International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Deep Blockchain Approach for Anomaly Detection in the Bitcoin Network

Swapna Siddamsetti[1], Dr. Muktevi Srivenkatesh[2]

Submitted: 27/08/2023 **Revised**: 18/10/2023 **Accepted**: 29/10/2023

Abstract: Long-term research has been done on anomaly detection. Its uses in the banking industry have made it easier to spot questionable hacker activity. However, it is more difficult to trick financial systems due to innovations in the financial sector like blockchain and artificial intelligence. Despite these technical developments, there have nevertheless been several instances of fraud. To address the anomaly detection issue, a variety of artificial intelligence algorithms have been put forth; while some findings seem to be remarkably encouraging, no clear victor has emerged. The transactional data of "Bitcoin," which is one of the public financial block chains, can be detected with the help of several anomaly detection algorithms. This paper makes a quantum leap toward bridging the gap between artificial intelligence and blockchain. In light of anomaly detection, this article also explains the importance of block chain technology and its use in the financial sector. Additionally, it pulls the bitcoin blockchain's transactional data and uses unsupervised machine learning algorithms to look for fraudulent transactions. Although various artificial intelligence algorithms have been proposed for anomaly detection, none have distinctly outperformed the rest. This paper delves into the intersection of artificial intelligence and blockchain, specifically focusing on the transactional data of Bitcoin, a leading public financial blockchain. We employ a range of unsupervised machine learning algorithms to identify potentially fraudulent transactions. A variety of techniques are assessed and contrasted, including isolation forest, cluster-based local outlier factor (CBLOF), deep autoencoder networks, and ensemble approaches service. This paper underscores the synergy of blockchain technology and anomaly detection in the realm of financial security, establishing a comprehensive perspective on modern approaches to fraud detection.

Keywords: *Anomaly Detection, Blockchain, Deep Learning, Fraud Detection, Unsupervised Learning,*

1. Introduction

As long as network structures have existed, there has been suspicious activity in them. Anomalies are entities or their behaviors that frequently exhibit anomalous behaviour inside the system. The identification of financial fraud, network intrusion, anti-money laundering, virus detection, and other cybersecurity applications all make heavy use of anomaly detection. In these networks, finding these abnormalities and stopping such criminal activity from occurring in the future is typically a shared objective. As technology develops, blockchain plays an increasingly crucial role in protecting these network architectures. According to Carlozo and Lou (2017), fraudulent records can be blocked with the help of the immutability property and the decentralised distributed feature of the block chain. However, users of a blockchain network may attempt to engage in illicit activity and, in some situations, succeed in tricking the system to operate in their favour. Modern, cutting-edge approaches for anomaly detection are created and put into practice with the support of centralized systems. Blockchain development technology necessitates the implementation of anomaly detection processes inside these systems. In this thesis, we take data from the publicly accessible bitcoin blockchain and analyze it. According to Nakamoto and Satoshi (2008), bitcoin is a peer-to-peer blockchain for digital currency that enables users to transfer and receive money anonymously between themselves without the necessity of a middleman. The records from the repository are joined together to form a chain, which represents the block chain technology. From a technological perspective, a blockchain is an append-only ledger with the feature of immutability that operates on the phenomena of peer-to-peer (P2P) networking by Saad et al. (2020) with decentralisation. Each block in the blockchain ledger contains many transactions, which can range in kind from financial transactions to data about transportation.

The field of cyber security has historically made extensive use of data analysis techniques Buczak et al. (2015), and more recently, the spread of cutting-edge machine learning techniques has made it possible to precisely detect cyberattacks and threats, both in real-time and during post-incident analysis (Usman et al, 2019; Mahdavifar et al,

[1] *S.D.M.College of Eng. &Tec[1]Department of Computer Science, GITAM School of Science, GITAM Deemed to be University, Vishakapatnam, and Assistant Professor,*
Department of Computer Science and Engineering, Neil Gogte Institute of Technology, Hyderabad, Telangana, India.
swapnangit2021@gmail.com
[2]*Associate Professor, Department of Computer Science, GITAM School of Science, GITAM Deemed to be University, Vishakapatnam, India.*
srivenkatesh.muktevi@gitam.edu
h, Fort Collins – 8023, USA
ORCID ID : 0000-3343-7165-777X
[2] *KLE.Institute of Technology, Tsukuba – 80309, JAPAN*
ORCID ID : 0000-3343-7165-777X
[3] *Computer Eng., Selcuk University, Konya – 42002, TURKEY*
ORCID ID : 0000-3343-7165-777X
** Corresponding Author Email: swapnangit2021@gmail.com*

2019). To assist intrusion detection and prevention systems, to detect system abuses and security breaches with the help of supervised and unsupervised learning algorithms are efficiently used. The packet-level or application-level data, which is a continuous stream of data, describes the underlying network behaviour and is often identified as an interesting situation. Despite these advantages, block chain technology does not provide complete security and is vulnerable to several problems and attacks (Chen et al, 2019; Hassan et al, 2019). For instance, to rob money from innocent users, several Ponzi methods have been discovered, and numerous fraudulent accounts are continuously generated to engage in money laundering. Similar to this, rogue forks are occasionally developed and used to outperform processing capacity and execute double spending within the network (Zhang et al. 2019). As a result, it is critical to accurately and promptly identify the occurrence of these vulnerabilities for the proper operation of a blockchain network. Anomaly detection for blockchain comes into play to make it possible to find and predict these kinds of attacks through blockchain.

This research aims to identify transactions that are unusual or suspicious in the bitcoin network, where all nodes are unlabelled and there is no way to tell whether a given transaction is fraudulent. Finding abnormalities in the bitcoin transaction network is the main objective. According to the authors' explanation in (Farren et a1. 2016), all kinds of financial transaction blockchain systems fraud detection issues that are relevant to the research are identified. This paper looks at a more general problem with finding anomalies in blockchains. This is because the problem can be used in different blockchain networks, like the blockchain of health services, the blockchain of the public sector, etc.

The rapid expansion of the digital financial world has fundamentally transformed the way we perceive, transact, and store value. Parallel to this evolution, unfortunately, is the escalating complexity of deceptive activities targeting these systems. Historically, the financial industry has seen a constant battle between the mechanisms of fraud detection and the intricacies of financial fraud itself. At the heart of this battle lies anomaly detection—a technique aiming to identify patterns in data that do not conform to expected behavior.

The banking industry, for instance, has relied extensively on anomaly detection for identifying suspicious activities. As these traditional systems grew in sophistication, so did the techniques of those wishing to exploit them. Enter the age of blockchain and artificial intelligence (AI): two technological powerhouses poised to revolutionize numerous sectors, especially finance. Blockchain, with its decentralized and transparent nature, promised unparalleled security, while AI's computational prowess offered the promise of predictive accuracy.

Bitcoin, representing the nexus of these innovations, emerged as a public financial blockchain, heralding a new era of decentralized currencies and offering rich, intricate data on financial transactions. Yet, the promise of security through blockchain technology was soon questioned as various fraudulent activities started surfacing in Bitcoin's ecosystem. The challenge then pivoted from detecting anomalies in traditional financial systems to identifying fraudulent patterns in the vast sea of public blockchain transactions.

While artificial intelligence boasts a spectrum of algorithms designed for anomaly detection, a clear victor in terms of efficacy remains elusive. Is there an AI algorithm that can seamlessly sift through the immense transactional data of Bitcoin and spot the proverbial 'needle in the haystack'? Can the amalgamation of AI and blockchain produce a system robust enough to detect and deter financial malpractices?

In this paper, we journey through this intersection, exploring various unsupervised machine learning algorithms specifically tailored to Bitcoin's transactional data. From the isolation forests that hinge on the partitioning of data to the depths of autoencoder networks that reconstruct information, a diverse set of techniques are scrutinized. The intention is not just to discern their individual merits but to appreciate the holistic picture they paint in the realm of financial security.

This paper endeavors to provide both, an insight into the modern challenges of anomaly detection in public financial blockchains and an evaluation of contemporary AI-driven techniques that address these challenges.

2. Background and Related Work

Anomaly detection methods can identify anomalous participants since their behaviour often differs dramatically from that of other valid participants. When developing anomaly detection models for blockchain networks, several crucial considerations must be made in order to obtain effective results. Having enough data to analyse is one of the key requirements for any anomaly detection algorithm to perform well. This phase of data collection is difficult in a typical anomaly detection context, and owing to different safeguards built into the blockchain data collection process, it becomes even harder in blockchain-based anomaly detection scenarios. Depending on the type of blockchain, there are different ways to collect data. For example, in a public blockchain, all participating nodes can access the data, which makes it easy to detect anomalies. However, in a private or consortium situation, the data is not accessible to the general public, and certain clearances

are needed before any data processing can be done. Additionally, the participating nodes in all of these categories are frequently identified by pseudonyms, making it challenging to locate the specific individuals even after classifying them as anomalies due to a lack of knowledge about them. Figure 1 shows some of the most important data sets that can be used to train models that can predict problems in blockchain networks.

An architecture for group-based blockchain anomaly detection was proposed by Ide and Tsuyoshi (2018). The noisy data generated from the sensors can be handled with the notion of deterministic smart contracts in the management of industrial assets based on predefined conditions. The technical key challenges like validation, establishment of consensus, and privacy of data can be traditionally addressed with the support of machine learning methods by formalising the collaborative anomaly detection goal as multi-task probabilistic dictionary learning. The Blockchain can be seen as a platform for collaborative learning rather than a decentralized system for managing data. It has features like being able to track changes and not being able to be changed, and it can be used to get to "Blockchain 3.0."

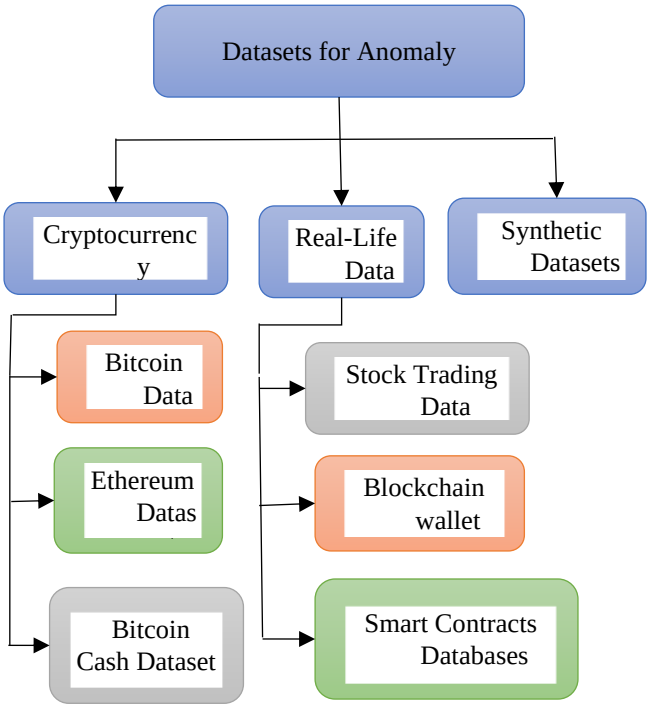


Fig 1 Datasets for detecting anomalies in blockchain

Datasets for Anomaly Detection

1. Cryptocurrency Sets: These datasets are derived directly from the transactional histories of various cryptocurrencies.

Bitcoin Dataset: Contains transaction records of Bitcoin. Typically, this dataset will involve inputs, outputs, amounts, timestamps, and block information.

Ethereum Dataset: Given Ethereum's capability to handle smart contracts, this dataset might also include smart contract interactions, gas prices, and contract executions in addition to simple transfers.

Bitcoin Cash Dataset: A fork of Bitcoin, the dataset will be similar to Bitcoin but will also reflect the specific nuances and rules of the Bitcoin Cash network.

2. Real-life Data Sets: These datasets encompass more traditional financial and transactional records, as well as blockchain applications in real-world scenarios.

Stock Trading Data: Consists of stock prices, volumes, buy/sell orders, and other related financial indicators. Could be used to detect anomalies in trading behavior or price manipulation.

Blockchain Wallet Data: This can encompass wallet balances, transaction histories, and even meta-information like the frequency of transactions, the diversity of received assets, etc.

Smart Contract Datasets: Apart from Ethereum, other blockchains like Binance Smart Chain or Polkadot also allow smart contract functionalities. This dataset will comprise contract calls, contract deployments, and interactions.

3. Synthetic Datasets: Artificially created datasets, either through simulation or by using algorithms. They are often used to test the efficacy of anomaly detection algorithms

under controlled conditions.

3. Proposed Model

This section explains the blockchain anomaly detection process' life cycle as well as the tools and frameworks that were utilised to carry out the tests for this research. Along with data examples from the exploratory analysis, it also discusses how the dataset was created and prepared for the modelling methods. The dataset used for this thesis was created from raw bitcoin blockchain data. Using the bitcoin client program, all data related to bitcoin was synchronised from the internet. The public ledger that houses all the data in the bitcoin blockchain is represented by the bitcoin currency unit (BTC). All transactions involving bitcoin from the time the network was founded to the present are included in the ledger data. Approximately 450,000,000 transactions were available in the bitcoin ledger, according to data from Blockchain.com (2019).

There might be a variety of sender and destination addresses for each transaction. Additionally, since none of the addresses are linked to any personal information, a single client can acquire many addresses, and each client will be unspecified. The year parameter was used to filter away a portion of the synced raw data from the bitcoin client. This subset, which is made up of data about 29,000,000 transactions, was taken out with the help of a Python snippet. BTC Thefts, BTC Hacks, and BTC Losses are just a few of the categories that Bitcoin Forum (2014) has covered in depth. Each category includes information on the red-flagged transaction of interest in each case, as well as the date and the amount of BTC that was stolen or lost.

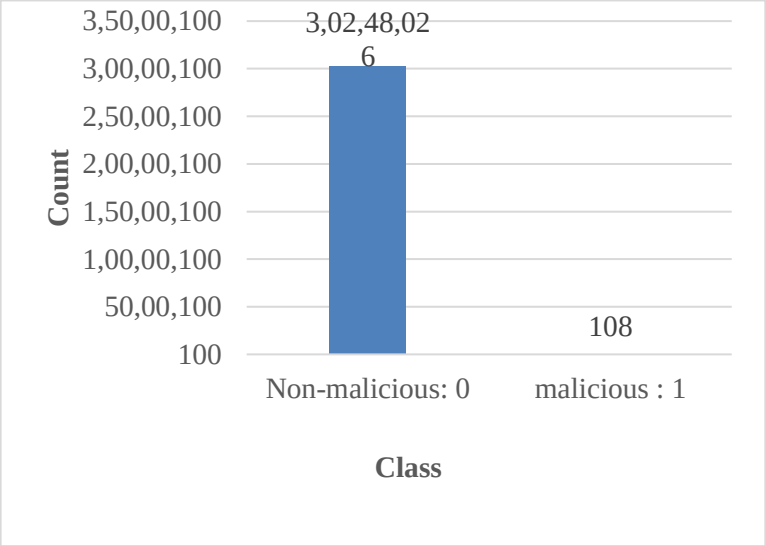


Fig 2 Exploratory Data Analysis

4. Exploratory Data Analysis

Several visualisations are used throughout the exploratory analysis of the data to aid in a deeper understanding of the data and the connections between its aspects. Additionally, it was useful in identifying class imbalances in the data and the association between its features. Understanding the kind of transformation that can be used on data to modify it for modelling was another benefit of the exploratory investigation. Some insights into the data were discovered through exploratory data analysis. The dataset's class distribution is severely unbalanced, as previously indicated, with non-malicious data points of a value of 30,248,026 and just 108 malicious transactions, or around 0.00035 percent of the total data, as seen in Figure 2.

When using the findings from EDA, it's important to identify any patterns or trends related to seasonality in the data. Seasonality refers to regular patterns that repeat over fixed intervals, such as daily, weekly, or monthly cycles. This could be especially relevant in time-series data or data that exhibits temporal patterns. Additionally, highlighting relationships between features can provide insights into potential dependencies within the data. Correlation analysis can help identify which features tend to move together or have some degree of influence on each other. The various problems can be rectified using the following methods:

Outlier Transactions:

Outliers are data points that significantly deviate from the rest of the distribution. Identifying and understanding outlier transactions can provide valuable insights into potential anomalies or errors in the data. Outliers might indicate fraudulent transactions, data entry errors, or other exceptional situations that could impact modelling accuracy.

Imbalanced Dataset and Addressing Imbalance:

An imbalanced dataset occurs when the distribution of classes (or target labels) in the dataset is heavily skewed. For example, in a binary classification problem, if one

class comprises the majority of instances while the other is significantly underrepresented, it's an imbalanced dataset.

Addressing this imbalance is crucial because most machine learning algorithms assume a relatively balanced class distribution. Techniques to handle imbalanced data include:

Oversampling: Generating more instances of the minority class to balance the distribution. This can be done through methods like random duplication or more sophisticated techniques like Synthetic Minority Over-sampling Technique (SMOTE).

Undersampling: Reducing the number of instances in the majority class to balance the distribution. This may involve randomly removing instances or selecting representative instances from the majority class.

SMOTE (Synthetic Minority Over-sampling Technique): This technique involves creating synthetic instances of the minority class by interpolating between existing instances. It helps balance the class distribution while avoiding exact duplication.

Cost-Sensitive Learning: Modifying the learning algorithm to give more weight to the minority class, effectively penalizing misclassifications of the minority class more than the majority class.

Ensemble Methods: Utilizing ensemble methods like Random Forest or Gradient Boosting, which can handle imbalanced data to some extent by design.

When discussing your plans to address the imbalanced dataset, it's important to explain which specific technique you intend to use, and why you believe that choice is appropriate for your dataset and problem domain.

Overall, by thoroughly addressing these points in our analysis, we set the stage for a more robust and effective modelling process.

5. Data Preprocessing

Standardisation is a basic prerequisite of many machine learning algorithms. Usually, this is accomplished by scaling to the unit variance after subtracting the mean. However, outliers might negatively affect the sample mean and variance, which will impact the results. The interquartile range and median may frequently produce superior results in these circumstances, which is why the robust scaler was specifically chosen. The features of the data are shown to be less skewed after the normalisation and sizing procedures have been carried out. The data are more realistic because all of the variables' units are the same. The proposed Anomaly Detection process is shown in Figure 3.

The normalization techniques which can be used as mentioned below:

Normalization techniques in data preprocessing are used to scale and transform the features (variables) in a dataset to a common range. This ensures that the features have similar magnitudes, which can improve the performance of machine learning algorithms and other statistical techniques. There are several normalization techniques available, each with its own advantages and disadvantages. Here are a few commonly used normalization techniques:

Min-Max Scaling (Normalization):

Min-Max scaling scales features to a specified range, usually between 0 and 1.The advantages are

Keeps the relationships between data points intact,Suitable for algorithms that expect input features to be within a specific range.The disadvantages are Sensitive to outliers, as they can disproportionately influence the scaling,does not handle outliers well, which can lead to a compressed range for most data points.

Standardization (Z-Score Scaling):

Standardization transforms features to have a mean of 0 and a standard deviation of 1. The advantages are less sensitive to outliers compared to Min-Max scaling,preserves the shape of the original distribution and is suitable for algorithms that assume Gaussian-distributed data.The disadvantages are does not bound features to a specific range, which might be required for some algorithms.

Robust Scaling:

Robust scaling is a technique that uses median and interquartile range (IQR) to scale features. It's less affected by outliers than Min-Max scaling and standardization.The advantages are resistant to outliers, making it suitable for datasets with extreme values,retains the distribution's shape better than Min-Max scaling.The disadvantages are scales features based on the IQR, which might not be ideal if the distribution of data is not approximately Gaussian.

Normalization by L2 Norm (Unit Vector Scaling):

This technique scales each data point to have a Euclidean norm (L2 norm) of 1. It's often used for text data and in cases where the magnitude of each feature is important. The advantages are useful for algorithms that rely on the magnitude of features,maintains the direction of the data points.The disadvantages are can lead to sparsity in some cases, especially if the data is sparse to begin with.

In our model we used robust scaling as it is suitable for our data collected.

The missing data if any while doing the normalisation can be handled using various mechanisms as mentioned below:

Listwise Deletion (Dropping Rows): This involves removing entire rows with missing values. It's a

straightforward approach, but it can lead to loss of valuable information, especially if the missing data is not random.

Pairwise Deletion (Dropping Columns): In this case, only columns with missing values are dropped. This can preserve more data but may lead to issues if missing values are widespread across multiple columns.

Imputation:

Imputation involves filling in missing values with estimated or predicted values.

Mean/Median Imputation: Replace missing values with the mean or median of the non-missing values in that column. This is suitable for numerical data.

Mode Imputation: Replace missing values with the mode (most frequent value) of the column. This is suitable for categorical data.

Regression Imputation: Use regression models to predict missing values based on other variables. This can be more accurate but requires a significant amount of data and proper feature selection.

K-Nearest Neighbors (KNN) Imputation: Fill missing values using the values of the k-nearest neighbors in the feature space.

Domain-specific Imputation:

In some cases, domain knowledge can guide the imputation process. For instance, if missing salary data is common for unemployed individuals, you could assign a specific value or label to represent unemployment.

Time-Series Interpolation:

In time-series data, missing values can often be interpolated based on the adjacent time points. Techniques like linear interpolation or spline interpolation can be useful here.

Advanced Techniques:

There are more advanced techniques such as multiple imputation, probabilistic models, and deep learning-based imputation methods that can handle complex scenarios. These approaches require a deep understanding of the data and the techniques themselves.

In this paper we proposed deep learning biased imputation for missing data.

The below methods are used for mislabelled instances of data:

Manual Inspection and Correction:

Begin by manually inspecting a subset of your data to identify and correct mislabeled instances. This process can be time-consuming but can help you gain a deeper understanding of the mislabeling issues and improve the quality of your dataset.

Cross-Validation:

Utilize cross-validation techniques like k-fold cross-validation to assess the robustness of your model against mislabeled data. Cross-validation can provide insights into whether your model's performance is consistent across different folds, indicating potential mislabeling issues.

Outlier Detection:

Treat mislabeled instances as outliers and use outlier detection techniques to identify them. If a data point is significantly distant from the cluster of similar points, it could be a mislabeled instance.

Majority Voting:

If multiple labels are assigned to a single instance, consider using majority voting to determine the correct label. This is particularly useful when a small fraction of instances have mislabeled data.

Human Expert Validation:

Consult domain experts or individuals with domain knowledge to validate and correct mislabeled instances. Their expertise can help you make accurate decisions about whether instances are truly mislabeled.

Bootstrapping:

Bootstrapping involves repeatedly resampling your dataset to create new training sets. During each resampling, you can change the labels of a certain percentage of instances and train models on these modified datasets. This can help identify mislabeled instances by observing changes in model performance.

Label Cleaning Models:

Build models specifically designed to identify and correct mislabeled instances. These models can learn from the relationships between features and labels and flag instances with high prediction uncertainty.

Semi-Supervised Learning:

Utilize semi-supervised learning techniques to leverage both labeled and unlabeled data. This can help improve the model's robustness to mislabeled instances in the labeled portion of the data.

Crowdsourcing:

Consider using crowdsourcing platforms to obtain annotations from multiple human annotators for each instance. By comparing the annotators' responses, you can identify and correct instances with inconsistent labels.

Error Analysis:

Conduct a thorough error analysis after model training. Analyze instances that were misclassified by the model and assess whether the model's mistakes are due to mislabeled data. This can guide your efforts in handling mislabeled instances.

We can use any of the method for mislabelled data.

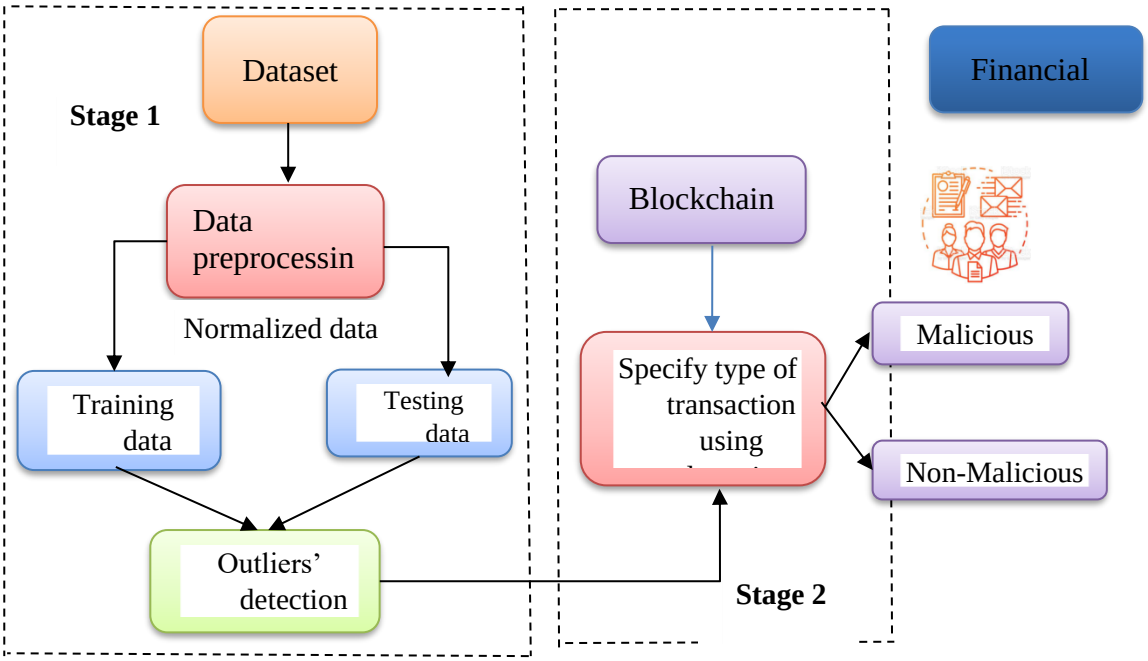


Fig 3 Proposed *Anomaly Detection process*

Different machine learning estimators can perform better by using this preprocessing step. However, the data is still somewhat biased, which might make modelling difficult. Correlation matrices are one of the key components to comprehending data when it comes to feature correlation. They can help us identify factors that strongly sway a particular transaction's likelihood of being harmful. A feature's capacity to influence malice can be shown by either an exceedingly positive or negative association. A second correlation matrix is made and looked at for a random subsample with evenly distributed classes to make sure that too much imbalance in the data doesn't hurt the correlation between features. All the data from this study's experiments is processed through a pipeline that filters, normalizes, and scales the data before it is ready for modelling. Despite the fact that this experiment is considered to be unsupervised, the dataset is classified as both a training dataset and a test dataset for the assessment. The isolation forest training dataset contains 80% of the data; non-malicious data points are 24,198,425, and malicious data points are 82. 20% of the test dataset contains non-malicious data points, or 6,049,601 data points are non-malicious and malicious data points are 26.

Isolation Forest

Isolation Forest is a decision tree-based unsupervised technique for anomaly detection. Data points that are few and abnormally defined are anomalies. The method used by Isolation Forest involves creating a structure of a tree based on attributes that are randomly chosen and processing the sample of the dataset into the tree. The random threshold that is between the minimum and maximum values of the chosen is used to create a branching structure. A sample is less likely to be an oddity as it moves further into the tree.

The following is a description of the algorithm: Let T serve as a tree node, q serves as selected features of a sample, the threshold value is p, and X serves as the dataset with n samples and d features per sample. T may either be an inner node or a leaf node (with two sub-nodes, T_{left} and T_{right}). The sample will remain in the T_{left} if the condition threshold $p > q$ is satisfied, else it will be forwarded to the T_{right}. This procedure is repeated until the node contains just one sample, all of the samples at the node have comparable values, or the tree has grown to its deepest potential level (length). The count of edges that arise from the root node of the tree to the outside node may be used to calculate the length of path h (x). The likelihood that sample x will be classified as an anomaly is increased by the smaller h (x). The sample x's anomaly score may be computed as follows:

$$s(x,n) = 2^{\frac{E(h(x))}{c(n)}} \quad (1)$$

where $c(n)$ denotes the assessment of the average $h(x)$ value for the tree's outside node and can be calculated using the following equation:

$$c(n) = \begin{cases} 2H(n-1) - \dfrac{2(n-1)}{n} & for\ n > 2 \\ 1 & for\ n > 2 \\ 0 & otherwise \end{cases} \quad (2)$$

using the formula $ln(i) + \gamma$ (where is Euler's constant) you can figure out the harmonic value, which is given by the value of $H(i)$.

The CBLOF method of finding anomalies in this technique is based on clustering the data, which is then used to create an anomalous value, like LOF. The clustering stage can be executed by using any arbitrary clustering technique. However, there will be a direct impact on the output quality of the clustering method. This has a direct impact on the quality of the CBLOF results. The software assigns each observation in a dataset named D to a cluster after grouping it using any clustering method. According to their respective sizes, the clusters are arranged in the following order: $|C_1|\ |C_2|\ |C_k|$, where C_1, C_2, ..., C_k represents all the clusters, and k represents the total number of available clusters. The union of all clusters should include all of the observations from dataset D, but any intersection of two clusters should provide an empty set. The next step involves looking for a border index value that distinguishes between small clusters (SC) and big clusters (LC). There are two different approaches to computing this boundary.

$$(|C_1| + |C_2| + \cdots . . + |C_b|) \geq |D|\alpha \quad (3)$$

$$(|C_b|/|C_{b+1}|) \geq \beta \quad (4)$$

A, β are variables that the user defines in the aforementioned equations. The data that the large clusters are made up of and is between [0,1]. (LC). The lower constraint on the relative size among two successive clusters is defined by the value, which must be set to be bigger than one.

Refinements

The provided description is a good starting point for understanding the Isolation Forest and CBLOF methods. However, a few modifications and clarifications can help improve the coherence and readability of the content.

Here are the suggested refinements:

Deep Autoencoders

Autoencoders can be used to successfully complete the task of unsupervised learning as a reduced representation of the latent properties that describe the portion of information. The autoencoder can be defined as a neural network that has been trained to replicate its input as its output. The two parts are shown in Figure 4.

- The encoder Θ, a neural network whose goal is to map an input $\vec{x}$ to a latent compact representation $\Theta(\vec{x}) = \vec{z} \in \mathbb{R}^K$.This mapping results in the process for embedding the original input into a latent vector whose size is K.
- A decoder Φ, is another neural network works with a, given K-dimensional vector $\vec{z}$, aims at producing an output $\Phi(\vec{z}) = \vec{y}$ that is close to the original input.

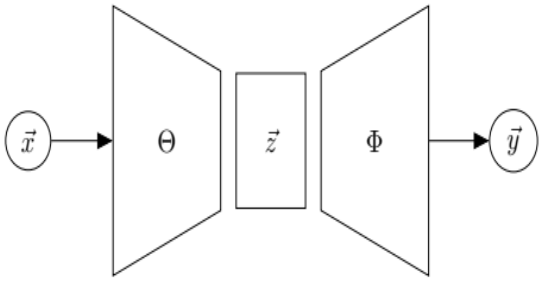


Fig 4 General Autoencoder structure

The features are not included in the summary above. In our approach, the autoencoder's input and output are seen as temporally marked events. Consequently, they might be referred to as recurrent neural networks (RNNs). An RNN supports sequential data as iterating over the sequence saves (partial) RAM for each event. In our strategy, we decided to use RNNs for implementing long-term memory networks (LSTM).

The resultant architecture is represented in Figure 5, designed as a recurrent autoencoder (RAE) model connected with sequence-to-sequence. In essence, the goal of RAE is to: (i) interpret every event in the sequence; (ii) extract the compact representation of each and every event in the sequence; and (iii) use the representation to create a current sequence that is a replica of (or facsimile of) the input provided. This may be defined mathematically as follows: We compute an output given an input sequence.

$O = [\vec{y}_1, \ldots, \vec{y}_n]$ where:

$$\vec{h}_t^{(e)} = LSTM_\theta\left(\vec{x}_t, \vec{h}_{t-1}^{(e)}\right)$$

$$\vec{z} = mlp_v\left(\vec{h}_n^{(e)}\right) \quad (5)$$

$$\vec{h}_t^{(d)} = LSTM_\phi\left(\vec{z}, \vec{h}_{t-1}^{(d)}\right)$$

$$\vec{y}_t = mlp_\varphi\left(\vec{h}_t^{(d)}\right)$$

Where, given the *t*-th event, and representing the encoder, with internal state; symmetrically, representing the decoder, with inner state. Additionally, and stand for multilayer networks that are parameterized by and, respectively. Since the main goal of RAE is to rebuild the input from a small representation, a reconstruction loss can be taken into account when training the model:

$$\ell(I, O) = \frac{1}{n}\sum_{t=1}^{n} \|\vec{x}_t - \vec{y}_t\|^2 \quad (6)$$

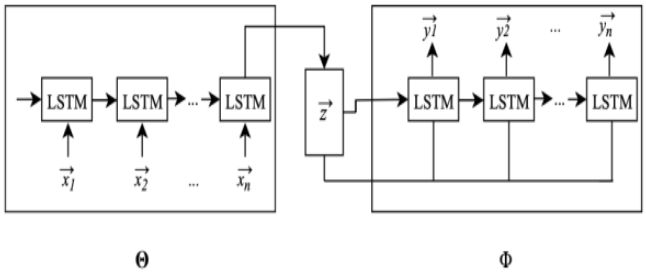


Fig 5 Recurrent Autoencoder (RAE)

Ensemble Classification

Instead of depending just on one algorithm, ensemble approaches use a variety of different algorithms to arrive at a judgement. As was already said, majority voting, which is one way to combine the predictive power of these algorithms, is a democratic way to come to a decision. The majority voting method is shown in equation (7), where the votes of each classifier C_j are used to predict the classification.

$$\hat{y} = mode\{C_1(x), C_2(x), \ldots., C_j(x)\} \quad (7)$$

Here, *A* is a distinct set of class labels, and *A* is the characteristic function. This discusses soft voting, another voting strategy. Soft voting, which is modeled by equation (8), takes into account the decision probabilities *p* of each classification method when making a final choice.

$$\hat{y} = arg\max_i \sum_{j=1}^{m} w_j \aleph_A\big(C_j(x) = i\big) \quad (8)$$

Refinements

Your exposition on the recurrent autoencoder (RAE) utilizing LSTM layers is technical and provides good insight into the model's functionality. However, there are some areas where you might want to refine or enhance the

6. Evaluation Metrics

Several evaluation metrics are used in this work are explained in this subsection. Because there is a large difference between the classes in this current study, traditional ways of giving classes don't work. But when many important evaluation metrics, like confusion matrix, recall, precision, and accuracy, are used together, the evaluation is fair. When dataset classes differ in size, it is sustainable to use micro average metrics, and it is crucial to see the system performance across the classes of data, it is crucial to use macro average metrics. For this study, macro-average measures are usually used to figure out how well the system works overall for both malicious and non-malicious classes.

TP+TN TP/TP+FP+FN+TN may be used to calculate accuracy, a performance metric that computes the ratio between the predicted observations and the total observations. It provides an extensive indication of how effectively a model has been trained, but it performs poorly and might be misleading for datasets that are unbalanced. On the other hand, balanced accuracy is a more realistic method of determining a model's correctness when the data is unbalanced. Equation (9) can be used to show that the balanced accuracy values are equal to the average recall score *R* for each class.

$$balanced\ accuarcy = \frac{1}{C}\sum_{j=1}^{C} R_j \quad (9)$$

Precision may be calculated using TP/TP+FP as the ratio of accurately predicted positive observations to all expected positive observations. Equation (10) shows the macro precision score for *C* classes, where TP_j and FP_j stand for true positive values and false positive values, respectively, for each class label *j*.

$$macro\ precision = \frac{1}{C}\sum_{j=1}^{C} \frac{TP_j}{TP_j + FP_j} \quad (10)$$

The recall score value is defined as the ratio of correctly predicted positive observations to all the observations in the actual positive class values and can be computed by TP/TP + FN.

The macro recall score value for *C* classes is defined as Equation (11) in which TP_j represents true positive values

and FN_j represents false negative values for each class label j.

$$macro\ recall = \frac{1}{C}\sum_{j=1}^{C}\frac{TP_j}{TP_j + FN_j} \quad (11)$$

A weighted average value of precision and recall is defined as an F-Score. It can also be defined as a harmonic mean of precision and recall values, which can be calculated by the equation $2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recal}}$. As equation (12) depicts, the macro-f1 score, which is the harmonic mean between precision value P_j and recall value R_j.

$$macro\ F1 = \frac{1}{C}\sum_{j=1}^{C}\frac{2 \times P_j \times R_j}{P_j + R_j} \quad (12)$$

7. Results and Discussions

This covers the evaluation results and the methodology used to achieve them for all algorithms. The evaluation was conducted using several evaluation indicators. The huge volume of data makes certain algorithms extremely time-intensive. However, to address the issue, we fitted 20 alternative models with precisely calibrated hyper-parameters using a randomly selected subsample of 1/10th of the training data. According to tests, parameters like training exactly 20 models and subsampling 1/10th of the data were determined, and the values were set when the model's performance remained stable regardless of the sample size or number of models used.

Deep Autoencoders: Deep neural networks can process a lot of input, but model calculation can take a very long time. The deep autoencoder network receives preprocessed data that has been divided into training and test sets. Using the Synthetic Minority Over-sampling Technique, 0.8 percent of the malicious data points are added to the training set. This is done to find a pattern of malicious data points that are hidden in the unbalanced datasets (SMOTE).

The autoencoder is trained with 256 batches and 100 epochs. The loss function for this use case is the mean squared logarithmic error (MSLE), which only looks at the proportional difference between the true and predicted values. Tanh and sigmoid functions are employed as the activation functions in the optimization algorithm Adam. The evaluation metrics produced by the deep autoencoder model are shown in Table 1. To determine these measures, training and test data were compared with the calculated model.

Table 1 Evaluation results of training data and test data on a deep autoencoder network.

Evaluation metric	Training Data	Linguistic variable
Balanced Accuracy	0.733119	0.803667
Macro precision	0.716962	0.500029
Macro Recall	0.733119	0.803667
Macro F1	0.724655	0.488155
ROC AUC	0.870147	0.907563
Time	25075.109	177.6881

Table 1 shows that the model had an accuracy rate of 80% when trained on test data. Figure 6 displays the area under the curve function known as the receiver operating characteristic curve (ROC), which for the test dataset is 90%. An examination of the metrics table and a visual representation of the ROC curve give a summary of the model's overall performance. As shown in Figure 6, the receiver operating characteristic curve (ROC) between the training and test sets of data does not significantly differ. However, a better model is often one with a higher AUC score. The recall of the autoencoder model produces an encouraging result since it quantifies the percentage of all relevant outcomes that the algorithm correctly detected. A higher recall measure is essential for the assessment in this use-case of distinguishing between dangerous and non-malicious data bits

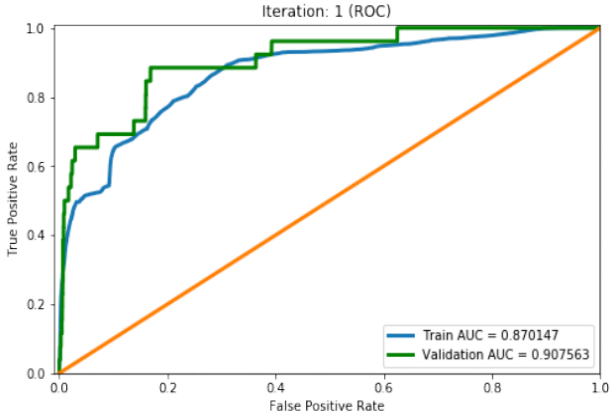


Fig 6 Training and validation of the autoencoder model's area under the curve (AUC) of the receiver operating characteristic curve (ROC)

The trade-off between the model's accuracy and recall at various levels of training and test data is shown in Figure 7. How a threshold should be extracted depends on the goal of the specified use case. The threshold value that is a slight less than 4 appears to be suitable. Figure 7 shows the accuracy vs. recall trade-off for the autoencoder model during training and validation.

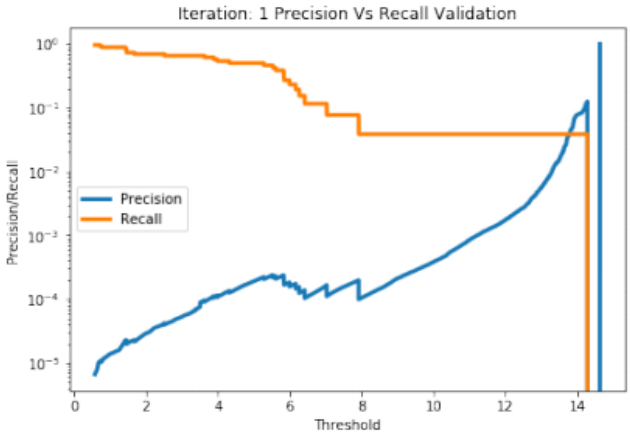


Fig 7 Training and validation precision versus recall trade-off for autoencoder model

Combinatorial Classification

An ensemble classifier receives the test set used in every experiment as input. All the models that were generated for the specified experiments in this paper were combined to create this ensemble classifier. There are 26 malicious data points and non-malicious data points with a total value of 6,049,601 in the preprocessed test data. A voting method is created using all of the pre-computed models using techniques such as Isolation Forest, Cluster-based Local Outlier Factor (CBLOF), and Deep Autoencoder. Each algorithm's classification output has an equal weight, and because this use-case just needs an ensemble approach for prediction, it is fast.

Table 2 displays statistics generated following the ensemble classifier's evaluation. A solid outcome is provided by the ensemble classification approach, which is based on a majority voting algorithm. Figure 8 shows that the area under the curve (AUC) of the receiver operating characteristic curve (ROC) shows that the ensemble method works much more consistently than other methods that have been tested.

Table 2 Evaluation metrics for ensemble classification

Test data	Balanced-Accuracy	Macro-Precision	Macro-Recall	Macro-F1	ROC AUC
	0.80140	0.50002	0.80140	0.4869	0.91576

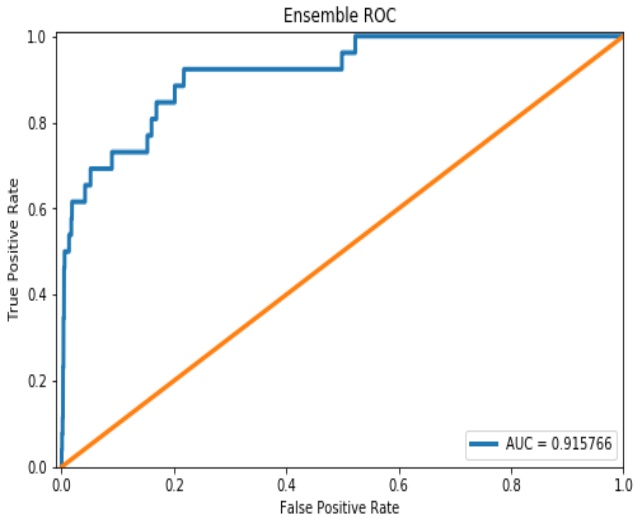


Fig 8 Area under the curve (AUC) of each model's receiver operating characteristic curves is grouped as a whole.

Since the ensemble classification method does not need its own training for the above-mentioned use-case, the evaluation was merely done for the test set. The trade-off between precision values and recall values of the ensemble categorization at various threshold levels is illustrated visually in Figure 9. A good threshold value will be automatically chosen based on an algorithm in the available library that we use to find anomalies.

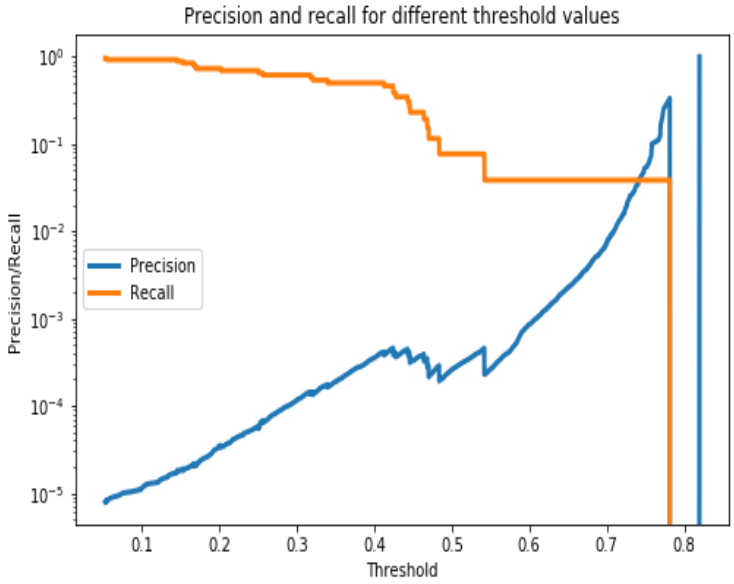


Fig 9 Trade-off between precision and recall for classifying all models as an ensemble.

It is clear from the confusion matrix in Figure 10 that the ensemble classifier delivered reliable and commendable results. Only 35% (9 out of 26) of the harmful data points were misclassified as false positives, while 65% (17 out of 26) were correctly detected. Also, only 5% (308,781 out of 6,049,601) of the non-malicious data points were correctly labeled, giving a 95% accuracy rate (5,740,820 out of 6,049,601).

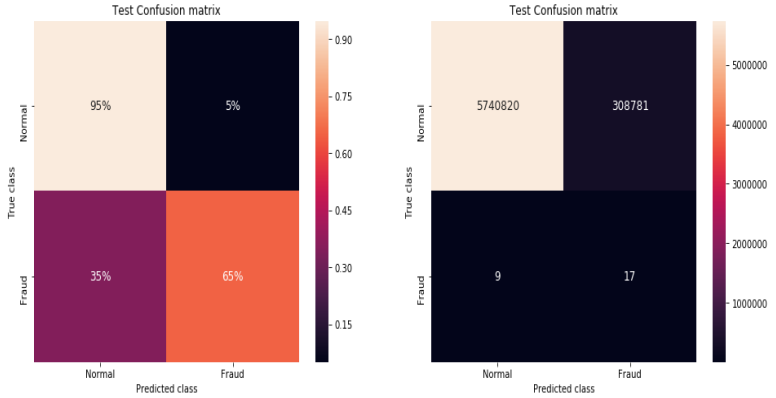


Fig 10 Ensemble classification confusion matrix for all models

However, resilience and ideal performance are not mutually exclusive. In terms of robustness, the ensemble technique appears to be a preferable option since it considers the votes of three classifiers while reaching its decision, and these votes can have different weights that can be changed depending on the use-case for each algorithm. Isolation Forest, CBLOF, and Autoencoder are in fierce competition for the best solo performance, though.

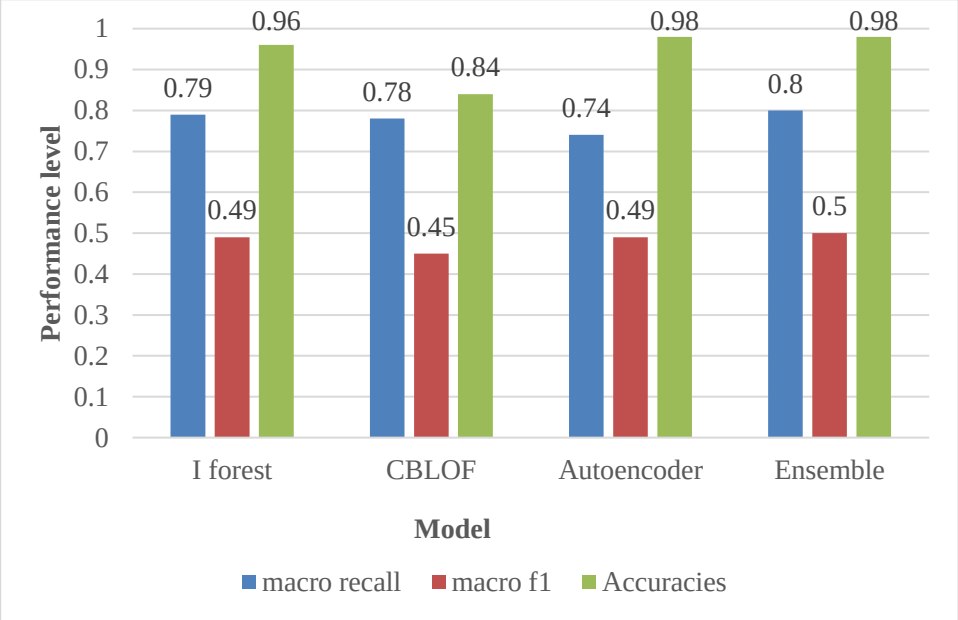
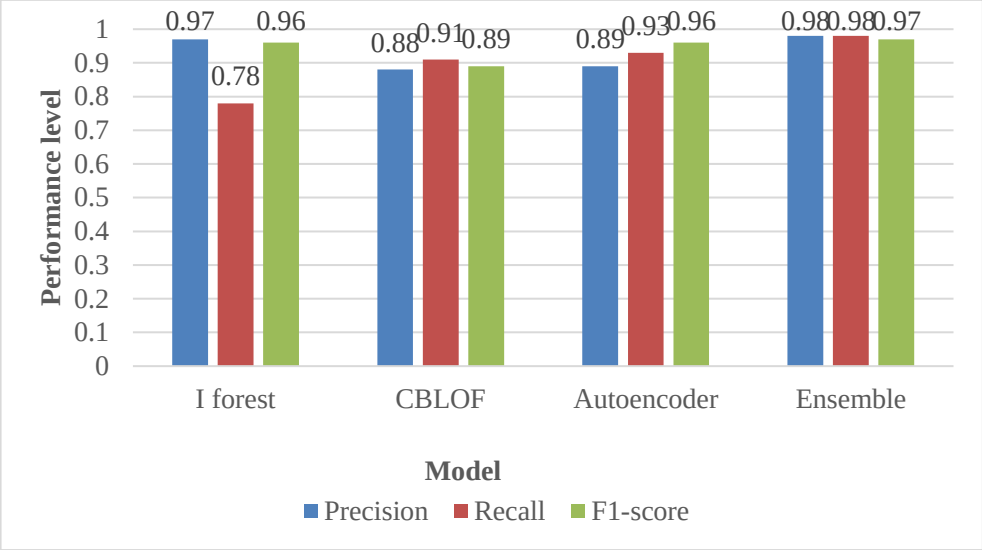


Fig 11 Performance of four models

Figure 11 shows a histogram of the accuracy of each and every model on the test dataset; in this specific use-case, it is not appropriate to choose a model purely based on accuracy. Only relying on accuracy might be deceiving. In situations when the entire performance of the system needs to be taken into account, macro assessment metrics are helpful indications. Figure 11 histogram shows a comparison of the macro recall for each method, and it also shows a comparison of the macro-f1 scores for each algorithm. According to these macro assessment measures' visual representations, the ensemble technique performs more consistently and robustly. An algorithm's robustness describes how well it works on different but related sets of data. Autoencoder and isolation forest, however, perform marginally better.



Fi 12 *Performance of four models*

8. Conclusion

Due to its relevance in conjunction, blockchain technology has drawn a great deal of interest from both academics and industry since its inception. It's feasible that more harmful data might have produced better outcomes. The dataset's modest number of malicious transactional data points means that there aren't many odd trends to look for. This issue was attempted to be solved by artificially producing harmful data points while being careful not to overdo it. However, the quality of anomaly detection is impacted, which makes the models function less well. The choice of characteristics collected from the blockchain transaction graph to identify abnormal activity appears to be a crucial element, as some variables may contain a special opportunity. But the goal of this thesis study is to find strange patterns using unsupervised learning methods. Adding a time-series component would make the job harder.

To address this limitation, we experimented with the generation of artificial malicious data points. Yet, walking this tightrope was challenging: while enhancing the dataset, we were also mindful not to skew it disproportionately. This artificial augmentation, unfortunately, affected the precision of our anomaly detection algorithms, causing the models to perform sub-optimally.

Another pivotal aspect emerged regarding the choice of features extracted from the blockchain transaction graph. Some features seemed to present unique opportunities in detecting anomalies, underscoring the critical role of feature selection in such studies. Notably, while the study aimed to detect anomalies using unsupervised learning methods, introducing a time-series component would invariably compound the complexity. Yet, this might be a necessary evolution for future work in this domain. In conclusion, while blockchain offers a fertile ground for research, this study underscores the intricate challenges of anomaly detection within its vast expanse. Future endeavors in this realm must consider the delicate balance between data augmentation and model accuracy, and the continuous quest for optimal feature selection.

References

[1] Carlozo, Lou (2017). \What is blockchain?" In: Journal of Accountancy 224.1, p. 29.

[2] Nakamoto, Satoshi (2008). \Bitcoin: A peer-to-peer electronic cash system". In: North-Denver-News (2015). Cybercrime- what are the costs to victims. url: http : / / northdenvernews.com/cybercrime-costs-victims (visited on 11/03/2018).

[3] Saad, Muhammad & Spaulding, Jeffrey & Njilla, Laurent & Kamhoua, Charles & Shetty, Sachin & Nyang, Daehun & Mohaisen, David. (2020). Exploring the Attack Surface of Blockchain: A Comprehensive Survey. IEEE Communications Surveys & Tutorials. PP. 1-1. 10.1109/COMST.2020.2975999.

[4] Buczak, Anna & Guven, Erhan. (2015). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications Surveys & Tutorials. 18. 1-1. 10.1109/COMST.2015.2494502.

[5] Usman, Muhammad & Ahmad Jan, Mian & He, Xiangjian & Chen, Jinjun. (2019). A Survey on Representation Learning Efforts in Cybersecurity Domain. ACM Computing Surveys. 52. 1-28. 10.1145/3331174.

[6] Mahdavifar, Samaneh & Ghorbani, Ali. (2019). Application of Deep Learning to Cybersecurity: A Survey. Neurocomputing. 347. 10.1016/j.neucom.2019.02.056.

[7] Chen, Weili & Zheng, Zibin & Ngai, Edith & Zheng, Peilin & Zhou, Yuren. (2019). Exploiting Blockchain Data to Detect Smart Ponzi Schemes on Ethereum. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2905769.

[8] Ul Hassan, Muneeb & Rehmani, Mubashir Husain & Chen, Jinjun. (2019). DEAL: Differentially Private Auction for Blockchain-Based Microgrids Energy Trading. IEEE Transactions on Services Computing. PP. 1-1. 10.1109/TSC.2019.2947471.

[9] Zhang, Shijie & Lee, Jong-Hyouk. (2019). Double-Spending with a Sybil Attack in the Bitcoin Decentralized Network. IEEE Transactions on Industrial Informatics. 15. 5715 - 5722. 10.1109/TII.2019.2921566.

[10] Farren, Derek, Thai Pham, and Marco Alban-Hidalgo (2016). \Low Latency Anomaly Detection and Bayesian Network Prediction of Anomaly Likelihood". In: arXiv preprint arXiv:1611.03898.

[11] Ide, Tsuyoshi. (2018). Collaborative Anomaly Detection on Blockchain from Noisy Sensor Data. 120-127. 10.1109/ICDMW.2018.00024.

[12] Abu Musa, Tahani & Bouras, Abdelaziz. (2021). Anomaly Detection in Blockchain-enabled Supply Chain: An Ontological Approach. 169-169. 10.29117/quarfe.2021.0169.

[13] Golomb, Tomer & Mirsky, Yisroel & Elovici, Yuval. (2018). CIoTA: Collaborative Anomaly Detection via Blockchain. 10.14722/diss.2018.23003.

[14] Guarascio, Massimo & Liguori, Angelica & Manco, Giuseppe & Ritacco, Ettore & Scicchitano, Francesco. (2020). Deep Autoencoder Ensembles for Anomaly Detection on Blockchain. 10.1007/978-3-030-59491-6_43.

[15] Ofori-Boateng, Dorcas & Segovia-Dominguez, Ignacio & Kantarcioglu, Murat & Akcora, Cuneyt & Gel, Yulia. (2021). Topological Anomaly Detection in Dynamic Multilayer Blockchain Networks.