the optimal number of clusters. Then, we assigned the dataset into 4 clusters:

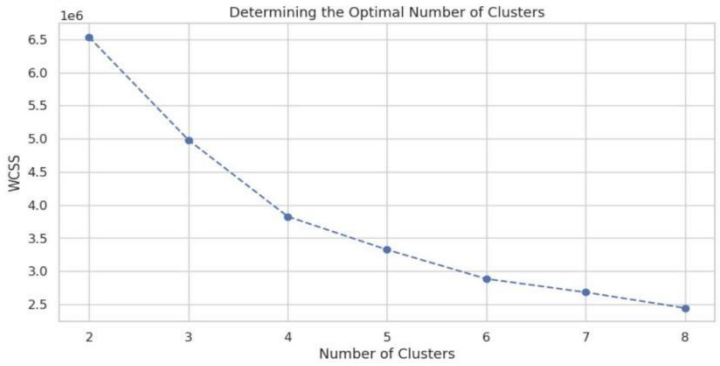

Figure 8: WCSS plot for K-Means

- **Cluster 0**: Comprises 275,445 observations; the majority are non-fraudulent, with only 15 identified as fraud.
- **Cluster 1**: A significantly large cluster with 229,245 observations, all non-fraudulent.
- **Cluster 2**: Consists of 82,165 non-fraud transactions and a substantial number of fraud cases, with 5,109 identified as fraud.
- **Cluster 3**: The smallest yet most critical group with 2,164 observations, having a high concentration of fraud with 2,076 fraud cases.

As shown in Figure 9, after running the K-Means clustering algorithm, some data points may be far from the cluster centroids. These are considered outliers. The outliers are determined based on their distance from the cluster center. The threshold for identifying an outlier is set as 1.5 times the interquartile range (IQR) above the third quartile. This is a statistical method commonly used to detect data points that deviate significantly from the rest of the dataset. We assume that the clusters are identified by K-means contain the true labels under "fraud", while outliers are the "suspect behaviors". We have generated confusion matrix for this, which helps evaluate how well the clustering performed in distinguishing between fraud and non-fraud cases, particularly with respect to sensitivity (ability to detect fraud) and precision (avoiding false positives).

To better evaluate the performance of K-Means, we have generated additional metrics. From Figure 10, high sensitivity suggests a strong ability to detect fraud. Precision is low, implying a high number of FP. The model is highly accurate in identifying non-fraud cases as most of the records are non-fraud. Cohen's Kappa indicates moderate agreement; there's room for model improvement. Note that this only captures an initial view of the data distribution.

Therefore, to analyze the internal structure and composition of three important clusters generated by the K-means algorithm, we have broken down the key points as follows:

Since Cluster 1 does not contain any fraudulent transactions, we only inspected the inner structures of the other three clusters by examining which transaction category most fraudulent records are from, and obtained the following findings:

- **Cluster 0** is primarily non-fraudulent with a very low average transaction amount, and the fraudulent transactions all belong to the category "leisure".
- **Cluster 2** has a mix of fraud and non-fraud cases but contains a high number of fraudulent cases. The average transaction amount for fraud cases in this cluster is significantly higher than that of non-fraud cases, and the fraud cases span various categories.
- **Cluster 3** stands out as a high-risk cluster due to the high proportion of fraud and high transaction values. The non-fraud cases are mostly from the category "travel," which is reasonable. The fraud cases are mostly distributed across the categories "hotel services," "travel," and "fashion."

Figure 9: Confusion matrix for “suspect behaviors” and actual fraud

```
Sensitivity: 0.781
Precision: 0.106
F1: 0.187
Accuracy: 0.918
Kappa: 0.169
```

Figure 10: Result for additional metric to evaluate performance of K-Means

3.4. Random Forest on cluster dataset

Random Forest is a popular and powerful supervised learning algorithm that can be used for both classification and regression tasks. It works by building a collection (or “forest”) of decision trees during training and outputs the class that is the mode of the classes (for classification) or the mean prediction (for regression) of the individual trees. Each tree is trained on a random subset of the data and features, which helps prevent overfitting and makes the model more robust. In this case, Random Forest selects the best feature rather than the most important feature among a random subset of data resulting in a better model. Thus, having a binary classification of fraud i.e., positive case (value 1) and non-fraud i.e., negative case (value 0) for the target category in the transaction amount. [ref6] Here, based on K-means data, we ran a Random Forest model with n estimators=100 and random_state=42, and visualized the results using a 3D plot. In Figure 11 and Figure 12, we could see that the confusion matrix shows a high precision of 0.998 and recall of 0.999 on the test data for non-Fraud. Additionally, the precision for Fraud is 0.901, and the recall is 0.793.

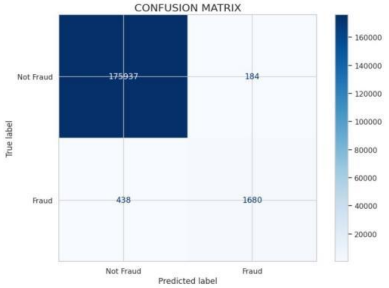

Figure 11: Confusion matrix

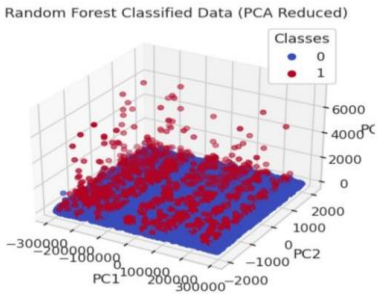

Figure 12: 3D visualization

4. Conclusion

From the previous part, we could conclude that 3D PCA performs better than 2D PCA, 3D PCA has much higher explanation rate, but there are still some points covering each other. Oversampling on the dataset according to number of frauds could solve the problem of unbalanced weights in dataset and performs well by receiving high recall and precision when applying XGBoost, LDA, FDA. In addition, K-means clustering overall performs fairly on our data, as the vast majority of the records are non-fraudulent. But there still are a large number of false positives, and Cohen's Kappa indicates that there's room for model improvement. Finally, our models are able to achieve a 0.997 validation accuracy and 0.978 on equal sampled dataset, which is very impressive. We can try our model on large dataset to further test its prediction power.

References

[1] Khyati Chaudhary, Jyoti Yadav, Bhawna Mallick. A Review of Fraud Detection Techniques: Credit Card. International Journal of Computer Applications, Volume 45, No.1, pp. 39-44, May 2012.

[2] Kaggle. Synthetic data from a financial paymentsystem. Retrieved from https://www.kaggle.com/ datasets/ ealaxi/ banksim1/ data

[3] https://www.kaggle.com/code/turkayavci/fraud-detection-on-bank-payments https://www.kaggle.com/ code/andr adaolteanu/ ii- fraud- detection- classify-cluster-pca

[4] Muhammad Zohaib Khan, Sarmad Ahmed Shaikh, Muneer Ahmed Shaikh, Kamlesh Kumar Khatri, Mahira Abdul Rauf, Ayesha Kalhoro, and Muhammad Adnan, The Performance Analysis of Machine Learning Algorithms for Credit Card Fraud Detection, International Journal of Online and Biomedical Engineering (iJOE), vol. 19, no. 3, pp. 82-98, 2023. doi:10.3991/ijoe.v19i03.35331.

[5] Keshetti Sreekala, Rayavarapu Sridivya, Nynalasetti Kondala Kameswara Rao, Raman Kumar Mandal, G. Jose Moses, and A. Lakshmanarao, A Hybrid Kmeans and ML Classification Approach for Credit Card Fraud Detection, in Proceedings of the IEEE Conference, 2020.

[6] Jonnalagadda Vaishnave, Priya Gupta, and Eesita Sen, Credit card fraud detection using Random Forest Algorithm, International Journal of Advance Research, Ideas and Innovations in Technology, vol. 5, no. 2, pp. 1797-1801, 2019.