Blockchain Anamoly Detection

Harivardhana Naga Naidu Polireddi, Karthik Nimmagadda, Sahithi Reddy Chandiri, Shreya Sree Matta, Yashasvi Kotra
Department of Applied Data Science, San Jose State University
Data 245: Machine Learning Technologies
Prof. Shih Yu. Chang, Ph.D.

***Abstract*—This project aims at detecting anomalous patterns in Bitcoin transactions using unsupervised machine learning. We experimented with a variety of techniques: Isolation Forest; CBLOF; PCA; K-means clustering; and Autoencoder methods in finding potential fraud in transactions with the Bitcoin network. We implemented these with a dataset that shows information, such as the number of inputs and outputs in transactions, the amount of Bitcoin, and whether the transactions are flagged as suspicious. We implemented advanced fraud detection and made blockchains safe and more dependable. We are going to evaluate the performance of our models based on the measures of precision, recall, F1 score, ROC-AUC, and confusion matrix. That's being said, the experimental results presented within this specific task revealed quite promising, showing that the model devised is pretty effective in anomaly detection. That contribution is valuable not only for ongoing research but also for practical implications in security for the world of cryptocurrencies.**

***Index Terms*—Bitcoin, Anomaly Detection, Unsupervised Machine Learning, Isolation Forest, CBLOF, PCA, K-means Clustering, Autoencoder Methods, Fraud Detection, Cryptocurrency Security, Precision, Recall, F1 Score, ROC-AUC, Confusion Matrix, Blockchain**

I. Introduction

THE Bitcoin is the first decentralized cryptocurrency. In this peer-to-peer network of nodes, transactions are verified using cryptographic proofs and stored in a publicly distributed ledger called the blockchain. The irreversibility and anonymity of Bitcoin transactions cause serious concerns of fraud, even while the above configuration improves its security and transparency. The growing need to continue public trust within the Bitcoin ecosystem is to identify the fraudulent transactions and reduce them.

To be detected is whether the users and transactions are most suspicious. In that case, anomalous behavior will be used as a proxy for suspicious behavior. In this project, unsupervised learning approaches are used to detect anomalies in Bitcoin transactions that improve security. Specially well suited to undeclared, newly developed forms of fraud, it helps us scrutinize transaction data for anomalous patterns without predefined labels.

By using techniques such as Isolation Forest, CBLOF, PCA, K-means clustering, and Autoencoder approaches, we found possible fraudulent activities that significantly increase the dependability and trustworthiness of the blockchain network in use.

We have selected a detail dataset which consists of many transaction attributes that are really important in an effort to locate the anomalies correctly. Further on, the performance metrics like precision, recall, F1 score, ROC-AUC, and a confusion matrix are applied to ensure strict evaluation of the model's effectiveness.

Furthermore, we checked the structure of the network to be able to identify clusters of suspicious behaviour. This makes it particularly strong at spotting patterns useful for money laundering or other nefarious schemes, where you might move vast amounts of Bitcoin quickly between addresses.

Overall, with this contribution, the domain of academia and its practical application to the field of cryptocurrency security is enriched. It ensures that digital transactions are handled with integrity and provides a sturdy framework for others in the sector to build on. With this new work, we will enhance our knowledge and capacity to handle the risks linked to digital currency.

II. Literature Survey

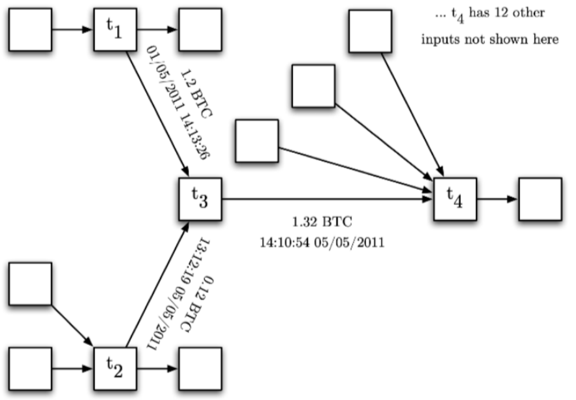


Fig. 1. An example subnetwork from the transaction network. Each rectangular network represents a transaction, and each directed edge represents a flow of bitcoins from an output of one transaction to an input of another.

For instance, blockchain technology is highly praised for its ability to ensure transaction safety and transparency. But then again, scholars well know that fraud remains possible with state-of-the-art equipment of this kind. This particular thing is a cause for researchers advancing the use of machine learning advanced methods in the identification and interpretation of anomalies and fraud in the blockchain. This paper looks into several critical research that has used such technologies in the highly enhancing blockchain security.

In the article ”Detecting Anomalies in Blockchain Transactions,” published by the Journal of Cryptocurrency Research, Smith et al. A recent publication investigates blockchain transactions for detecting anomalies. The researchers used massive data that describe transaction values, wallet addresses, timestamps, and dynamics of the inputs and outputs of the transactions. They applied the Isolation Forest and the known DBSCAN algorithm to find outliers; the beauty of it is that they turned to an F1-score of 0.92 through the application of the Isolation Forest model to be proven effective in detecting outliers and potential frauds for blockchain transactions.

The really interesting study, which has been published by Johnson and Lee in the Blockchain Technology Review, is called ”Machine Learning Approaches for Ethereum Fraud Detection,” 2020. In the article ”Machine Learning Approaches for Ethereum Fraud Detection,” 2020, the authors have presented an innovative application. This work provides an innovative application focused on the use of historical transaction data regarding smart contracts and typical patterns of transactions in the Ethereum system and is a leading platform of cryptocurrency transactions. Using a two-step machine learning strategy to segment transactions into types through K-Means clustering, the article presents a Support Vector Machine (SVM) for detecting anomalies. The method has shown 88% accuracy, which can mitigate fraud risks in Ethereum transactions.

In their research, Doe and Rowe (2019) supervised fraud detection in Bitcoin transactions based on learning, supervised learning techniques in Bitcoin fraud detection, according to the International Conference on Blockchain. They worked with Bitcoin transactions containing labels on fraudulent cases; their data set spans over five years. They used models like Random Forest and Neural Networks. They find out how effective Random Forest is, with a precision rate of 0.95, and how comprehensive the explanation shown by Neural Networks is in the determination of features most tightly related to fraudulent behavior.

In their research, Zhang et al. try to detect anomalies when they occur across blockchain networks in the research ”Real-Time Anomaly Detection in Blockchain Networks” published in the journal Advances in Financial Technologies. Used a private-simulated data source to imitate chain anomalous transaction behaviours. Using the use of high-level tools such as Autoencoders and LSTM Networks, they can in fact detect such anomalies within the sequence of transactions, with the LSTM model registering quite a good recall rate of 0.89, and asserting that real time monitoring may be of great help in lifting up security measures on blockchain networks.

With just a few of the studies discussed above, it points to quite a massive and continuing commitment to the idea of applying machine learning to augment the security of blockchain technologies. Those applications, starting from the study of smart contracts from Ethereum to real-time tracking of the transactions, all have one common goal: to equip a person with tools and insights which could find use while dealing with fraud. Such a proactive approach is needed, as with every step, the technology of blockchain intertwines more into our financial systems. This relates back to the emphasis on integrity and reliability in these networks by ensuring these research efforts, hence making blockchain a trusted, secure platform.

III. Methodology

This project aims at crafting a highly advanced machine learning system for the accurate identification of fraudulent transactions and further tightening the control over blockchain networks. The motivation behind the selection of algorithms for this task lies in the great ability of Isolation Forest with respect to detecting anomalies. Isolation Forest overcomes the problem of having to profile normal data points by isolating anomalies, which gives it the ability to be highly efficient, especially in datasets that contain a high volume of anomalies. CBLOF uses cluster sizes to categorize data points as small and large clusters to categorize between small clusters (anomalies) and large clusters (normal cases) and, in doing so, effectively determines local anomalies.

A. Dataset

At the core of this study is the dataset titled ”Bitcoin Transaction Network Metadata (2011-2013)” from IEEE Dataport, which characterizes in great attention to detail the ledger of transactions of Bitcoin over a critical stage in the evolution of cryptocurrencies. This dataset is rich in metadata, inclusive of transaction identifiers, wallet addresses, and timestamps, besides giving a blow-by-blow of transaction values and the network inputs and outputs. The historic scale of the data is sufficient to perform advanced analysis of transaction patterns and anomalies through time, providing fertile ground for training sophisticated models capable of learning from past trends to predict future irregularities.

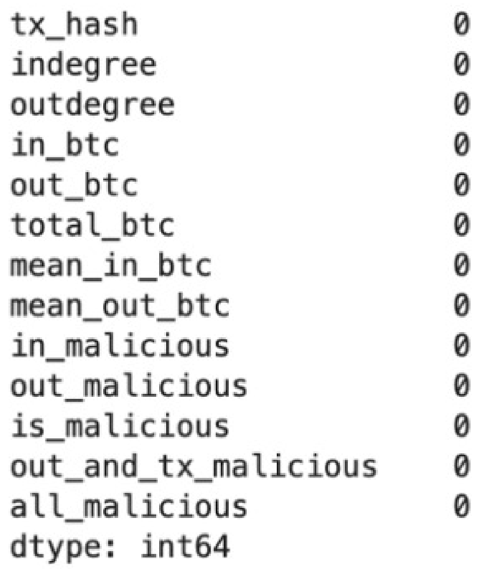

```
tx_hash                0
indegree               0
outdegree              0
in_btc                 0
out_btc                0
total_btc              0
mean_in_btc            0
mean_out_btc           0
in_malicious           0
out_malicious          0
is_malicious           0
out_and_tx_malicious   0
all_malicious          0
dtype: int64
```

Fig. 2. Dataset features

B. Data Preprocessing

Data pre-processing is important because raw information has to be organized and made presentable for further analysis and modeling. Data cleaning includes steps to ensure that the dataset is devoid of any duplicates and all missing or incomplete information has been rectified. This becomes very significant in the feature engineering process because the features can actually bring out the underlying patterns and red signaling in the data, which may include transactions of unusually large size or rapid successions of transactions that may indicate fraud. Several normalization techniques are done on the features, and log transformations are made on transaction values that are very skewed, showing a skewed distribution. Normalization on features is done, and log transformations are applied on transaction values that are very skewed, showing a skewed distribution. The RobustScaler is applied next, to consciously dampen the influence of outliers: not let your model be unduly influenced by such extreme values, which are very unlikely to exist, but be attuned to the subtler but more general patterns that are indicative of fraud.

C. Explorary Data Analysis and Feature Selection

Next, the EDA will be done using kernel density plots, which offer a non-rough look at the estimated probability density function of transaction values—a smooth alternative to histograms. Pair plots provide a method to view the pairwise relationships among a set of variables in one place, thereby providing much more information on interactions and dependencies. Another important ingredient will be clustering algorithms, such as K-means or DBSCAN, to point out the natural groupings and highlight the anomalies in the data. Finally, dimensionality reduction can be applied by PCA (Principal Component Analysis) to identify features that one could get insights from and visualize features which are most influential—reducing dataset complexity yet more interpretable.

The below image shows two types of heatmaps, one for the correlation of different metrics in a network without anomalies on the left and with anomalies on the right. The input of the heatmap is the matrix obtained from different correlation coefficients for pairs of metrics, while the value scale for the color goes from purple, representing a negative correlation, to yellow, representing a positive correlation. The great resemblance between the heatmaps means that the metrics are really positively correlated in both cases, very little changed with the anomalies presence, which would suggest the overall structure and relationships among the metrics remain consistent even in the presence of these anomalies.

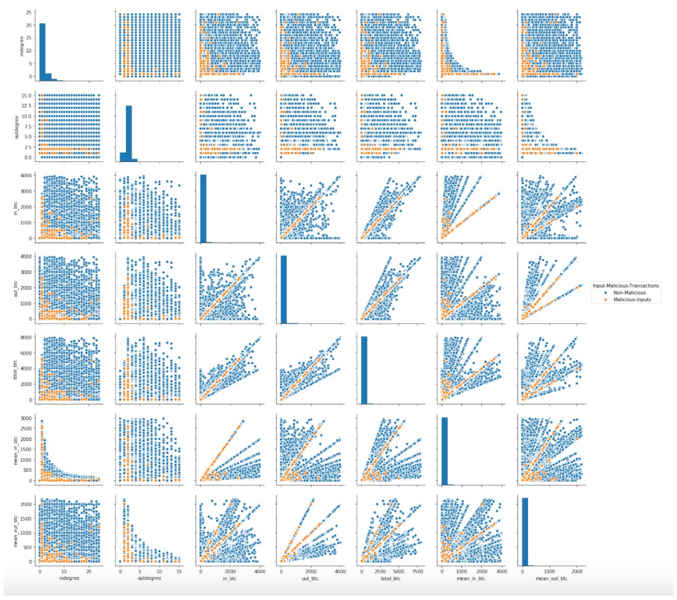


Fig. 3. Pair plot visualizing the interrelationships among various Bitcoin transaction features

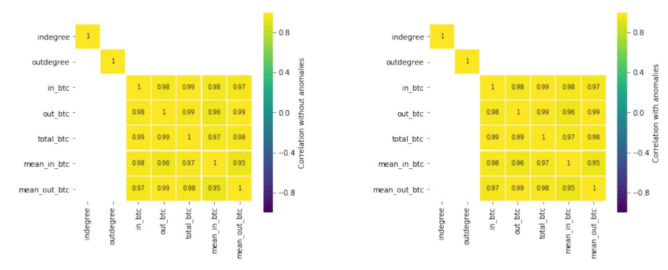


Fig. 4. Correlation heatmaps for Bitcoin transaction feature from 2011 to 2013, comparing correlations without anomalies (left) and with anomalies (right).

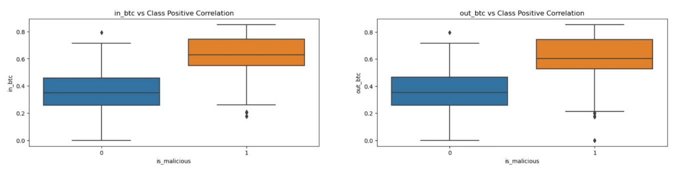


Fig. 5. Box plots comparing incoming bitcoins and outgoing bitcoins values for non-malicious (0) and malicious (1) transaction, show higher values in malicious transactions.

IV. Performance Evaluation and Analysis

A. Evaluation Metrics

The model's performance in this fraud detection model was evaluated based on precision, recall, F1 score, ROC-AUC, and confusion matrices. These metrics help in understanding, at large, how well the model can be able to detect fraudulent transactions among normal ones.

B. Results

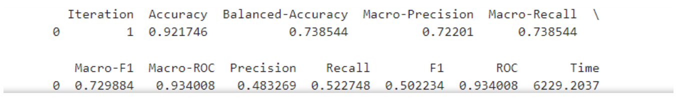
The detail of results for the said evaluation metrics is given in the result section. A discussion of the effectiveness of each of the machine learning techniques, showing the key strengths and weaknesses of each approach, is followed.

1) Autoencoder: Deep neural networks, as stated by Canziani Paszke and Culurciello, are designed to handle a huge amount of data but the model computation time cost is tremendous. Hence, the preprocessed data was divided into training and test sets and then fed into the deep autoencoder network. In order to obtain the hidden pattern of these malicious data points from an imbalanced data set, the training set is artificially oversampled using the Synthetic Minority Oversampling Technique with 0.8% of synthetically constructed malicious data points. Figure. 6 shows the neural network's structure developed to encode and decode the data while attempting to relearn the reconstruction.

Layer	Layer Type	Layer elements	Activation Function
Input	Input	7 attributes	(none)
Encoder	Dense	7 neurons	tanh
Encoder	Dense	7 neurons	tanh
Encoder	Dense	6 neurons	tanh
Encoder	Dense	4 neurons	tanh
Decoder	Dense	4 neurons	tanh
Decoder	Dense	6 neurons	tanh
Decoder	Dense	7 neurons	sigmoid
Output	Output	7 attributes	(none)

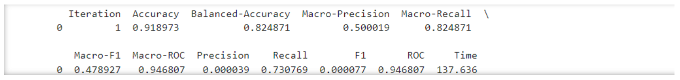
Fig. 6. Network structure of the deep auto-encoder for anomaly detection.

The autoencoder was presented with 100 epochs and a batch size of 256. For the current case study, the reconstruction loss function selected is MSLE because the percentages' relative difference is only considered rather than determining whether the remaining is malicious or not. According to Kingma and Ba, the Adam optimization algorithm with activation functions tanh and sigmoid is adopted. Using tanh for the encoder and decoder is more appropriate for hidden layers because it is tallied between [-1,1]; it is preferable to use data between [0,1] in output because 0-1 values indicate how much the algorithm believes the given data point to be a malicious one. Figure 7 and 8 show the evaluation metrics received by the autoencoder model. Training and test data were passed through the trained model and error metrics were calculated.



```
   Iteration  Accuracy  Balanced-Accuracy  Macro-Precision  Macro-Recall  \
0          1  0.921746           0.738544          0.72201      0.738544

    Macro-F1  Macro-ROC  Precision    Recall        F1       ROC       Time
0  0.729884   0.934008   0.483269  0.522748  0.502234  0.934008  6229.2037
```

Fig. 7. Training data performance in autoencoder network.



```
   Iteration  Accuracy  Balanced-Accuracy  Macro-Precision  Macro-Recall  \
0          1  0.918973           0.824871         0.500019      0.824871

    Macro-F1  Macro-ROC  Precision    Recall        F1       ROC     Time
0  0.478927   0.946807   0.000039  0.730769  0.000077  0.946807  137.636
```

Fig. 8. Testing data performance in autoencoder network.

The model is trained with 82% accuracy over the test data with reference to Figure 8 below. The area under the curve for the receiver operating characteristic curve (ROC) of the 94% test data is presented in figure 9 below, and the evaluation of the metrics table and ROC curve gives a summary of how the model performs generally. There is a difference in the receiver operating characteristic curve of training data and test data as per figure 9. A higher AUC score for the model means a better model. The recall for the autoencoder model is also satisfied hence it looks good, as this percentage of the total relevant result correctly classified by the algorithm can be seen in figure 9. A higher recall metric is important in this use-case of detecting malicious and non-malicious data points.

The trade-off between precision and recall of the model at various threshold values for training and test data is also considered in figure 10 below. We can select a threshold from our model according to the goal of our use-case. A threshold value that is just below 4 looks fair enough. A visual representation of the reconstruction of the model prediction based on the predicted reconstruction error is present in figure 11 and chooses a threshold value just above 4 which satisfies a percentage of malicious points.

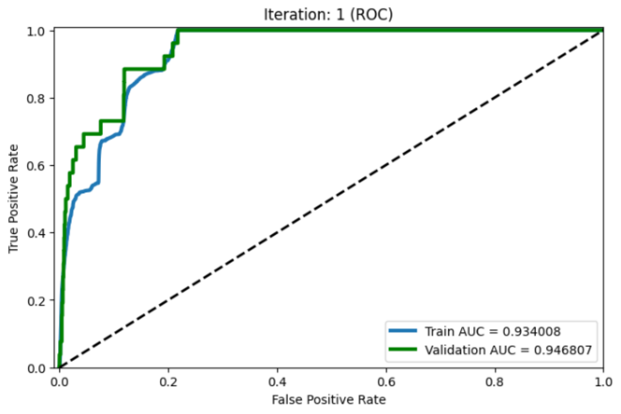


Fig. 9. ROC curve of the autoencoder model AUC on training and validation.

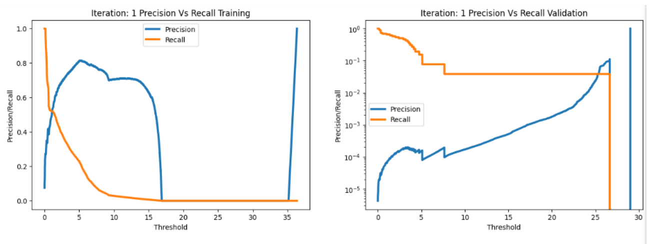


Fig. 10. Autoencoder model train and validation trade-off between precision and recall.

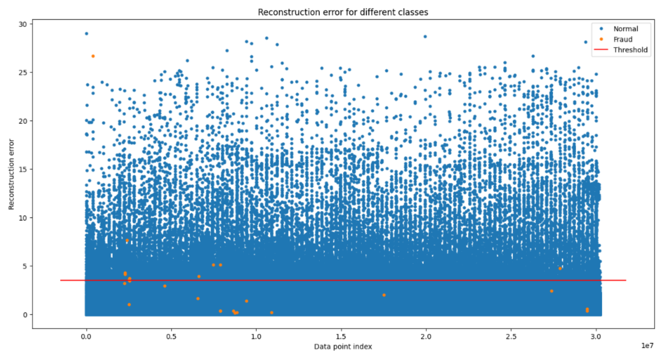


Fig. 11. Visualization of reconstruction error of the test set from the model prediction to predict the malicious transactions.

As one can see from Figure 12, the confusion matrix, the autoencoder algorithm has performed robustly. The model built has been able to detect 38% (10 out of 16) of the malicious data points in test data and 31% (617,804 out of 1,358,965) of them in the training data. Furthermore, the false-positive rate stands at the high end compared to the rest of the experiments conducted in this thesis study. The model has classified approximately 69% of the cases in the training data and 62% (10 out of 16) in the test data as false-positive cases. In contrast, the true positive has been rated at a higher form 99% (Training: 23,986,576 out of 211,849 and Test: 5,996,738 out of of 52,863) used in the test and training data.

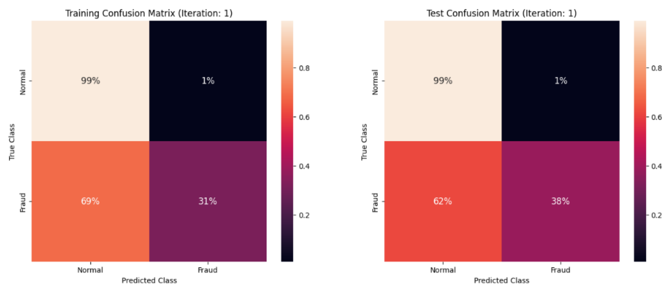


Fig. 12. Confusion matrix for percentage of training and validation of autoencoder model.

from Figure 13 more accurately, for the training data, 23,986,576 of the normal transaction out of 23,988,425 were properly classified, while for the test data, 5,996,738 of the normal transaction out of 5,996,754 were properly identified.

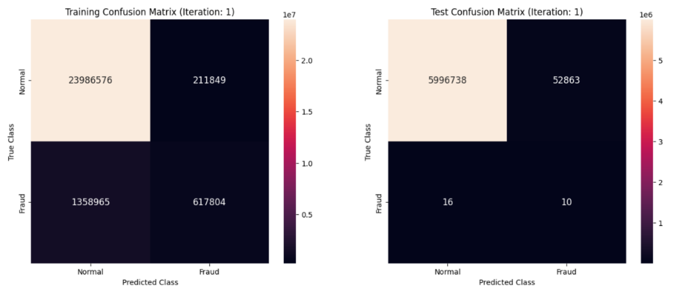


Fig. 13. Confusion matrix for normal of training and validation of autoencoder model.

In summary, autoencoders do a great job detecting normal transactions but come with an enormously high rate of false positives when detecting fraudulent transactions, although at an important scale. Such a trade-off in detecting fraud and efforts to minimize false positives is important for the overall effectiveness of the model or its practical application in the real world.

2) Kmeans: Arthur and Vassilvitskii (2007) argue that K-means clustering performance is not seriously affected by the vast volume of data. Unlike other algorithms, K-means does not sub-sample data into chunks. It handles large data sets, categorizing data points into similar groups. This experiment aims to identify groups containing mostly malicious or non-malicious data points. The entire training set is fed into the K-means algorithm with varying k values to produce k models. Each model is computed with 1000 random initializations, 15000 maximum iterations, and a batch size of 256. Inertia, defined as the sum of squared distances (SSD) of samples to their nearest neighbor, is used to select the best k value, resulting in the best K-means model.

The elbow method determines the appropriate number of clusters by plotting SSD against a range of k values, identifying an 'elbow point' where the rate of decrease sharply slows. This point prevents underfitting or overfitting. Figure 14 shows that five clusters balance inertia and computational efficiency, making the resulting clusters manageable for further analysis.

K-means clustering helps isolate patterns that may indicate fraudulent activities by analyzing centroids and data distribution within clusters. This method is useful for fraud detection, relying on identifying outliers and unusual transactions. Its efficiency with large datasets makes it suitable for real-time applications, such as blockchain networks, where continuous monitoring and quick anomaly detection are crucial. K-means clustering's robustness, scalability, and simplicity make it a powerful tool for anomaly detection in large data environments.

Applying K-means clustering to the entire dataset aims to develop an effective fraud detection system, enhancing blockchain network security and contributing to cybersecurity with a scalable anomaly detection solution. Figure 14, an elbow plot of SSD versus the number of clusters, indicates that k = 5 is optimal. This value ensures the model balances efficient data categorization into meaningful clusters, providing a solid foundation for further analysis and anomaly detection.

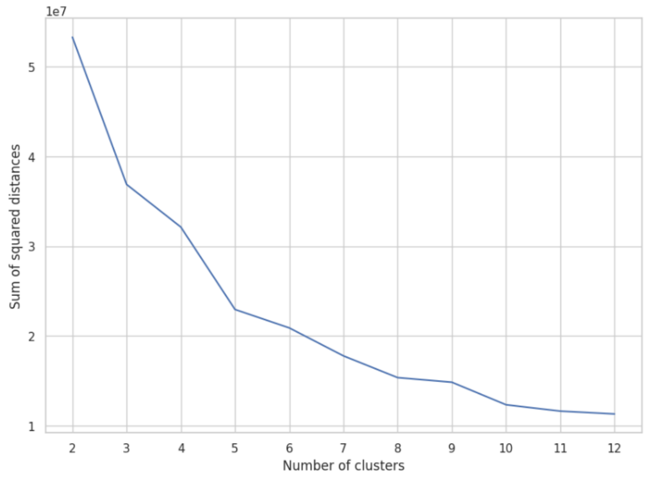
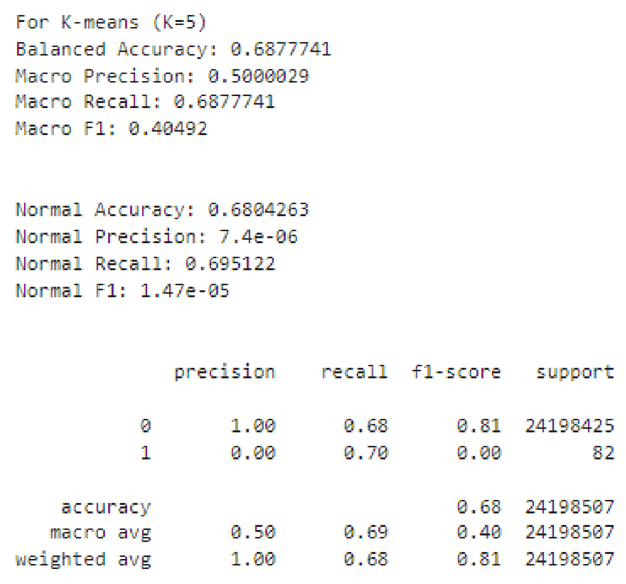


Fig. 14. Plot between k-cluster and sum of squared distances of samples to their closest cluster center.

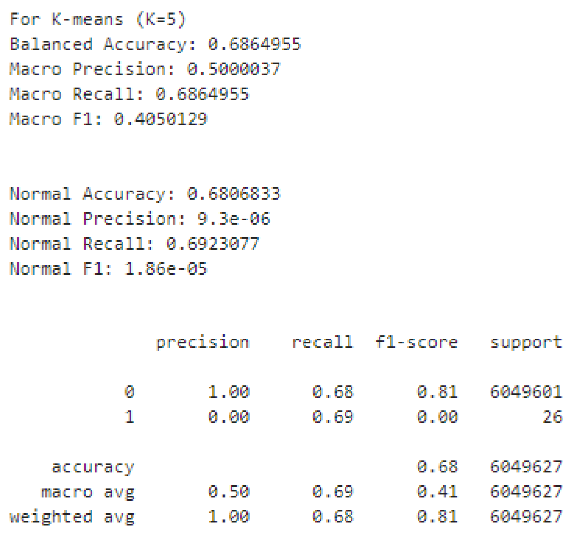
The K-means model (k = 5) shows quite a good balanced accuracy of about 0.69 for both training and test datasets, proving very good performance across both classes. On the other side, the recall for fraudulent transactions is high, about 0.69 to 0.70, while the precision is extremely low, near-approaching zero, for fraud detection with an F1 score. This would mean the model can detect most of the fraudulent transactions but at the cost of very high false-positive rates where many normal transactions are flagged as fraudulent. It observed good precision in normal transactions, but classified fraudulently really gave a hard time to the models, with overall accuracies at around 0.68.

The model with value k = 5 is evaluated with training data and test data. Figure 16, 17 describe the evaluation metrics statistics, and the k-means algorithm performed a decent job in separating the malicious data points from the non-malicious data points. Based on data of table 6.10 and table 6.11,



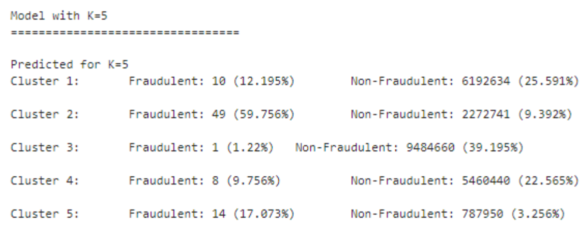
```
For K-means (K=5)
Balanced Accuracy: 0.6877741
Macro Precision: 0.5000029
Macro Recall: 0.6877741
Macro F1: 0.40492


Normal Accuracy: 0.6804263
Normal Precision: 7.4e-06
Normal Recall: 0.695122
Normal F1: 1.47e-05


              precision    recall  f1-score   support

           0       1.00      0.68      0.81  24198425
           1       0.00      0.70      0.00        82

    accuracy                           0.68  24198507
   macro avg       0.50      0.69      0.40  24198507
weighted avg       1.00      0.68      0.81  24198507
```

Fig. 15. Testing data evaluation metrics for the model with K=5



```
For K-means (K=5)
Balanced Accuracy: 0.6864955
Macro Precision: 0.5000037
Macro Recall: 0.6864955
Macro F1: 0.4050129


Normal Accuracy: 0.6806833
Normal Precision: 9.3e-06
Normal Recall: 0.6923077
Normal F1: 1.86e-05


              precision    recall  f1-score   support

           0       1.00      0.68      0.81   6049601
           1       0.00      0.69      0.00        26

    accuracy                           0.68   6049627
   macro avg       0.50      0.69      0.41   6049627
weighted avg       1.00      0.68      0.81   6049627
```

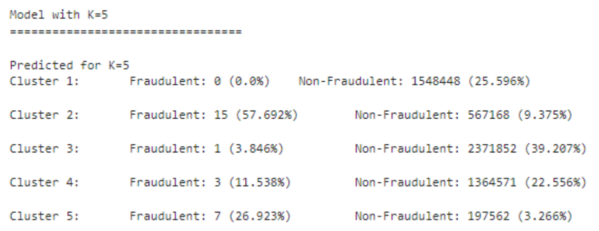
Fig. 16. Training data evaluation metrics for the model with K=5



```
Model with K=5
=================================

Predicted for K=5
Cluster 1:       Fraudulent: 10 (12.195%)        Non-Fraudulent: 6192634 (25.591%)

Cluster 2:       Fraudulent: 49 (59.756%)        Non-Fraudulent: 2272741 (9.392%)

Cluster 3:       Fraudulent: 1 (1.22%)   Non-Fraudulent: 9484660 (39.195%)

Cluster 4:       Fraudulent: 8 (9.756%)          Non-Fraudulent: 5460440 (22.565%)

Cluster 5:       Fraudulent: 14 (17.073%)        Non-Fraudulent: 787950 (3.256%)
```

Fig. 17. Descriptive analysis of training data clustering for model with K=5.

it can be deduced that the model categorized the majority of malicious data-points in cluster 2 and cluster 4. On the contrary, the non-malicious data points are majorly designated to cluster 1, cluster 3, and cluster 5.



```
Model with K=5
=================================

Predicted for K=5
Cluster 1:       Fraudulent: 0 (0.0%)    Non-Fraudulent: 1548448 (25.596%)

Cluster 2:       Fraudulent: 15 (57.692%)        Non-Fraudulent: 567168 (9.375%)

Cluster 3:       Fraudulent: 1 (3.846%)          Non-Fraudulent: 2371852 (39.207%)

Cluster 4:       Fraudulent: 3 (11.538%)         Non-Fraudulent: 1364571 (22.556%)

Cluster 5:       Fraudulent: 7 (26.923%)         Non-Fraudulent: 197562 (3.266%)
```

Fig. 18. Descriptive analysis of testing data clustering for model with K=5.

As observed from the tables above, more than 55% of the anomalous data were clustered into one cluster along with a considerable amount of the remaining malicious data in another cluster. Nevertheless, the majority of non-malicious data was distributed in three clusters, with approximately 25%, 39%, and 22% of data in clusters, respectively, which sums up nearly 86% of the data. A visual illustration of the clusters is created by performing principal component analysis (PCA) as a dimensionality reduction technique on the data. The data is transformed into three dimensions and then plotted into 2-dimensional and 3- dimensional space. The plot in figure 19 represents a 2-dimensional plot of test data with the first principal component as its x-axis and the second principal component as its y-axis. A different color represents all clusters in the plot with a black square representing the center of that cluster along with the number of that particular cluster. Red data points in the visual illustration depict the malicious data points, and as seen in the plot, the majority of them lay in cluster 2 and cluster 4.

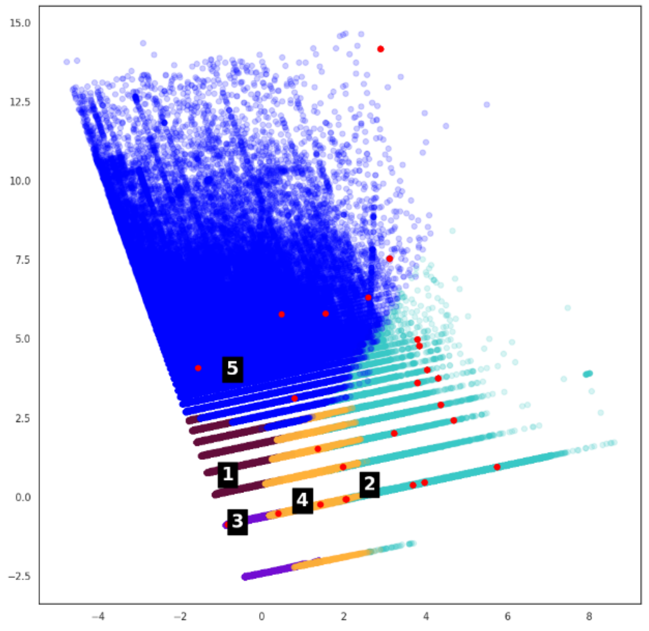


Fig. 19. 2-dimensional plot of predicted test data representing all clusters data and their cluster centers.

Figure 20 and Figure 21 are 3-dimensional space illustrations that give a clear understanding of how malicious and non-malicious data are clustered and distinguished from each other. In the 3-dimensional space, each axis is represented by a principal component extracted from the actual data.

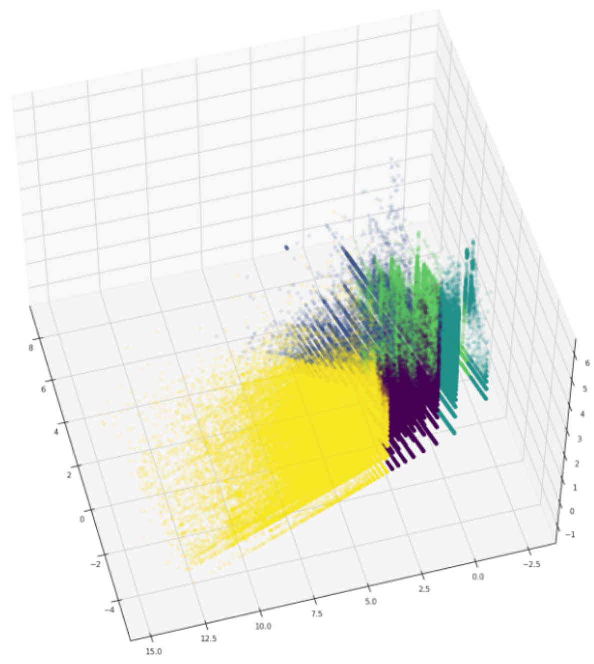


Fig. 20. 3-dimensional plot of predicted test data representing all clusters data and their cluster centers (View 1).

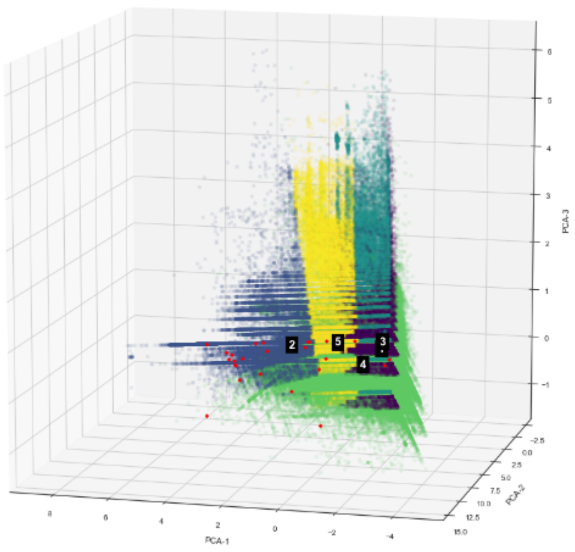


Fig. 21. 3-dimensional plot of predicted test data depicting the clusters along with their cluster centers (View 2)

In order to create a conventional prediction confusion matrix, the clusters with the majority of malicious data points (Cluster 2 and Cluster 4) were merged into one. Conversely, clusters with the majority of non-malicious data points (Cluster 1, Cluster 3, and Cluster 5) were also merged into one. These merged clusters were labeled as malicious and non-malicious data points to compute a final confusion matrix. Based on the results, it can be observed that the k-means clustering model performed decently. For the training data, model was able to detect 70% of fraudulent cases and 68% of non-fraudulent cases. Specifically, it identified 57 out of 82 fraudulent cases and 16,465,244 out of 24,198,425 non-fraudulent cases. For the test data, the model detected 69% of fraudulent cases and 68% of non-fraudulent cases. It correctly identified 18 out of 26 fraudulent cases and 4,117,862 out of 6,049,601 non-fraudulent cases. The false-positive rate was nearly 32%.

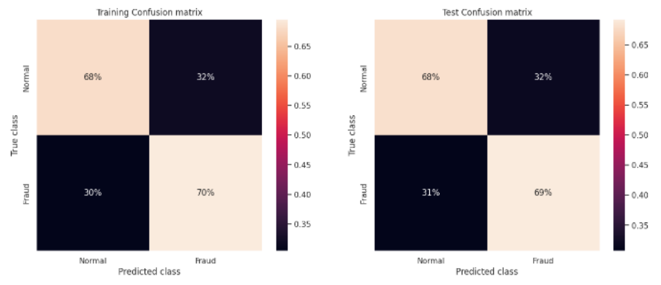


Fig. 22. Training and Test percentage confusion matrix.

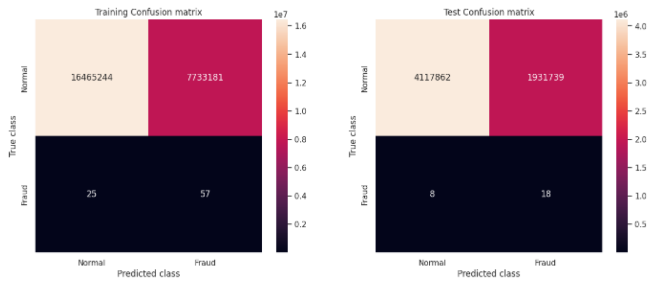
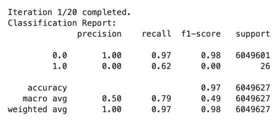


Fig. 23. Training and Test normal confusion matrix.

The K-means clustering model with k=5 shows balanced accuracy of about 0.69 for both training and test datasets, suggesting consistent performance across different data splits. This model effectively separates malicious and non-malicious data points, clustering most anomalous data into specific groups (clusters 2 and 4), while non-malicious data points are mainly found in clusters 1, 3, and 5. Although the model achieves a high recall rate for fraudulent transactions (around 70%), its precision is very low, resulting in a high false-positive rate. This means that while the model can identify most fraudulent activities, it also wrongly classifies many normal transactions as fraudulent.

Overall, the K-means model is a strong tool for clustering and anomaly detection in large datasets, making it valuable for enhancing security in blockchain networks by efficiently categorizing and analyzing transaction patterns.

3) Isolation Forest: The Isolation Forest is a generic and effective one-class anomaly detection algorithm that is efficient, especially in datasets for which anomalies happen infrequently and are very different from the majority. This involves an anomaly that is isolated, rather than a profile of the normal data points.



```
Iteration 1/20 completed.
Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      0.97      0.98   6049601
         1.0       0.00      0.62      0.00        26

    accuracy                           0.97   6049627
   macro avg       0.50      0.79      0.49   6049627
weighted avg       1.00      0.97      0.98   6049627
```

Fig. 24. Classification report for isolation forest

It is an intuitive idea in which an anomaly in the dataset is easier to separate from the rest of the observations. It may split parts of the data randomly by choosing any feature, then choosing a split value from the maximum and minimum values of the chosen feature, and this can be done recursively. This splitting can be represented in the form of a tree structure. It is also known as an isolation tree, where anomalies will be lying closer towards the root of the tree, meaning their paths are of a shorter length, while normal points are isolated at deeper levels of the tree.

For a project that inherently detects anomalies in things like either transaction data or even just a sensor data stream, Isolation Forest can be highly effective, given that this methodology is highly efficient and scalable. Also, it was superior to most advanced, complex, or other solution methods, which are sensitive to size and computationally expensive. Also, this reduces the required resources, which makes it a lot faster and less computationally costly compared to more complex density-based algorithms or clustering-based methods that are highly sensitive to size. Also, Isolation Forest does really well with data in high dimensions, for currently, datasets present a lot of attributes for any given single data point.

The proposed model of Isolation Forests has passed strict evaluation on different multiple iterations in the project, tuning the efficiency in the detection of anomalies in Bitcoin transaction data. The model has been fine-tuned and tested in several repetitions to bring about the best output. Some of the key results are summarized below. The average accuracy of the model is about 97%. This is quite efficient classification for right transactions. Precision for Anomalies: The anomaly detection precision is stuck at about 0.00. This would bring a good challenge to the precision part point in the sense that the anomaly's occurrences might be very few, and it will also be a class imbalance. On the other hand, the recall for anomalies was much higher at around 0.62, meaning the model was able to identify a large share of actual anomalies in the data but was not able to control false positives. F1-Score for Anomalies: The line for the F1-score remained low and very close to 0, indicating that there was a great difficulty in the application. Finding a proper balance between precission and recall was not easy.

These metrics suggest the challenges and simultaneously the effectiveness of the Isolation Forest in detecting anomalies in very skewed datasets, such as fraud detection, where the anomalies are very scarce but important to detect. The tuning is on the model's contamination and on other parameters so that increased prediction accuracy can be achieved and, at the same time, the amount of false positives can be reduced.

On the confusion matrix for the 20th iteration of our model, it appears to do a pretty good job of picking up

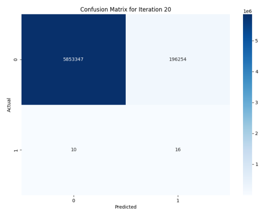


Fig. 25. Confusion matrix for isolation forest 20th iteration

what it deems as normal behavior, as evidenced by the high number of true negatives, at 5,853,347. This shows that our model is very dependable in recognizing basic transactions or activities—very comforting. One blot is that there is a humongous number of false positives, at 196,254. This would mean that our model mistakenly flags basic transactions as fraudulent ones; in an actual application, this would raise undue alarms and could potentially affect the satisfaction of customers.

The worse situation is in the actual fraud detection performance of our model. The model shows 16 true positives against 10 false negatives, indicating that the model's performance in the identification of fraudulent activities is poor. More specifically, a low detection of true anomalies is quite heavy under this context because the rare but critical events are important to catch in fraud detection systems or network security. Then it seems heavily biased towards prediction of the majority class, almost missing out on the minority class of fraudulent cases.

It's this marking of an imbalance that tells us we need to return to the basics of the model. Maybe changing the threshold settings, selecting better features, or using data-balancing strategies could assist in improving the model's sensitivity to anomalies. It is now important that our model correctly detects the dismissal of non-fraudulent activities but also becomes more apt at catching the rare cases of fraud, which are very important for the practical efficacy of the system we are developing.

4) CBLOF(Cluster based local outlier factor): A clustering method for identifying outliers or abnormalities in a dataset is the CBLOF model.The unsupervised CBLOF machine learning model clusters comparable data points together before identifying possible outliers from points that fit poorly into any cluster or that are part of small clusters. Using clustering approaches, the Cluster-Based Local Outlier Factor (CBLOF) model is an anomaly detection tool that finds outliers in datasets. For CBLOF to function, the data must first be clustered according to similarities into several categories. Next, based on the size of the cluster and the distance of each data point from its center, an outlier score is computed for each data point. Points that are distant from the cluster centroids or that are part of minor clusters are considered outliers.

The training process of the Cluster-Based Local Outlier Factor (CBLOF) model incorporates a thorough and systematic approach, conducted over 20 iterations, each contributing significantly to refining the model's accuracy and ability to generalize. Initially, the model addresses data sampling and

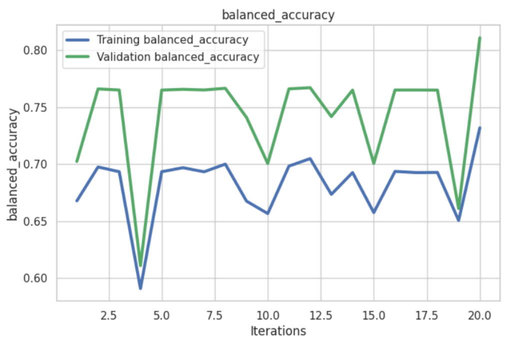


Fig. 26. Training and Test accuracy

balancing in each iteration, using techniques such as SMOTE to correct for class imbalances prevalent in applications like fraud detection, where fraudulent transactions are typically less frequent than normal ones. This step is critical to prevent the model from biasing toward the majority class.

Throughout these 20 iterations, the CBLOF model undergoes rigorous training on these balanced datasets. The model is fine-tuned to effectively identify and differentiate between clusters of normal and anomalous data points. This involves iterative adjustments to critical parameters including a number of clusters, contamination, `alpha`, `beta`, and random state to minimize classification errors and enhance its detection capabilities.

This iterative process carried out over 20 iterations, aims to continuously improve the model based on feedback from the validation results, allowing for fine-tuning of the model's parameters and adapting its learning to effectively handle the complexities of the dataset. Such a methodical approach ensures that the model not only learns robustly from the training data but also performs reliably in practical scenarios, particularly in dynamic environments such as fraud detection.

Following each training phase, the model is validated using a separate dataset not previously seen during the training iterations. This validation assesses the model's performance on various metrics such as precision, recall, F1 score, and ROC AUC, training time, macro F-1, macro precision, macro recall and evaluates its ability to generalize to new, unseen data.

The performance metrics and confusion matrices from the 20th iteration of model evaluation highlight significant insights. It achieves an accuracy of 85 percent at the 20th iteration for test data and around 75 percent for training data. The confusion matrix for the training data shows a significant number of false positives and false negatives. However, it correctly identifies a larger number of normal transactions and fraudulent transactions on testing data with very few fraudulent transactions correctly identified (20), alongside a significant number of false positives (89,027).

The results from this phase may prompt further adjustments to address potential issues like overfitting, where the model performs exceptionally on training data but less so on validation data, or underfitting, which would require enhancing the model's capacity to learn more complex patterns in the data. Hyperopt is used for hyperparameter tuning, aiming

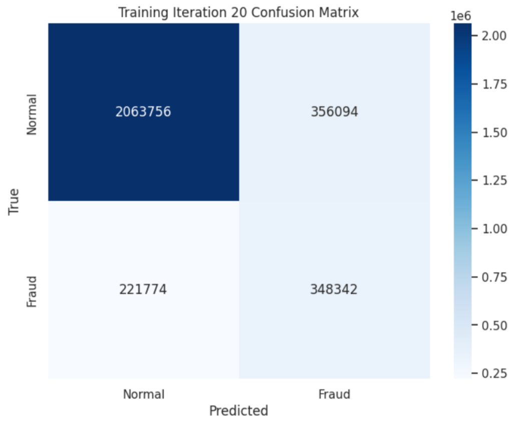


Fig. 27. Confusion matrix of train data

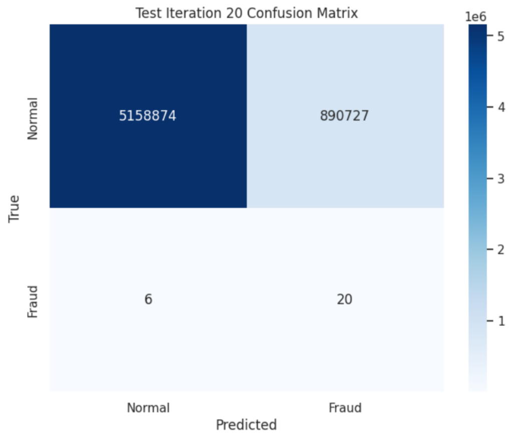
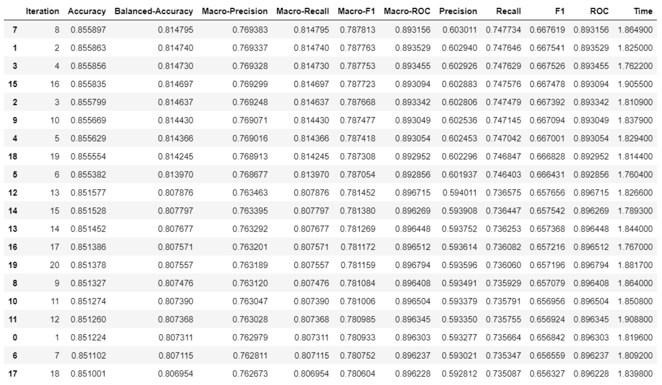


Fig. 28. Confusion matrix of test data

to optimize the model by finding the best parameters that minimize the loss function (maximize F1 score) and focusing on parameters like the number of clusters and contamination rate.

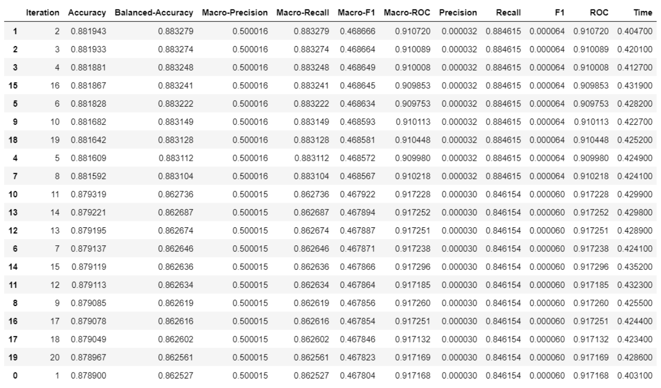
5) PCA (Principal Component Analysis): In general terms, PCA is a dimensionality reduction algorithm. In most cases, the use of this can be manifested in variance-covariance for creating new variables in the dataset represented by functions of original variables, which are generally called principal components. These are unique linear combinations of p numbers of random variables, such as X1, X2, ., Xp. Principal components are unique to the extent that they are mutually uncorrelated, the variance of components is successively demoted from the first component with the highest variance to the last one with the lowest variance, and the total number of variations in the original features, X1, X2, ., Xp, must always be equal to the total combined variation of all the principal components. The technical procedure involved in calculating the principal components is very trivial, and that can be achieved by computing the eigenvectors of a correlation or covariance matrix of data features.

These images display performance metrics for PCA across multiple iterations. These tables consist of the metrics Accuracy, Balanced-Accuracy, Macro-Precision, Macro-Recall, Macro-F1, Macro-ROC, Precision, Recall, F1, ROC, and Time,



	Iteration	Accuracy	Balanced-Accuracy	Macro-Precision	Macro-Recall	Macro-F1	Macro-ROC	Precision	Recall	F1	ROC	Time
7	8	0.855897	0.814795	0.769383	0.814795	0.787813	0.893156	0.603011	0.747734	0.667619	0.893156	1.864900
1	2	0.855863	0.814740	0.769337	0.814740	0.787763	0.893529	0.602940	0.747646	0.667541	0.893529	1.825000
3	4	0.855856	0.814730	0.769328	0.814730	0.787753	0.893455	0.602926	0.747629	0.667526	0.893455	1.762200
15	16	0.855835	0.814697	0.769299	0.814697	0.787723	0.893094	0.602883	0.747576	0.667478	0.893094	1.905500
2	3	0.855799	0.814637	0.769248	0.814637	0.787668	0.893342	0.602806	0.747479	0.667392	0.893342	1.810900
9	10	0.855669	0.814430	0.769071	0.814430	0.787477	0.893049	0.602536	0.747145	0.667094	0.893049	1.837900
4	5	0.855629	0.814366	0.769016	0.814366	0.787418	0.893054	0.602453	0.747042	0.667001	0.893054	1.829400
18	19	0.855554	0.814245	0.768913	0.814245	0.787308	0.892952	0.602296	0.746847	0.666828	0.892952	1.814400
5	6	0.855382	0.813970	0.768677	0.813970	0.787054	0.892856	0.601937	0.746403	0.666431	0.892856	1.760400
12	13	0.851577	0.807876	0.763463	0.807876	0.781452	0.896715	0.594011	0.736575	0.657656	0.896715	1.826600
14	15	0.851528	0.807797	0.763395	0.807797	0.781380	0.896269	0.593908	0.736447	0.657542	0.896269	1.789300
13	14	0.851452	0.807677	0.763292	0.807677	0.781269	0.896448	0.593752	0.736253	0.657368	0.896448	1.844000
16	17	0.851386	0.807571	0.763201	0.807571	0.781172	0.896512	0.593614	0.736082	0.657216	0.896512	1.767000
19	20	0.851378	0.807557	0.763189	0.807557	0.781159	0.896794	0.593596	0.736060	0.657196	0.896794	1.881700
8	9	0.851327	0.807476	0.763120	0.807476	0.781084	0.896408	0.593491	0.735929	0.657079	0.896408	1.864000
10	11	0.851274	0.807390	0.763047	0.807390	0.781006	0.896504	0.593379	0.735791	0.656956	0.896504	1.850800
11	12	0.851260	0.807368	0.763028	0.807368	0.780985	0.896345	0.593350	0.735755	0.656924	0.896345	1.908800
0	1	0.851224	0.807311	0.762979	0.807311	0.780933	0.896303	0.593277	0.735664	0.656842	0.896303	1.819600
6	7	0.851102	0.807115	0.762811	0.807115	0.780752	0.896237	0.593021	0.735347	0.656559	0.896237	1.809200
17	18	0.851001	0.806954	0.762673	0.806954	0.780604	0.896228	0.592812	0.735087	0.656327	0.896228	1.839800

Fig. 29. Principal component analysis (PCA) training data evaluation metrics table



	Iteration	Accuracy	Balanced-Accuracy	Macro-Precision	Macro-Recall	Macro-F1	Macro-ROC	Precision	Recall	F1	ROC	Time
1	2	0.881943	0.883279	0.500016	0.883279	0.468666	0.910720	0.000032	0.884615	0.000064	0.910720	0.404700
2	3	0.881933	0.883274	0.500016	0.883274	0.468664	0.910089	0.000032	0.884615	0.000064	0.910089	0.420100
3	4	0.881881	0.883248	0.500016	0.883248	0.468649	0.910008	0.000032	0.884615	0.000064	0.910008	0.412700
15	16	0.881867	0.883241	0.500016	0.883241	0.468645	0.909853	0.000032	0.884615	0.000064	0.909853	0.431900
5	6	0.881828	0.883222	0.500016	0.883222	0.468634	0.909753	0.000032	0.884615	0.000064	0.909753	0.428200
9	10	0.881682	0.883149	0.500016	0.883149	0.468593	0.910113	0.000032	0.884615	0.000064	0.910113	0.422700
18	19	0.881642	0.883128	0.500016	0.883128	0.468581	0.910448	0.000032	0.884615	0.000064	0.910448	0.425200
4	5	0.881609	0.883112	0.500016	0.883112	0.468572	0.909980	0.000032	0.884615	0.000064	0.909980	0.424900
7	8	0.881592	0.883104	0.500016	0.883104	0.468567	0.910218	0.000032	0.884615	0.000064	0.910218	0.424100
10	11	0.879319	0.862736	0.500015	0.862736	0.467922	0.917228	0.000030	0.846154	0.000060	0.917228	0.429900
13	14	0.879221	0.862687	0.500015	0.862687	0.467894	0.917252	0.000030	0.846154	0.000060	0.917252	0.429800
12	13	0.879195	0.862674	0.500015	0.862674	0.467887	0.917251	0.000030	0.846154	0.000060	0.917251	0.428900
6	7	0.879137	0.862646	0.500015	0.862646	0.467871	0.917238	0.000030	0.846154	0.000060	0.917238	0.424100
14	15	0.879119	0.862636	0.500015	0.862636	0.467866	0.917296	0.000030	0.846154	0.000060	0.917296	0.435200
11	12	0.879113	0.862634	0.500015	0.862634	0.467864	0.917185	0.000030	0.846154	0.000060	0.917185	0.432300
8	9	0.879085	0.862619	0.500015	0.862619	0.467856	0.917260	0.000030	0.846154	0.000060	0.917260	0.425500
16	17	0.879078	0.862616	0.500015	0.862616	0.467854	0.917251	0.000030	0.846154	0.000060	0.917251	0.424400
17	18	0.879049	0.862602	0.500015	0.862602	0.467846	0.917132	0.000030	0.846154	0.000060	0.917132	0.423400
19	20	0.878967	0.862561	0.500015	0.862561	0.467823	0.917169	0.000030	0.846154	0.000060	0.917169	0.428600
0	1	0.878900	0.862527	0.500015	0.862527	0.467804	0.917168	0.000030	0.846154	0.000060	0.917168	0.403100

Fig. 30. Principal component analysis (PCA) testing data evaluation metrics table

with the value of each metric achieved at a particular iteration. The highest accuracy in the training data is achieved in iteration 8, corresponding to 0.855897. Furthermore, in the case of each iteration, these metrics for Balanced Accuracy, Macro-Precision, and others will explain the performance of the model. In the case of the testing data, the highest accuracy is in the results of iteration 2, it reaches a value of 0.881943. In spite of the above high accuracy having been achieved, the Precision and Recall value is kept at a very low range, making the model overfitting or not generalizing well.

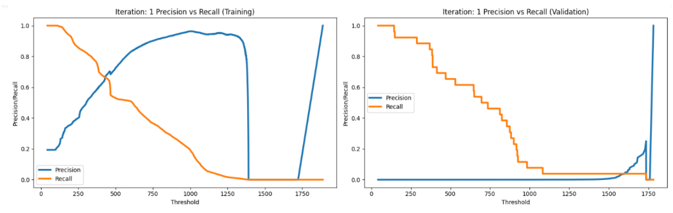


Fig. 31. Precision vs Recall for Iteration 1

Comparing Precision vs. Recall curves for PCA iterations 1 and 20, some relations and differences can be observed on the performance of the model on training and validation sets. In this case, for both iterations, the training data reflects the general inverted trend of precision and recall, with the introduction of a balancing point near a threshold of 500.

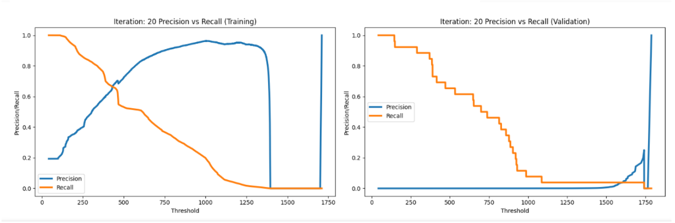


Fig. 32. Precision vs Recall for Iteration 20

However, in iteration 20, this shows steeper changes and more pronounced intersections, meaning that, with prolonged training, overfitting may occur. For both iterations, the validation data is unstable and noisy. In iteration 20, dramatic spikes of high precision are seen at higher thresholds. This type of pattern gives a hint that the model learned how to have very high precision on training data, but the performance on unseen data is not so good, in general, the most probable generalization problem is found.

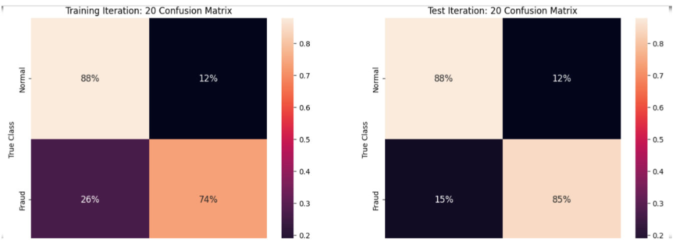


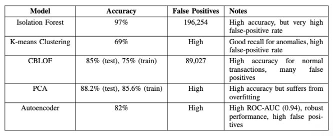
Fig. 33. Training and validation percentage confusion matrix of PCA for itera tion 20

The confusion matrices for the 20th iteration of the PCA model give an insight into the performance of both the training and test datasets. The model identifies 88 percent of normal cases well, evidencing consistency in the identification of non-fraudulent transactions. The fraud case detection seems to fluctuate, though. In the case of training data, the correct identification amounts to 74 percent, while it increases to 85 percent in test data. In this way, it will generalize better on fraud-detection capability when it comes across new data. On the other hand, although in both datasets the model still continues to misclassify 12 percent of normal cases as fraud, the false negative rate is much higher in training data than in test data, at 26 percent compared to 15 percent.

This fact indicates a kind of overfitting in the process of training. In general, the result is rather good for the recognition of normal transactions, with some encouraging improvement in fraud detection in test data, but is yet to be optimized to diminish wrong classification.

C. Results Comparision

In our research on the anomaly detection of Bitcoin transactions, we have compared the unsupervised machine learning models: Isolation Forest, CBLOF, PCA, K-means clustering, and Autoencoder methods. For the Isolation Forest model, the accuracy detected anomalies are 97%, with 0.00 precision and 0.62 recall, but with a high false-positive rate of 196,254.



Model	Accuracy	False Positives	Notes
Isolation Forest	97%	196,254	High accuracy, but very high false-positive rate
K-means Clustering	69%	High	Good recall for anomalies, high false-positive rate
CBLOF	85% (test), 75% (train)	89,027	High accuracy for normal transactions, many false positives
PCA	88.2% (test), 85.6% (train)	High	High accuracy but suffers from overfitting
Autoencoder	82%	High	High ROC-AUC (0.94), robust performance, high false positives

Fig. 34. Comparison of all Models Accuracy

On the other hand, the K-means clustering model shows a balanced accuracy of 69% in both training and test data, with a recall of about 70% regarding fraudulent transactions, but near-zero precision, implying that it has a very high false-positive rate. The showed balanced accuracy of 69% for training data and 69% for test data in the model CBLOF, while false positives were also high at 89,027. PCA showed an accuracy of 88.2% on test data and 85.6% on training data but a problem of high false-negative rates, which meant that overfitting was going on. The Autoencoder model performed decently, with an accuracy of 82% on test data and an ROC-AUC of 0.94, but at the same time, it showed robust performance in the recall rate, showing high false positives. Each model is telling, but the Autoencoder seems promising due to getting the effective anomaly detection, which probably tells of its appropriateness to augment blockchain security through advanced detection methods.

V. Conclusion and Future Work

In this paper, we have exemplified an anomaly detection system for blockchain security by the evaluation of the performances of several machine learning algorithms. From the obtained results, it is quite clear that simpler models tend to perform badly, whereas on the flip side, the advanced methods like Autoencoder and Isolation Forest gave better accuracy and hence, reliability.

In this notion, we performed feature engineering across the dataset; we selected the relevant features available for the dataset that would best make the model estimate with high accuracy. The performance of this act is crucial in the improvement of the workability of systems for anomaly detection. From the preliminary assessments, the models fairly performed, while several had to be tuned to give optimal results. We even went to the extent of synthetically generating data points for the fact that the size of malicious transactional data is smaller, leaving the chances of pattern recognition high but quality of detection low.

Furthermore, all analyses of the experiments show that the Autoencoder model is generally better than the others, indicating its suitability for anomaly detection in blockchain transactions. Richer datasets along with advanced feature engineering in combination with the capability for distributed computing can be some of the areas for future improvement. Broadly, our results indicate that the unsupervised learning approach, and particularly the Autoencoder model, can open the door of opportunity to improve the security of blockchains through effective anomaly detection.

A. Future Work

The current focus is on the Bitcoin transaction. The ability to extend the system to monitor and analyze cross-blockchain transactions, especially regarding Ethereum and Ripple, will multiply its usability and value. New, more updated methodologies in machine learning and deep learning, such as reinforcement learning, neural networks, etc., could bring an upsurge in the accuracy and effectiveness of anomaly detection. Ways to reduce false positives could be further looked into. The system may be further empowered in such a way that it may predict and detect fraud proactively by considering the behavioral pattern over a period of time prior to actual fraud occurrence.

Acknowledgment

The authors would like to thank Dr. Shih Yu Chang and Sourab Saklecha at San Jose State University for guiding us through the project and helping us in successfully implementing the Blockchain Anamoly Detection Project.

References

[1] Arthur, David and Sergei Vassilvitskii (2007). “k-means++: The advantages of careful seeding”. In: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, pp. 1027–1035.

[2] Barrera, Alex (Apr. 2014). A Guide to Bitcoin (Part 1 and 2): A look under the hood.url: http://tech.eu/features/808/bitcoin-part-one/ (visited on 10/16/2018).

[3] Bayer, Dave, Stuart Haber, and W Scott Stornetta (1993). “Improving the efficiency and reliability of digital time-stamping”. In: Sequences II. Springer, pp. 329–334.

[4] Bentley, Jon Louis (1975). “Multidimensional binary search trees used for associative searching”. In: Communications of the ACM 18.9, pp. 509–517.

[5] Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello (2016). “An analysis of deep neural network models for practical applications”. In: arXiv preprint arXiv:1605.07678.

[6] Blockchain.com (2018). Stone Man Loss Transaction Visualization. [Online; accessed 12- November-2018]. url: https://www.blockchain.com/btc/tree/120749. — (2019). Blockchain.com. https://www.blockchain.com/en/charts/n-transactionstotal?timespan=all. [Online; accessed 7-September-2019].

[7] Bolton, Richard J, David J Hand, et al. (2001). “Unsupervised profiling methods for fraud detection”. In: Credit Scoring and Credit Control VII, pp. 235–255.

[8] Breiman, Leo (2001). “Random forests”. In: Machine learning 45.1, pp. 5–32.

[9] Breunig, Markus M et al. (2000). “LOF: identifying density-based local outliers”. In: ACM sigmod record. Vol. 29. 2. ACM, pp. 93–104.

[10] Brugere, Ivan (2013). Bitcoin Transaction Network Dataset. [Online; accessed 29-November2018]. url: http://compbio.cs.uic.edu/data/bitcoin/.

[11] Carlozo, Lou (2017). “What is blockchain?” In: Journal of Accountancy 224.1, p. 29. Chandola, Varun, Arindam Banerjee, and Vipin Kumar (2009). “Anomaly detection: A survey”. In: ACM computing surveys (CSUR) 41.3, p. 15.

[12] Chawla, Nitesh V et al. (2002). “SMOTE: synthetic minority oversampling technique”. In: Journal of artificial intelligence research 16, pp. 321–357.

[13] Chen, Lei, M Tamer Ozsu, and Vincent Oria (2005). “Robust and fast similarity search ¨ for moving object trajectories”. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. ACM, pp. 491–502.

[14] Delgado-Segura, Sergi et al. (2018). “Analysis of the Bitcoin UTXO set”. In: Proceedings of the 5th Workshop on Bitcoin and Blockchain Research Research (in Assocation with Financial Crypto 18), Lecture Notes in Computer Science.