Comp4801 – Final Report

Project Title:

AI-Driven Blockchain Forensics

(**FYP23012**)

Member (UID):

Chow Pak Hang (3035787920)

Supervised by Dr. Chow, Kam Pui

Date of Submission:

2024-04-26

Abstract

Upon the rapid growth of blockchain technology, more criminal activities are presented with the use of cryptocurrency and related services, resulting in significant financial loss. While the forensics tool on blockchain is scarce in detection of malicious accounts and graphical visualization of transaction behaviour. Hence, this project is aims to develop application to investigate behaviour of anomaly accounts by leveraging the machine learning technique. It has explores the library such as Support Vector Machine (SVM), Random Forest (RF), and Light Gradient Boosting Machine (LightGBM), unsupervised learning including K-means, Hierarchical Clustering. In addition, it visualizes transactions of account in graphical approach, which is in form of node-link diagram. In the process of model training, a total of 13941 labelled accounts (5819 anomaly and 8122 normal) are used for the implementation of the Classifier. The sampling methods using Synthetic Minority Oversampling Technique (SMOTE) with TomekLinks have attempted to improve the issues of data imbalances. The best supervised model using LightGBM achieved an average recall, specificity of about 98%, with average false positive rate (FPR) of 1.86% and false negative rate (FNR) of 3.78%. While the unsupervised learning models using OneClassSVM achieved best 97.8% of specificity and FPR of 2.2% under certain conditions. The application is built with Ionic with the integration of both the anomaly account detection and transaction visualizer.

Acknowledgment

Thank you for the support of the project supervisor, Dr. Chow, Kam Pui and the opportunity granted by the Department of Computer Science for doing this project.

Table of Contents

List of Figures

List of Tables

List of Equations

Abbreviations

Abbreviation	Definition
ADASYN	Adaptive Synthetic Sampling
API	Application Programming Interface
AUC	Area Under Curve
CA	Contract account
DBSCAN	Density-based spatial clustering of applications with noise
DeFi	Decentralized Finance
DT	Decision Tree
ENN	Edited Nearest Neighbor
EOA	Externally owned account
Ether	ETH
FNR	False negative rate
FPR	False positive rate
GBDT	Gradient Boosting Decison Tree
GMM	Gaussian Mixture Model
HC	Hierarchical Clustering
HDBSCAN	Hierarchical Density-based spatial clustering of applications with noise
ICO	Initial Coin Offering
IDS	Intrusion Detection System
KNN	K Nearest Neighbors
LR	Logistic Regression
NFT	Non-Fungible Token
LightGBM	Light Gradient Boosting Machine
RF	Random Forest
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristic
SHAP	Shapley Additive Explanations
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
t-SNE	t-Distributed Stochastic Neighbor embedding
TNR	True negative rate
TPR	True positive rate
XAI	Explainable Artificial Intelligence
XGBoost	Extreme Gradient Boosting

1 Introduction

In 2009, the Bitcoin platform launched based on the novel distributed ledger database, “blockchain", which is the transaction platform contains substantial properties such as decentralize, immutable, and transparent [1]. Recently, upon the rapid development of various applications on blockchain, in particularly, the Decentralized Finance (DeFi), Non-Fungible Token (NFT), Initial Coin Offering (ICO), and intermediary platform have shown that blockchain technology have provide a variety of benefits in financial industry and other areas [2]. For instance, cross-border payment allows minimized processing costs and efficient transactions in a more secure environment [2].

However, more cybercrimes are occurring on these blockchain platforms, especially for the public blockchains that is open-source and accessible to everyone, for instance, phishing, scam, fraudulent, fake ICO, and money laundering have cause considerable economic loss for the consumer [3]. It is recognized some of the properties of blockchain can favor the criminal activities. As an illustration, the property of decentralization, indicating no single party can control the platform, have enhanced the integrity on transaction, while it may increase the difficulty for the authorities to prohibit the illegal activities within the blockchain [4]. Moreover, the majority of public blockchains are having pseudo-anonymous identifiers for the entities involved in these blockchains, this makes the investigation towards suspicious and Anomaly accounts challenging [2], [4].

Presently, there are a variety of blockchain explorer websites that can query specific account data from multiple blockchains. In addition, certain intrusion detection systems (IDS) are established from scholars and experts for the recognition of anomalous activities in blockchain. Nonetheless, several blockchain explorers lacking the ability to visualize the transaction history [5] while the proposed IDSs are varying in quality and strategies [6]. As a result, it is motivated to provide a practical application that is publicly accessible for the detection of blockchain accounts and graphical visualization, concurrently, evaluating the existing anomalous classifier models for improvement of IDS in the future.

In the project, the Ethereum platform is the primary focus for forensics. Ethereum is widely adopted for the applications of smart contract, which allows diverse usage such as finance, logistics, and art in a simple and secure manner [2]. Likewise, IDS in Ethereum blockchain can be applicable to different area and targeting Ethereum is beneficial to further studies. The Anomaly account in this project is mainly referred as the Ethereum account undertaking several illegal activities such as phishing, scamming, fraud, and money laundering. With usage of Ionic, it have implemented the cross platform (web and mobile) applications, and machine learning model for the detection of anomaly accounts.

1.1 Objectives

This project is mainly consisting of 3 objectives, which are explained in terms of the benefits of the project and rationale of related works in Section 3.

Objective 1: To Detect Anomalous Accounts on Ethereum Blockchain

Objective 2: To Visualize Transaction History in Graphical Approach

Objective 3: To Evaluate Existing Models on Anomalous Detection

1.2 Project Schedule and Status

For the status of the project, it is on schedule and have completed the all the proposed works and stages. It has completed of Stage 1: Anomaly Account Detection, Stage 2: Transaction Visualizer, and Stage 3: The integration of the application. The details of proposed project schedule with status are stated in **Table 1.1**.

Stage	Deliverables	Details	Status
0	**Preliminary Research** Aug – Sep 2023	- Research and doing project plan	Done
-	**Phrase 1 milestones** Aug – Sep	- Complete Project Plan - Setup of Project Webpage	Done
1	**Data Preparation** Oct	- Collect blockchain dataset with normal and malicious activity	Done
1	**Stage 1:** **Anomaly Account Detection** Oct – Dec	- Data pre-processing, Algorithms Research - Model Training and Evaluation of Machine learning algorithms	Done
-	**Phrase 2 milestones** Oct – Jan	Interim Report & First Presentation - Preparation and Finalize - Preliminary Implementation	Done
2	**Stage 2:** **Data Analysis with Graphical Visualization** Dec – Mar 2024	- Organize account data with graphical method - Analyse account data for visualization	Done
3	**Stage 3:** **Front-End Development & Integration** Jan – Mar 2024	- Development of application and Integration of classifier model - Allow user input and graphic representations of transaction	Done
-	**Phrase 3 milestones** Mar – Apr 2024	- Preparation of Final Presentation and finalized Report - Finalized tested Implementation	Done
-	**Project Exhibition** Apr	Project Exhibition: - 3-min Video and Poster	Done

Table 1.1: Proposed project schedule with description and deadlines. Green text represents the completed stage

1.3 Outline

In the remaining part of this report, Section 2 introduces the background of the Ethereum blockchain and analyze the related works in blockchain forensics. Section 3 describes and justifies the methodology for the implementation of the application, with the procedures and technical aspects. Sequentially, Section 4 discusses the results established, difficulties, and limitations, and potential solutions for this project. Moreover, Section 5 describes the project status and future work. Conclusively, Section 6 concludes the main content of the report.

2 Background and Related Works

In this section, the background knowledge of the Ethereum blockchain platform is introduced (Section 2.1), the related works (Section 2.2) on Anomaly account detection and the visualization of the transaction in blockchain are described and compared.

2.1 Accounts and Transactions on Ethereum

In the Ethereum, all operation and the account states are maintained by the Ethereum Virtual Machine (EVM), which also ensure the valid state of the Ethereum environment [7]. There are two types of account in Ethereum, namely the Externally owned account (EOA) and the Contract account (CA) [8]. EOA is responsible for the transfer of tokens and Ether (ETH) (the native cryptocurrency in Ethereum) with another EOA. While CA is the smart contract deployed to the blockchain which is controlled by the self-executing code, only EOAs and other smart contracts can initiate the transaction from the CA [9]. In this report, the transactions between the EOAs are denoted as the *normal* transaction that recorded on the ledger, while the transactions executed from the CAs are denoted as *internal* transaction [10]. In 2022, the Ethereum have switched the consensus algorithms from Proof-of-Work (PoW) to Proof-of-Stake (PoS), which significantly reduce the energy consumption as compared to the computationally intensive process in PoW [11].

On the other hand, all the transactions are associated with the gas and gas price in Ethereum. Gas is defined as the amount of computation required for executing that transaction on the blockchain, gas (with value of 21000) are fixed for certain operation such as transferring Ethers [12]. The transaction fee is calculated from multiplies of unit of gas used and the gas price [12]. While the gas price is consisting of the base and optional priority fee, the amount of priority fee can be adjusted by the account initiated the transaction [12]. A higher priority fee increases the probability for including the transaction into the next block, indicating higher chance for faster transaction [12]. The basic structure of the transaction records in Ethereum is shown in **Table 2.1**, which summarize the important attributes that are analyzed in this project.

Field	Description
From	The field contains the sender's address of the transaction
To	The field contains the receiver's address of the transaction
Value	The field contains the amount of ETH in terms of in terms of wei (1 ETH = 10^{18} weis)
Data	The optional filed that is empty for ETH transfer, contains bytecode of contract at deployment
Gas Used	The field contains the gas used in the transaction
Gas Price	The field contains the gas price in the transaction
Gas Limit	The field contains the gas limit in the transaction, which set the maximum gas to be used for CA execution to avoid infinite loop of execution
Timestamp	The field contains the timestamp for the transaction being executed and included in the block

Table 2.1: The basic structure of an Ethereum transaction records, listing the important fields and their description

2.2 Related Works on Blockchain Forensics

2.2.1 Anomaly Account Detection

In [13], Farrugia et al. utilized Extreme Gradient Boosting (XGBoost) to classify abnormal accounts from datasets of 4681 accounts that consisting of 2179 anomaly accounts and 2502 ordinary accounts. It has achieved an average accuracy of 96.3%. However, the approach is believed to be rather simple and may need more experiment in detecting the malicious accounts in the large network of Ethereum [13]. Other studies on the detection of the anomaly (or malicious) account have utilized the supervised and unsupervised learning respectively [10]. It has used the supervised algorithms such as

Random Forest (RF), K-nearest neighbor (KNN), and Extreme Gradient Boosting (XGBoost) methods, while using the unsupervised methods such as K-Means, Density-based spatial clustering of applications with noise (DBSCAN), and Hierarchical Density-based spatial clustering of applications with noise (HDBSCAN) for the analysis [10]. It has performed feature extraction from the temporal properties of the account.

Furthermore, there are several research on specific types of the anomaly accounts. [14] proposed the recognition based on Network Embedding algorithm (utilized the proposed *trans2vec* algorithm) which detects the phishing accounts on Ethereum. The dataset has 1,259 addresses labeled as phishing out of 500 million address and another 1,259 normal addresses. The graph used in the model is involved of the second order transaction network, more than 60,000 nodes and 200,000 links on average in each subnetwork out of 50 random generated subnetwork [14]. It is believed the large amount of transaction data from accounts is complex and may not be efficient in distinguishing anomaly accounts in real-time application without a reliable and efficient computation resources. Another work has analyzed the detection of specific types of anomaly activities, the money laundering for the transactions associated with the accounts [15]. It has proposed the GTN2vec graph embeddings algorithms with an average accuracy of 95.7%, it is suggested to outweigh other related graph embedding methods [15].

On the other hand, there are several research on the account's diversity of Ethereum blockchain [16] [17]. It has utilised the clustering techniques to divide Ethereum account into certain parts. For example, it can be used to detecting the abnormal behaviours present in certain accounts. The two types of methods: heuristics methods and machine learning based approaches. The first methods may rely on the previously recognised pattern such as reused address for deposit, participation in certain Ethereum-based layers. However, it is undeniable that lacking the authoritative labels for the dataset may result in poor performance in majority of the categories of accounts in Ethereum while difficult to assess the accuracy of the detection [16].

Despite multiple research [10], [13] - [15] are conducted on the malicious account recognition, they are diverse in strategies and assumptions. In contrast, this project have consider several algorithms in detecting the anomaly activates from accounts, moreover, providing the evaluation of the various models and an application that utilized the classifier model, which allow ordinary people and researcher to recognize the abnormal behaviour of accounts.

2.2.2 Graphical Visualization of blockchain accounts

As claimed by [5], most blockchain visualization tools are merely using simple chart and time series methods, where these approaches may not be effective in tracking money flow through transaction. For instance, the renowned blockchain explorer, Etherscan.io have certain features of analytics using the line chart, bar chart, and heatmap to represent the statistics of transactions, and account status in time series [18], however, without the transaction in form of the money-flow graph.

In the use of money flow graph for the transaction history of the blockchain account, it is believed to be beneficial to certain fields including the tracing of the money flow, visualize the money flow among multiple addresses, and recognize the pattern and behaviours of the anomaly accounts [19]. In the sight of lacking dedicated visualization platform for Ethereum blockchain [5], it is motivated to visualize the transaction history of account in this project, which is believed to be valuable for the future research on the anomaly activities in blockchain forensics.

[20] have explored the dynamic patterns of Bitcoin transaction with the large scale of data. The Visualization have used the ForceAtlas2 algorithm available in the SigmaJS library with the top-down approaches. With the usage of stylized representation, for instance, the color and size, of nodes, it is pointed out that the address associations, difference in frequency and amount of transaction can be visualized. However, the UTXO structure of Bitcoin's transaction is difference from the account-based model of Ethereum, while it may provide certain heuristics for some features of the transaction, the structure such as finding the origins of money from in and out spending of transaction cannot apply to the area of Ethereum.

3 Methodology

In this section, the approaches for achieving the objectives in Section 1.1, including the implementation of anomaly account classifier (Section 3.1), graphical visualization of transaction history (Section 3.2), and integration of application and technical implication (Section 3.3) are introduced in general and technical level with justifications.

3.1 Anomaly Account Detection

To differentiate malicious account in Ethereum blockchain, the anomaly account classifier is built with the leveraging of machine learning algorithms. It takes input from user for a specific Ethereum account address, then the address data is being collected and processed in feature extraction and cleansing procedures. Furthermore, the extracted data are analyzed with the use of the classifier, the predicted risks for determining the anomaly activities mapped into 5 level to indicate the probability of the risks. To achieve the anomaly account recognizer for detection of illegal activities, several stages are divided and explained in in data collection, data pre-processing, and model training and comparison. The proposed framework is illustrated in **Figure 3.1**.

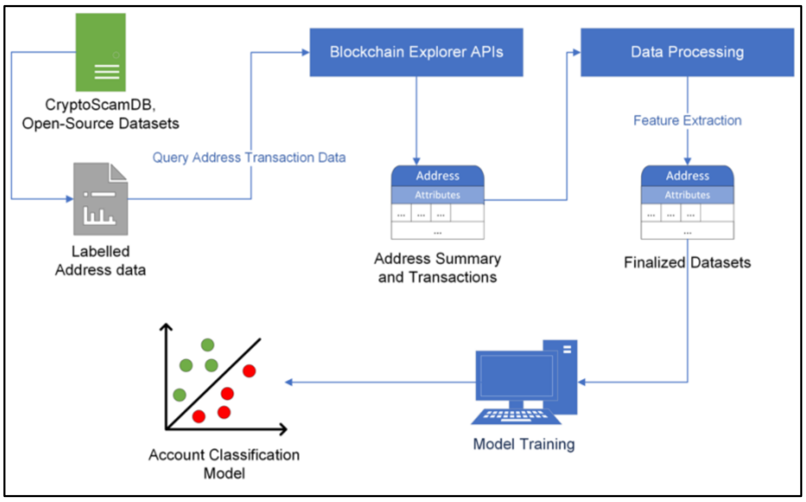


Figure 3.1: Proposed approaches on implementation of Anomaly Account Classifier

3.1.1 Data Collection

To build the classifier, labelled dataset for normal and anomaly accounts is necessary for supervised and semi-supervised learning. For unsupervised learning, it is suitable for large dataset (i.e., blockchain), however, it may require large number of accounts for the analysis and more difficult for evaluating its performance [21]. In this project, the dataset is mainly used for the model training for supervised and unsupervised learning.

Since there is no labelling on the public Ethereum blockchain, the data is collected through several databases and open-source datasets. Labelled anomaly Data are assembled through certain sources, including the academic open-source dataset [13], public cryptocurrency scam database, CryptoScamDB [22], and renowned data science resources website, Kaggle [23]. It is believed that these datasets have a high quality in correctness of the labels since various experts and research are involving in the validation. For technical tools, Python and JavaScript are utilized for web crawling purpose.

3.1.2 Data Pre-processing

On the other hand, all the collected account addresses (anomaly and normal) are validated and pre-processed. For instance, they are input to a blockchain explorer called Etherscan.io [18] for checking the public name tag for “Phish/ Hack” label. For feature extraction, a selection of attributes from the accounts are extracted as the features for model training and prediction. The features are extracted based on academic evidence, heuristics, and statistical values for enhancing the effectiveness of the classifier model since the choice of features is vital for establishing model with precise detection of anomalies [6].

The dimension reduction techniques are used, including the Principal Component Analysis (PCA) and the T-Distributed Stochastic Neighbor Embedding (T-SNE), which are described as followings:

<u>Principal Component Analysis (PCA)</u>

Principal Component Analysis (PCA) is being the traditional methods for dimension reduction, which could reduce the number of features while having similar or better performance. It is widely used in the area of dimension reduction. In brief, the PCA project the data from high dimensions into a lower dimensions, which attempt to maintain the data variance to avoid loss of the information of the features of the data [24] [25]. In other words, the PCA is type of statistical approaches, mapping the features by rotating the axis of the feature space [24]. Moreover, it can reduce the data noise and improve training efficiency [25]. Nevertheless, it is indicated that the non-linear relations between features or components (i.e., components of features in PCA) may be lost in the initial datasets, while it may influence the model performances.

<u>t-Distributed Stochastic Neighbor Embedding (t-SNE)</u>

The principle of t-Distributed Stochastic Neighbor Embedding (t-SNE) involving of the applying of SNE and the conversion of high-dimension features into the conditional probabilities with the measurement of Euclidean distances [26]. In theoretical, the computation of probability similarity would have a higher performance for both linear and non-linear dataset as compared to the PCA [26].

3.1.3 Model Training and Comparison

Moreover, the finalized data are processed for model training with various machine learning algorithms for comparison of the performance. To have a higher effectiveness for evaluating the models, the collected data is divided into 3 datasets, training data, validating data, and testing data. The data for training and validating consist of 90% of the collected data, which is mainly used for the fine-tuning of the hyperparameter. While testing consist of 10% of the collected data, which is mainly used for stimulating the performance of the final models as real-world datasets.

In this project, the supervised learning and unsupervised learning algorithms are used for evaluation. For supervised learning, it includes Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM), XGBoost, and Light Gradient Boosting Machine (LightGBM). While LR is act as the baseline model for comparing the model performance. While the unsupervised learning included the K-means, Hierarchical clustering (HC), DBSCAN, HDBSCAN, and Gaussian Mixture Model (GMM). The unsupervised learning models are mainly used for analyzing the account-based pattern, while the supervised model is mainly used for the detection of anomaly accounts.

In optimizing the model performance, the Elbow methods, Silhouette score, and Davies-Bouldin index are used for evaluation. Specifically, the elbow methods rely on the computation of sum of the squared errors (SSE), and the selection of cluster number is determined by the inflection point using the line graph. The formula for sum of the squared errors (SSE) is denoted in **Equation 3.1**.

$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} |p - mi|^2$$

Equation 3.1: Formula of sum of the squared errors (SSE), where there are k clusters; C_i represent the ith cluster among k clusters; p represent the data point within cluster C_i; mi is the centroids corresponding to the cluster C_i

The inflection point is the point that the curve of SSE become a fairly smooth curve against each value of k (i.e., number of clusters). It is suggested that the determination of the elbow points could be subjective due to the visual methods [27], for example, the graph in **Figure 3.2** showing that the optimized number of clusters may varies from 6 to 10. Hence, the evaluation of the algorithms is relying on other metrics.

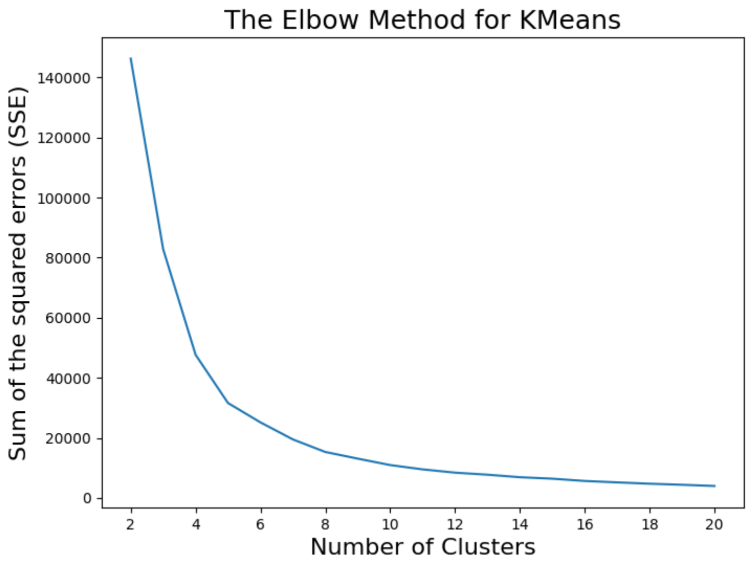


Figure 3.2: Graph showing the Elbow method for Kmeans algorithms with sum of squared errors (SSE) with different number of clusters

The silhouette score measures the extent for how data points fit into the dedicated clusters, which evaluate onto the similarity of intra-cluster and dissimilarity of inter-cluster similar. Silhouette score is ranging from -1 to 1, the higher its score indicate more separated clusters and increase the effectiveness of the clustering [28]. The formula for the score of silhouette analysis for each data point is displayed in **Equation 3.2**. The overall silhouette score is the means of silhouette coefficients for all data samples, which take account into distance between and within clusters.

$$S(i) = \frac{b(i) - a(i)}{\max\big(a(i), b(i)\big)}$$

Equation 3.2: Formula for computing the silhouette score in clustering analysis for each data point i. [28]. $S(i)$ compute the silhouette coefficients for each data point; $b(i)$ compute the mean distance within same clusters, $a(i)$ compute the mean nearest-cluster distance for each samples respectively

The Davies-Bouldin index is computed from two components, which involving of minimizing the intra-cluster variance, and maximizing the inter-clusters distance [29]. The Davies-Bouldin index is defined in **Equation 3.3**.

$$DB(k) = \frac{1}{k}\sum_{i,j=1}^{k} \max_{i \neq j}\left\{\frac{S_i + S_j}{d\big(\overline{x_\iota}, \overline{x_j}\big)}\right\}, \qquad where\ S_i = \frac{1}{n_i}\sum_{j=1}^{n_i} d\big(x_j, \overline{x_\iota}\ \big)$$

Equation 3.3: Formula for the computation of Davies-Bouldin index, defined with k clusters. S_i represent the average distance in ith clusters between data points x_j and cluster's center $\overline{x_\iota}$

Moreover, the significant features in affecting the performance of the models are determined by the utilization of Explainable Artificial Intelligence (XAI) methods, Shapley Additive Explanations (SHAP) is mainly used in this project. The SHAP techniques helps to interpret the importance of the features in the black-box models in machine learning, where the relation of input and the output of the models may be difficult to recognize with complex operations. In addition, multiple models are compared and evaluated in terms of accuracy, precision, and other essential elements. Consequently, a supervised machine learning model with highest effectiveness are employed into the application.

In technical aspects, Python and TypeScript are utilized for machine learning and deep learning, the models are either trained in the GPU farm from the Department of

Computer Science or the local machine with MacOS system. This can facilitate accomplishment of the model comparison and the integration of prediction model. The final model is hosted as API for distinguishing the types of accounts (see **Section 3.3** for details).

3.2 Graph Visualization of transaction history

To visualize the transaction history of an Ethereum account, several steps are depicted below, namely the real-time address data collection, filtered with number of unique address, and visualization of processed data. Initially, the input account address data with selected features is retrieved through the blockchain explorer APIs. Owing to considerable number of transactions for the account, certain important transactions are selected exclusively for enhancing the quality and readability for further visualization. To simplify the process, a reasonable number of latest transactions with the unique addresses are selected.

Subsequent to the transaction filtering, the prior data is visualized in the form of nose-link diagrams (demonstrated in **Figure 3.3**) and presented in the interface of the application, which provide interactive operation and customization of graph. In overall, the nodes are the account addresses for receive and send transaction associated with the input account, the links (or edges) are the amounts of transaction value and have predetermined pointing directions (send or receive from that account).

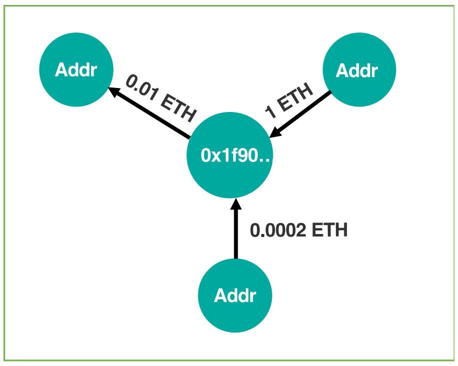


Figure 3.3: Demonstration of Node-link Diagram. The central node (green circle) is the input account, and the surrounding node are the account address that having transaction with the input account address. The arrow indicates the role of sender and recipients. The diagram is modified on demonstration purpose. Adapted from [30]

3.2.1 Basic Design of the Visualization

The *account* addresses represented as the nodes would have dynamic size, where the degrees of account nodes determine the size of the nodes. In other words, if the account is having more transactions, it is having the relatively larger size than the account addresses with fewer transactions.

Moreover, the *account* nodes are mainly with 3 colours according to their different transaction behaviours. The detailed types of nodes are:

✧ Green nodes colour for account only as sender in transactions (i.e., send transactions)

✧ Blue nodes colour for account only as recipients in transactions

✧ Red nodes colour for other types of account (i.e., account having send and receive transactions)

The directed edges connecting different nodes (i.e., accounts) represent the transactions between the relevant accounts with values in the unit of ETH (i.e., standard unit in Ethereum), indicating the flow of money along with arrow directions.

On the other hand, the visualized money flow graph is responsive with user interaction. The user can interact with the graph within the application to inspect the potential money flow direction and the related address features respectively (see **Figure 3.4**). In specific, the user can click the specific node (account address) to retrieve the analytics of transaction from several latest transactions. In addition, when user hover over the specific node (account address) (i.e., mouse floating over the node), the related analytics of transaction is showed in a dialog or a specific area.

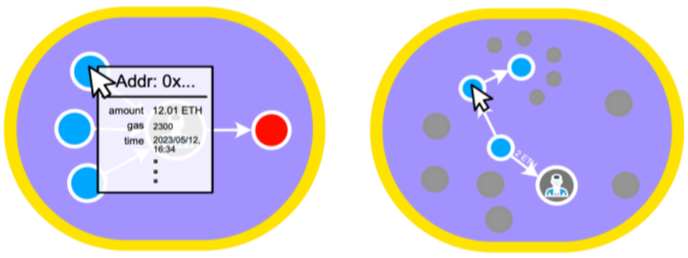


Figure 3.4: The interactive features of the visualized money flow graph for more details of accounts on click (figure on the left) and show only the associated accounts with transaction on hover over specific account node, where the hidden node are in grey color (figure on the right)

3.2.2 Visualisation Algorithms and Technology

For the transaction visualizers in the application, it is mainly composed of the force layout algorithm, the ForceAtLas2, and the JavaScript visualization library, SigmaJS. The former provides the visualization of transaction graph with scalability and flexibility, while the latter provide the features for user interaction and dynamic appearance of graph (i.e., colours, size of the nodes) within the application.

For the ForceAtLas2 algorithm, it converts the connected graph (with nodes and edges) into a force directed layout. In another words, the algorithms stimulate the physical system with the reaction of force that imitated by various attributes [31]. For instance, this may refer to the degree of the connected components and direction of directed edges. Consequently, the ForceAtLas2 algorithm spatialize the network, where the nodes are separated as if the repulsive force applied on the charged electrical particles [31]. Furthermore, the ForceAtLas2 algorithms utilized the techniques with the Barnes–Hut optimization, which effectively reduce the time complexity in generating the graph, from $O(N^2)$ to $O(N \log N)$, where N refers to the number of nodes (i.e., address in this project) [20].

On the other hand, SigmaJS is being the visualization libraries in JavaScript [32], which is designed for visualizing and interacting with the network graphs in the application [32]. For example, it allows user to interact with the nodes and edges for certain dynamic features such as hide the non-connected components on hovered and click to show more details. Moreover, it can be integrated with the ForceAtLas2 algorithms to visualize the graph in the form of money-flow graph. Therefore, the usage of ForcAtLas2 algorithms and SigmaJS helps to produce the money flow graph that could show the potential transaction pattern and have better interpretation of the data of the address.

3.3 Integration of Application and Technical Implication

In this project, the application has the graphical interface for input account address and predict the risks associated with potential anomaly activities. The user can input arbitrary Ethereum address to recognize the risks in 5 levels and visualize the latest transaction of the address in the form of money flow graph in the results page. The detail of the application is denoted in **Section 4.4**.

The application is primarily developed in Ionic framework and Angular as front-end. Since Ionic Framework provides cross platform application (in web and mobile), this allows convenient usage of forensics tools to inspect abnormal accounts in Ethereum. For the collection of the input's address and outcome of the model prediction, it is developed using the Python with the backend Flask server, which is deployed using the Microsoft Azure platform, in addition, it generates the data required for the visualization of the graph in the application. In the implemented API, it has basically two endpoints to handle the GET request from the application. The endpoints URLs and the example output structure are shown in **Figure 3.5** and **Figure 3.6**.

```
GET       /address/[Ethereum_Address]
```

(e.g., /address/0x0a52ecaa61268c6a5cf9cd6b1378531a4672601b)

```
Return: {
        "code": 200,
        "data": {
            "total_transaction_count": [
                    114
            ],
            "balance": [
                8.1990612963e-05
            ],
            … (more features not disclosed here)
        }
        "msg": "",
        "probability (illicit)": [
                99.8133
        ]
}
```

Figure 3.5: The example output for the GET Request for the endpoints /address/[Ethereum_Address], where [Ethereum_Address] is the custom Ethereum address to be input

GET   /transaction/[Ethereum_Address]

```
Return: {
        "code": 200,
        "data": [
                {
                    "blockNumber": "16266837",
                    "timeStamp": "1672032515",
                    "hash":
            "0xc598312d5b378c4beaff3045473cf22d9ff90109206e48c7baf2dab173989599",
                    "nonce": "13",
                    "blockHash":
            "0x783e56c0725b7958a18ea0f4491119f86368918764bc9fde5c169000fe8b428c",
                    "transactionIndex": "61",
                    "from": "0x0a52ecaa61268c6a5cf9cd6b1378531a4672601b",
                    "to": "0x9f12243d60c301d4e01a3d24bb620e8ffb40f855",
                    "value": "524217569903228006012",
                    "gas": "21000",
                    "gasPrice": "12221548679",
                    "isError": "0",
                    "txreceipt_status": "1",
                    "input": "0x",
                    "contractAddress": "",
                    "cumulativeGasUsed": "4996740",
                    "gasUsed": "21000",
                    "confirmations": "3386256",
                    "methodId": "0x",
                    "functionName": ""
            ],
            … (more transactions not disclosed here)
```

Figure 3.6: The example output for the GET Request for the endpoints /transaction/[Ethereum_Address], where [Ethereum_Address] is the custom Ethereum address to be input

4 Result and Discussion

In this section, the outcomes in the implementation of the Anomaly Account Detection are described and analyzed. It includes the data collected and the procedures of pre-processing for the anomaly detection (Section 4.1), the outcomes and analysis from the the supervised anomaly account detection (**Section 4.2**) and the unsupervised anomaly detection (**Section 4.3**), and for the visualization and application (**Section 4.4**) respectively.

4.1 Datasets Collection and Processing

4.1.1 Data Collection for anomaly detection

In the collection of labelled account data in Ethereum blockchain, there are 20802 accounts (10662 anomaly, 10139 normal accounts) collected from different sources (CryptoScamDB [22], Kaggle [23], academic journals [6] [13]) initially for anomaly accounts. While the normal accounts are mainly collected form the Ethereum main blockchain randomly within different period of time, they are cross-validate from those anomaly accounts. In overall, the account summary and latest 10,000 transaction records are collected with Blockchain explorer called the Etherscan.io [18] on or before the time of 10 November 2023. The last transaction time of the accounts is ranging from August 2015 to November 2023. The accounts without transaction records are denoted as invalid and are filtered in the view of the fact that no information is displayed to determine its nature (as normal or anomaly). After filtering the duplicate and invalid accounts, a total of 13941 accounts (5819 anomaly and 8122 normal accounts) are resulted, the distribution in percentage for the accounts is shown in **Figure 4.1**. However, due to the slight imbalance of datasets from number of anomaly and normal accounts, this may cause inaccuracy and overfitting of models training, in specific, having lower predictive performance for the minority class (i.e., anomaly account) [6] [33] [34].

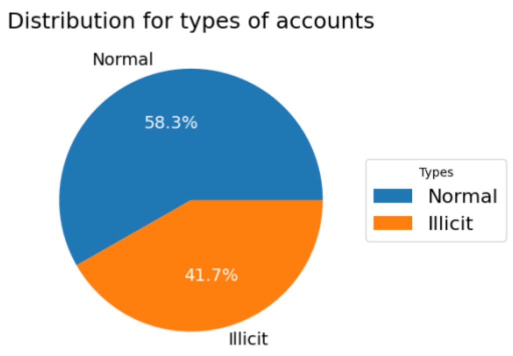


Figure 4.1: Distribution for types of collected accounts in form of pie chart

The collected data are mainly in the form of time-series data, where each of the transaction associated with address are recorded with the time of transaction. In definition, time-series data "is a sequence of observations taken sequentially in time" [35]. For utilizing these time-series data, the transaction data occurs at different time, it is required to convert the time series x_i for each sample address into lower dimensionality for machine learning that used for the supervised or unsupervised learning. The sample details of each transaction in one of the collected address is shown in **Figure 4.2**.The process of feature extraction from the time-series data is introduced in **Section** Error! Reference source not found..

```
{
        "blockHash":
        "0x29a0c0f7cc25a84f444deccb75c6716310cca4f36ffc9b060445bb256c6ac61c",
        "blockNumber": "13793208",
        "confirmations": "4641501",
        "contractAddress": "",
        "cumulativeGasUsed":    "4839801",
        "from": "0x00009277775ac7d0d59eaad8fee3d10ac6c805e8",
        "functionName": "",
        "gas": "21000",
        "gasPrice": "60313047492",
        "gasUsed": "21000",
        "hash":
        "0x9095c86bc7fa3a215da5b0ed5c6c27ee1c9888705aa65ccc14a462432336d01c",
        "input": "0x",
        "isError": "0",
        "methodId": "0x",
        "nonce": "753",
        "timeStamp": "1639349582",
        "to": "0xfe1b6aa4f75ae5475d29e2eaa9e5fe33871834e9",
        "transactionIndex": "58",
        "txreceipt_status": "1",
        "value": "465188923809954681"
}
```

Figure 4.2: The example data for the normal transaction with the address, which shows the various information regarding the transaction. Each transaction is with the timestamp that record the time of transaction

4.1.2 *Feature Extraction*

In processing of the transaction data collected in data preparation phrase, the python package, Tsfresh is used, which is the python library that automatically extract the features based on the time-series data. In particularly, it includes the several statistics computations such as minimum, maximum, mean of several attributes of the transaction. The Tsfresh have extracted more than 1,500 features for each address. While the large number of features may not be appropriate for the machine learning, where the performance of model may be affected. For instance, there is the curse of dimensionality, which may result in results in lack of model identifiability, instability, overfitting and numerical instabilities [25].

The features extraction is performed on the account data with selective attributes and information regarding its transaction history and status of the account. In filtering the features in the dataset for the model training, the correlation between different attributes, the feature importance, and the distribution of types of accounts are utilized. Moreover, there are addition features integrated into the dataset, these external features are determined with the several academic evidences and the heuristics [3], [6], [9], [13] - [15] that suggest the significance of certain attributes in the model predication.

The features extracted are generally divided by three types: time of transaction and between specific account status, fees related to transaction, and counting of specific occurrence of transaction. It is believed a comprehensive extraction of features is necessary for generalizing the behaviour of the account for the model fitting. In exanimating the features, *RStudio* is used for visualization of the distribution of features among normal and anomaly accounts. **Error! Reference source not found.** shows the distribution of first and last transaction time for the normal and anomaly accounts respectively. Although the normal accounts are collected randomly, the first and last transaction may be primarily distributed at specific time, for example, near the time of 2018. While the density of transaction time for anomaly account may be distributed more evenly than normal accounts. It is observed than a considerably low density of first transaction time of normal account as compared to that of anomaly accounts, while it may not provide any information for nature of accounts by mere first time of transaction. Hence, the specific time of transaction is excluded from the features to avoid overfitting and false detection of the models in training and testing phrases.

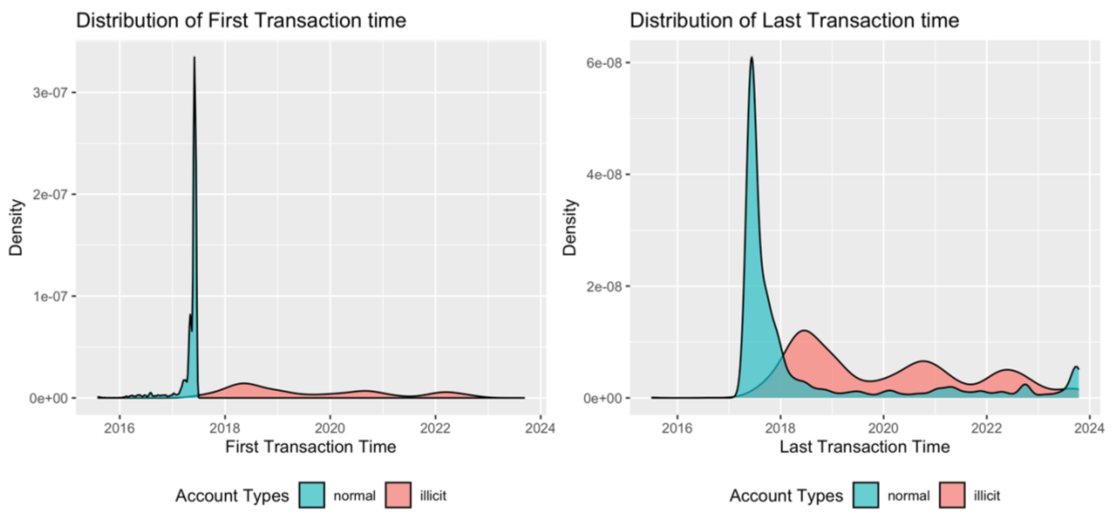


Figure 4.3: Distribution of first transation time (on left) and distribution of last transaction time (on right) among the two types of account

As in **Error! Reference source not found.**, it displays the distribution of total transaction count among normal and anomaly accounts respectively. The density plots of normal and anomaly accounts are distributed similarly except for certain range of transaction count.

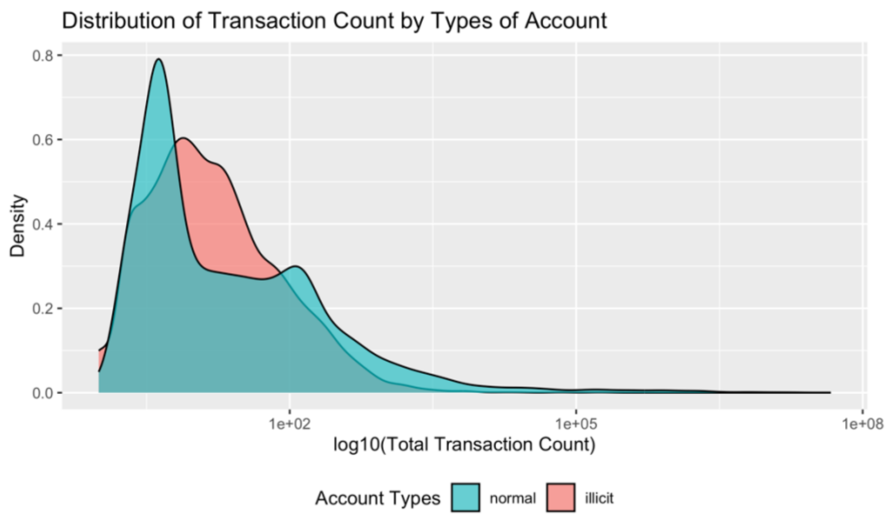


Figure 4.4: Distribution of transaction count among normal and anomaly of account

With the specific transaction mechanism in Ethereum (as stated in Section 2.1), it is suggested the criminals (owner of the anomaly account) (e.g., related to scamming) may set a higher gas price to provide higher incentive for validators to include this transaction at a higher speed [10]. As a result, the features related to gas, gas price, and the transaction fee are extracted from the transactions of each account. In addition, the number of unique address that transacted with the targeted address is extracted other than the auto-generated features from the Tsfresh library. A full list of 64 filtered extracted features is presented in **Error! Reference source not found.**.

	Feature name	Description	Data type
1	balance	The current balance of the Ethereum account	Float
2	transaction_count	The total number of transactions made by the account	Integer
3	send_amount	The total amount sent by the account	Float
4	receive_amount	The total amount received by the account	Float
5	token_amount	The total amount of tokens held by the account	Float
6	total_token_value	The total value of tokens held by the account	Float
7	total_transaction_count	The total number of transactions made by the account, including internal transactions	Integer
8	num_of_normal_transaction	The number of regular transactions made by the account	Integer
9	out_transaction_percent	The percentage of outgoing transactions compared to total transactions	Float
10	in_transaction_percent	The percentage of incoming transactions compared to total transactions	Float
11	max_val_send	The largest amount sent by the account in a single transaction	Float
12	min_val_send	The smallest amount sent by the account in a single transaction	Float
13	mean_val_send	The average amount sent by the account in all transactions	Float
14	stdev_val_send	The standard deviation of the amounts sent by the account in all transactions	Float
15	max_val_recv	The largest amount received by the account in a single transaction	Float
16	min_val_recv	The smallest amount received by the account in a single transaction	Float
17	mean_val_recv	The average amount received by the account in all transactions	Float
18	stdev_val_recv	The standard deviation of the amounts received by the account in all transactions	Float
19	max_gas_price	The highest gas price paid by the account in a single transaction	Float
20	min_gas_price	The lowest gas price paid by the account in a single transaction	Float
21	mean_gas_price	The average gas price paid by the account in all transactions	Float

22	stdev_gas_price	The standard deviation of the gas prices paid by the account in all transactions	Float
23	mean_transaction_fee	The average transaction fee paid by the account in all transactions	Float
24	max_transaction_fee	The highest transaction fee paid by the account in a single transaction	Float
25	min_transaction_fee	The lowest transaction fee paid by the account in a single transaction	Float
26	stdev_transaction_price	The standard deviation of the transaction fees paid by the account in all transactions	Float
27	uniq_send_address_num	The number of unique addresses the account has sent transactions to	Integer
28	uniq_receive_address_num	The number of unique addresses the account has received transactions from	Integer
29	zero_val_tx_num	The number of transactions with a value of 0	Integer
30	zero_val_send_tx_num	The number of outgoing transactions with a value of 0	Integer
31	zero_val_recv_tx_num	The number of incoming transactions with a value of 0	Integer
32	mean_time_between_tx	The average time between transactions made by the account (in second)	Float
33	mean_time_between_send_tx	The average time between outgoing transactions made by the account (in second)	Float
34	mean_time_between_recv_tx	The average time between incoming transactions made by the account (in second)	Float
35	highestBalance	The highest balance the account has had	Float
36	lowestBalance	The lowest balance the account has had	Float
37	num_of_internal_transaction	The number of internal transactions made by the account	Float
38	internal_out_transaction_percent	The percentage of outgoing internal transactions compared to total internal transactions	Float
39	internal_in_transaction_percent	The percentage of incoming internal transactions compared to total internal transactions	Float
40	internal_max_val_send	The largest amount sent in a single internal transaction by the account	Float
41	internal_min_val_send	The smallest amount sent in a single internal transaction by the account	Float
42	internal_mean_val_send	The average amount sent in all internal transactions by the account	Float
43	internal_stdev_val_send	The standard deviation of the amounts sent in all internal transactions by the account	Float
44	internal_max_val_recv	The largest amount received in a single internal transaction by the account	Float
45	internal_min_val_recv	The smallest amount received in a single internal transaction by the account	Float
46	internal_mean_val_recv	The average amount received in all internal transactions by the account	Float
47	internal_stdev_val_recv	The standard deviation of the amounts received in all internal transactions by the account	Float
48	internal_max_gas	The highest gas price paid by the account in a single internal transaction	Float
49	internal_min_gas	The lowest gas price paid by the account in a single internal transaction	Float
50	internal_mean_gas	The average gas price paid by the account in all internal transactions	Float
51	internal_stdev_gas_price	The standard deviation of the gas prices paid by the account in all internal transactions	Integer

52	internal_uniq_send_address_num	The number of unique addresses the account has sent internal transactions to	Integer
53	internal_uniq_receive_address_num	The number of unique addresses the account has received internal transactions from	Integer
54	internal_zero_val_tx_num	The number of internal transactions with a value of 0	Integer
55	internal_zero_val_send_tx_num	The number of outgoing internal transactions with a value of 0	Integer
56	internal_zero_val_recv_tx_num	The number of incoming internal transactions with a value of 0	Integer
57	internal_mean_time_between_tx	The average time between internal transactions made by the account (in second)	Float
58	internal_mean_time_between_send_tx	The average time between outgoing internal transactions made by the account (in second)	Float
59	internal_mean_time_between_recv_tx	The average time between incoming internal transactions made by the account (in second)	Float
60	time_diff_between_min_balance_and_first_tx	The time difference between the first transaction made by the account and when the balance was at its lowest (in second)	Float
61	time_diff_between_max_balance_and_first_tx	The time difference between the first transaction made by the account and when the balance was at its highest (in second)	Float
62	time_diff_between_min_balance_and_last_tx	The time difference between the last transaction made by the account and when the balance was at its lowest (in second)	Float
63	time_diff_between_max_balance_and_last_tx	The time difference between the last transaction made by the account and when the balance was at its highest (in second)	Float
64	time_diff_between_first_and_last_tx	The time difference between the first transaction and last transaction from the account (in second)	Float

Table 4.1: Table listing the 64 extracted features of account data with descriptions for model fitting and prediction

4.1.3 Data Pre-processing

In data preprocessing, normalization of the data values is important for model fitting and classification [36]. The purpose of normalization is to transform the data into the narrow and similar scale, which favor several machine learning algorithms that compute the distances between or within different features [36]. In general, there are two types of widely used approaches used for testing: Min-Max Normalization and Standardization (or called Zero-value Normalization) [36].

The Min-Max Normalization uses the minimum and maximum value to transform the data into the fixed bound, mostly referred to range between 0 and 1, or between -1 and 1 [36]. It is effective when the data distribution is unknown, however, algorithms' performance would be affected by the value outside the minimum and maximum value (called the "outliers") used in model fitting [37] [38]. The equation of Min-Max Normalization is shown in **Error! Reference source not found.**.

$$x_{new} = x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Equation 4.1: *Equation of the Min-Max Normalization for the calculation of the scaled value; x_{new} and x_{norm} are the scaled value; x is the original variable value; x_{min} is the minimum value among the data points, x_{max} is the maximum value among the data* [36]

Standardization uses the Z-score, consisting of the mean data value and the standard deviation. It does not scale the data in the fixed range as compared to the Min-Max Normalization. It is more effective when the data follow the normal (or gaussian) distribution [38]. The equation of the Standardization is shown in **Error! Reference source not found.**.

$$x_{new} = Z = \frac{x - \bar{x}}{\sigma}$$

Equation 4.2: Equation of the Standardization (or Z-Score Normalization) for the calculation of the scaled value; x_{new} and Z are the scaled value; x is the original variable value; $\bar{x}$ is the variable mean value; σ is the variable standard deviation [36]

For the Ethereum account and transaction data, it is assumed majority of the features align in the gaussian distribution due to the substantial amount of transaction in the blockchain. For evaluating the effect between the two types of normalization, experiment is performed in different machine learning algorithms including logistic regression, KNN, SVM, and other tree-based algorithms. With the use of k-fold cross validation, all the non-tree-based algorithms used Standardization have a higher performance from 1% to 8% in accuracy, macro F1, and weighted F1 as compared to the Min-Max Normalization. While the tree-based algorithms: RF, XGBoost, and LightGBM are slightly influenced, about 1% higher F1 in Standardization as compared to the Min-Max Normalization, which clearly reflected that tree-based algorithms are less impacted by the normalization or feature scaling with existing research [39]. In the view of the higher performance of the standardization than the Min-Max Normalization, therefore, Standardization is being adapted in the phase of data pre-processing currently.

For the data imbalance issues, the relatively low samples of minority classes (the anomaly accounts) would cause the biased prediction on the model performance where misleading conclusion may be resulted. Consequently, to mitigate the issues, several strategies are considered and are experimented. In general, it has utilized the under-sampling, over-sampling, which could adjust the amount of the majority or minority class to obtain a more balanced dataset. Logistic Regression is used as the baseline model for the model fitting used the data applying Adaptive Synthetic Sampling (ADASYN), Edited Nearest Neighbor (ENN), Synthetic Minority Oversampling Technique (SMOTE), and Tomek Links.

The comparison of Receiver Operating Characteristic (ROC) curves with the unsampled dataset and the various sampling methods is shown in **Figure 4.5**.

In the experiment, the Area Under Curve (AUC) for merely ENN having the lowest value of 0.8212, while dataset sampled with SMOTE and Tomek Links having highest AUC of 0.8558.

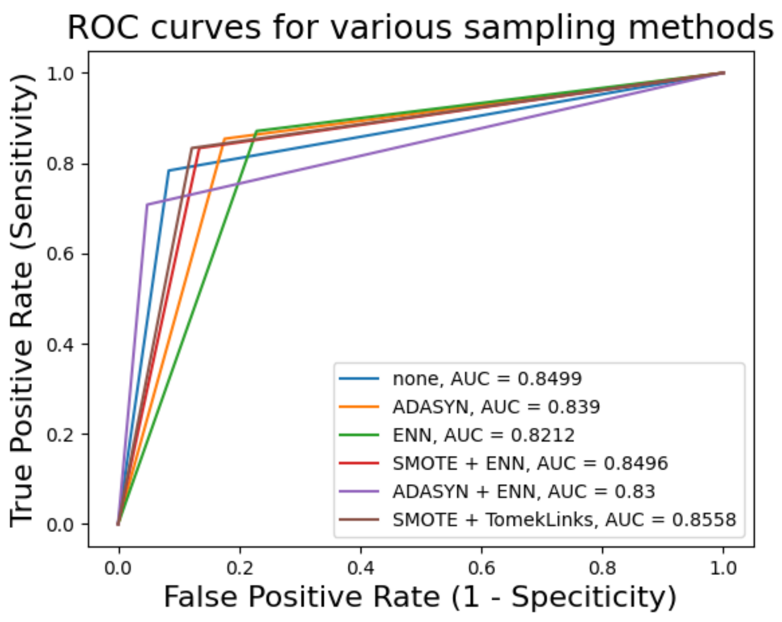


Figure 4.5: ROC curves for the various sampling methods including ADASYN, ENN, and SMOTE with ENN, and TomekLinks, and the data without sampling using Logistic Regression

In comparing the ROC curves between original datasets and that sampled with SMOTE and Tomek Links, a higher true positive rate (sensitivity) is resulted from sampling of SMOTE and Tomek Links. Hence, it probably referring the higher performance in recognizing the accounts as anomaly given that the accounts are anomaly. Since the purpose of this project is to detect the anomaly and malicious account, the sampling for the dataset have used the algorithms of SMOTE and Tomek Links.

To validate the performance of the sampling, the another algorithms including RF, KNN, and LightGBM are tested using k-fold cross validation, it has achieved about 1% to 10% increase in precision, recall, and specificity. The distribution of the types of account in the training dataset are shown in **Figure 4.6**, which resulted in 50.0% of normal and anomaly account respectively.

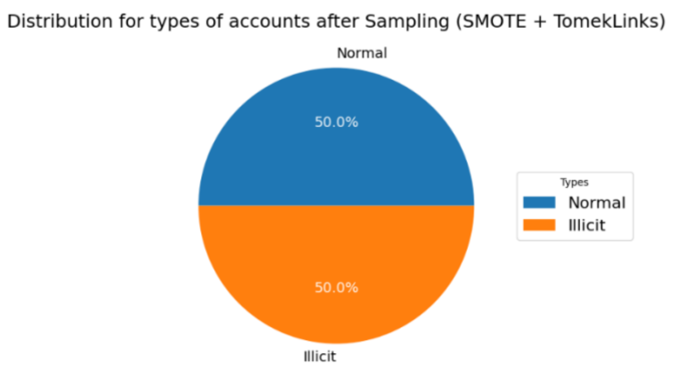


Figure 4.6: Distribution for types of Ethereum accounts in dataset after the SMOTE and TomekLinks sampling in pie chart

4.2 Supervised Anomaly Account Detection

The datasets are used for the model training in the supervised learning for the anomaly account detection (or classification in machine learning). The results are organized in the performance in model evaluation (**Section 4.2.1**), SHAP analysis (**Section 4.2.2**), and the limitation of the supervised classifier (**Section 4.2.3**).

4.2.1 Model Evaluation

For the model evaluation, the positive class is the anomaly account and negative class is the normal account in this project for calculation the scoring metrics. The evaluation is tested on Logistic Regression (LR), RF, KNN, XGBoost, LightGBM, SVM, and the stacking of five algorithms (SVM, RF, KNN, LR, and decision tree (DT)). The performance is evaluated using accuracy, precision, recall, specificity, F1, macro and weighted average of F1. Nevertheless, considering the biased accuracy resulting from the uneven distribution of the accounts in the test data (which is not sampled to have better evaluation of the final results). Hence, the recall, and specificity, and F1 would be more effective in evaluating the model performance. Since it is more important to recognize the anomaly account as positive instead of negative, it should be lower the false negative rate (FNR), which indicates for a higher recall. While specificity is considered along with recall. The **Equation 4.3** and **Equation 4.4** show the equations for recall and specificity respectively.

$$Recall = TPR = \frac{TP}{TP + FN}$$

Equation 4.3: The equation for calculating the recall, which is also referred to true positive rate (TPR). TP stands for true positive while FN stands for false negative

$$Specificity = TNR = \frac{TN}{TN + FP}$$

Equation 4.4: The equation for calculating the specificity, which is also referred to true negative rate (TNR). TN stands for true negative while FP stands for false positive

Table 4.2 summarize the performance of various machine learning algorithms in model training using the k-fold cross validation, with k equals to 10. In the evaluation, the baseline model, LR have an accuracy of 79.48%, while having a significantly lower performance in recall of 64.69%. Among all tested models, the models based on Gradient Boosting Decision Tree (GBDT) (i.e., LightGBM and XGBoost) have the highest performance. LightGBM achieved the highest performance with average accuracy of 96.96%, recall of 96.22% with average false positive rate (FPR) of 1.86% and false negative rate (FNR) of 3.78%. While XGBoost shared similar scoring as LightGBM, with slightly lower performance.

Method	**Accuracy**	**Precision**	**Recall (TPR)**	**Specificity (TNR)**	**F1**	**Macro_F1**	**Weighted_F1**
Logistic Regression	0.8478	0.8771	0.8090	0.8866	0.8416	0.8476	0.8476
Random Forest	0.9703	0.9708	0.9699	0.9707	0.9703	0.9703	0.9703
KNN	0.9303	0.9282	0.9328	0.9278	0.9304	0.9303	0.9303
XGBoost	0.9753	0.9753	0.9754	0.9752	0.9753	0.9753	0.9753
LightGBM	0.9804	0.9806	0.9803	0.9805	0.9804	0.9804	0.9804
SVM	0.9251	0.9396	0.9085	0.9416	0.9236	0.9250	0.9250
Stack (RF, KNN, LR, DT, SVM)	0.9705	0.9707	0.9703	0.9706	0.9704	0.9705	0.9705

Table 4.2: Evaluation of various machine learning algorithms using k-fold cross validation, using scoring metrics including accuracy, precision, recall, F1, macro F1, and weighted F1; The text in red indicate the highest scoring and method with highest performance

In addition, the deep learning approaches are tested using the python Keras and Tensorflow library. For instance, it has tested with the single layer perceptron, Shared

Feature Extraction Layer. However, they are not performed better than the machine learning methods in the **Table 4.2**. In the table, it indicated that the LightGBM and XGBoost having similar performance while both of them are types of ensemble learning approaches. In the comparison among LightGBM and XGBoost, LightGBM have certain benefits such as faster speeds and less memory consumption in predication (or training) [40] [41].

Furthermore, in testing the performance of LightGBM on the classification. The model has achieved at least 98.0% among normal and anomaly classes in all the scoring metrics (precision, recall, f1-score) in macro and weighted average respectively, with false positive rate of 1.23% and false negative rate of 2.75%. Nonetheless, all the metrics for anomaly classes are lowered than the normal class, it is probably owing to the fewer data of anomaly accounts in training phase, causing more incorrect prediction [33].

Although It shows a relative few number of false negative and positive respectively, resulting in about 2% incorrect predictions, could result in considerable amount of incorrect prediction in the real blockchain data. Moreover, there are more false negative predictions than false positive, which may result in inability to detect anomaly behaviour if presence. Nevertheless, this approximation may be rather simplified since the data distribution and behaviours of the anomaly accounts could be more complicated, resulting in lower performance. Consequently, a larger amount and coverage of testing data is required to have a more comprehensive analysis of the performance of the classifier.

4.2.2 SHAP analysis

With the usage of the SHAP techniques, it has found the top 10 features influence the model decisions on classifying anomaly accounts to a large extent (see **Figure 4.7**), which are number of unique address that received from, the minimum gas price in transactions, mean gas price in transaction, total sending amount in transaction, time between different status and other transaction related attributes, the description of these features are indicated in **Error! Reference source not found.**. In **Figure 4.7**, it indicated that a higher number of unique address send into the account have a larger positive impact for the model to determine the account as anomaly. From the gas price, the malicious account may pay higher gas price in the transaction for fast payment [10], however, the value of maximum gas price indicate a less impact on the model prediction. In summarize, the top 10 important features may show certain of the characteristics of those anomaly account on Ethereum to some extent, while it may be biased owing to uneven distribution of the account.

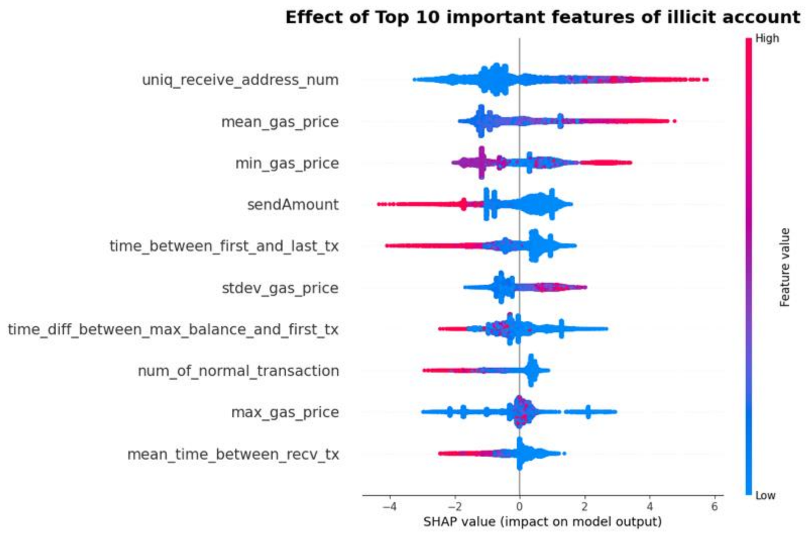


Figure 4.7: The effect of top 10 important features of the anomaly account. The higher SHAP value indicate a favorable and positive influence for determine the account as anomaly, and vice versa.

4.2.3 Limitations of the Classifier

In the collection of the transaction data for each account, the latest 10,000 transaction records for each of normal and internal transaction are collected and analyzed instead of all the historical transactions. These restrictions are established by the blockchain explorer APIs to limiting the computing resources [18]. Although there are certain strategies that can retrieve all the historical transaction records, majority of them are time consuming. For instance, the Google BigQuery [42] that having the public Ethereum blockchain data, would require considerable amount of time for retrieving the full transaction records, thus, may not be practical owing to the limited time, while extra expenditure may be induced for those querying. For the collected data, there are less than 1.09% of the collected account have more than 10,000 transaction records among the 13,941 accounts. Their transactions count is ranging from 10,269 to 46,335,494. Hence, the time for retrieving the transaction of these accounts may be significantly large. Instead of all the historical transaction, it is assumed that the latest 10,000 transaction is sufficient for generalizing the latest behaviour of that account, which may pose restriction on change of behaviour before those 10,000 transactions.

On the other hand, there are the restraints on evaluating the effectiveness of the classifier for the recognition of the malicious account, owing to the scarce and fixed labeled data available amount of dataset is certainly not adequate to represent the majority of the existing Ethereum blockchain data, which these data merely account for less than 1% among the daily active Ethereum unique address [43]. In addition, the classifier that merely rely on the supervised machine learning algorithms may not be reliable without knowing the real types of the accounts, false prediction may be generated on the real-world scenarios that without the account labels. In comparing with several academic paper that detecting the anomaly account in Ethereum blockchain. The performance of model is fairly similar to the equivalent research [10] [13] [21] [34] while it is merely relies on the quality of the labels for the quality of the datasets.

As a result, the greatest challenge is the validity of the unlabeled normal account, the accounts without anomaly label are not necessary to be normal. In addressing these issues, unsupervised learning is tested in **Section 4.3** for a more comprehensive classification without predetermined labels. In addition, To improve the quality of the classifier, it might need to adapt with the new data to recognize the uncovered pattern, for example, query from open-source database for that account address from user input and used for re-training in real-time. As a result, other methods must be adapted in addition with the ML models to prove a more reliable classification results.

4.3 Unsupervised Anomaly Detection

4.3.1 Dimension Reduction

For unsupervised learning, the dimension reduction is necessary and vital for interpret the pattern of the data under the model fitting and prediction. In this phrase, the techniques of PCA and T-SNE are used and compared. In PCA, the explained variance is a measure of the total variance in the original dataset is explained by each principal component. It indicates the cumulative explained variance ratio (EVR) for different number of components, where the cumulative EVR represent the extent of information retained from the original dataset. The maximum cumulative EVR is 1 for the summation of EVR of each components. In the result, the cumulative EVR is larger than 0.95 for more than 37 components. While the cumulative EVR reach 1 if more than 59 components. The PCA is conducted on the Ethereum dataset with different number of components extracted as shown in **Figure 4.8**.

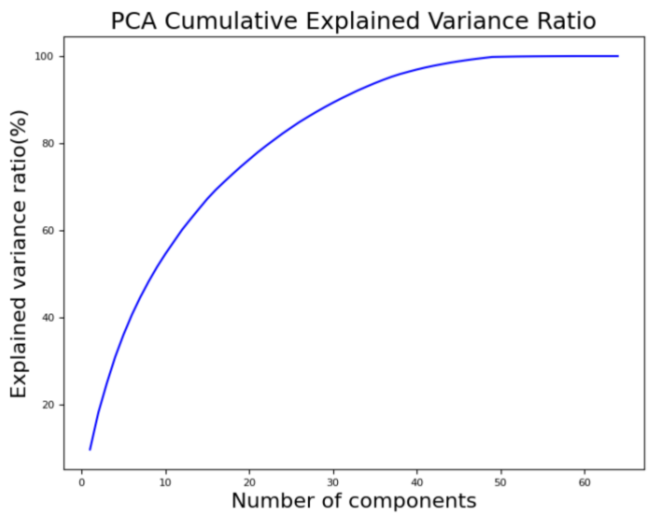


Figure 4.8: The PCA cumulative explained variance ratio among different number of components

4.3.2 Clustering

In contrast with supervised learning, the unsupervised learning does not rely on the ground truth labels of the accounts. It is essential that the unlabeled account (i.e., normal) may not be normal, while they are not flagged at the time of data collection. In the project, it has used with K-means, Hierarchical clustering, DBSCAN, HDBSCAN, and GMM for the clustering of the account. To evaluate the effect of the clustering, a range of number of clusters are being used.

All the tested model having higher performance in clustering with number of clusters more than 2, which is different from the account’s labels collected. The optimized number of clusters is ranging from 4 to 16 among the clustering models. In K-means, Hierarchical clustering, HDBSCAN, and GMM, more than 70% of the accounts are within the same clusters. While for DBSCAN, the distribution of the accounts in clusters is more varies where the highest amount of account in clusters is about 34%. In optimizing the best number of clusters for various clustering algorithms, the corresponding hyperparameter are selected and tested. The unsupervised clustering models are fine-tuned with the optimization of the hyperparameters for highest Silhouette score, the hyperparameter for optimization and their respective values are showed in **Table 4.3**.

Model	**Hyperparameter for optimize**		**Values for Testing**		**Adapted**
K-Means	n_clusters		2,3,4,5,6,7,8,9,10		7
Hierarchical Cluster	n_clusters				7
DBSCAN	min_samples	eps	min_samples: 2, 3, 4, 5, 6, 7, 8, 9	eps: 0.001 to 1 (separated by 0.05, end exclusive)	min_samples: 9
					eps: 0.951

HDBSCAN		min_cluster_size		min_cluster_size: 10 to 1000 (each separated by 20, end exclusive)	min_samples: 4
					min_cluster_size: 480
Gaussian Mixture Model	n_components		2,3,4,5,6,7,8,9,10		4

Table 4.3: The hyperparamter used for testing and adapted for optimization of the unsupervised clustering models of K-means, Hierarchical clustering, DBSCAN, HDBSCAN, and GMM

The evaluation of unsupervised clustering models including K-Means, Hierarchical Cluster, DBSCAN, HDBSCAN, and GMM are listed in **Table 4.4**. It is showed that the models using the Hierarchical Cluster achieved the highest Silhouette score of 0.615 and the lowest Davies-Bouldin index, followed by the K-means algorithms. The number of clusters among these algorithms are ranging from 4 to 16, the percentage of outliers is ranging from about 7% to 17.8%.

Model	**number of Clusters**	**number of Outliners**	**Silhouette score**	**Davies-Bouldin index**
K-Means	7	0	0.412	2.110
Hierarchical Cluster	7	0	0.615	0.968
DBSCAN	16	2549	0.185	2.112
HDBSCAN	4	995	0.364	3.121
Gaussian Mixture Model	4	0	0.111	3.138

Table 4.4: The Evaluation using Silhouette score and Davies-Bouldin index among the unsupervised models: K-means, Hierarchical clustering, DBSCAN, HDBSCAN, and GMM

On the other hand, the performance of unsupervised clustering is compared with the two types of dimension reduction methods: PCA and T-SNE. The difference in performances among these techniques are merely slightly with the wide range of scores owing to the hyperparameter, where the PCA achieved slightly higher for less than 1%

in Silhouette score and Davies-Bouldin index. For demonstration purpose, the variance in clustering for PCA and t-SNE with 3 principal components are plotted with the 3-dimensional graph. The **Figure 4.9** and **Figure 4.10** shows the 3-dimensional graph for the clustering results for Hierarchical clustering and K-means clustering respectively.

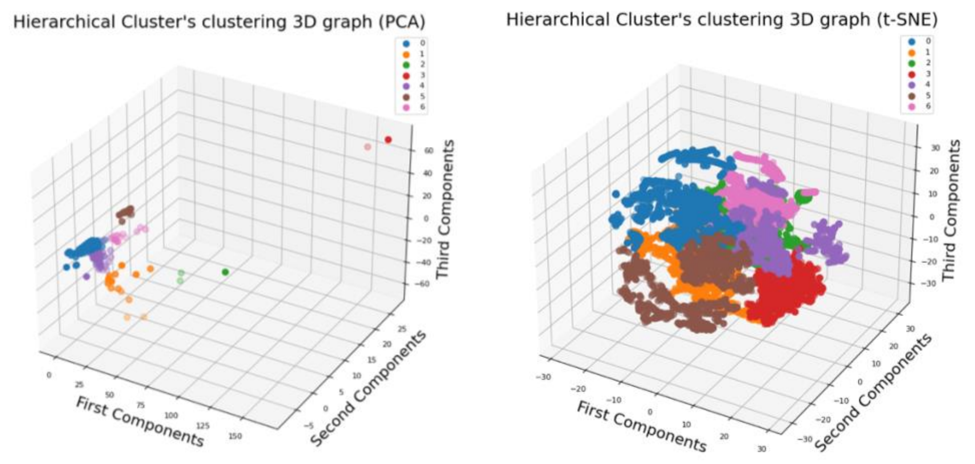


Figure 4.9: The clustering graph in Hierarchical Clusters with three principal components, with the use of PCA (Principal Component Analysis) (figure on left) and t-SNE (T-Distributed Stochastic Neighbor Embedding) figure on right)

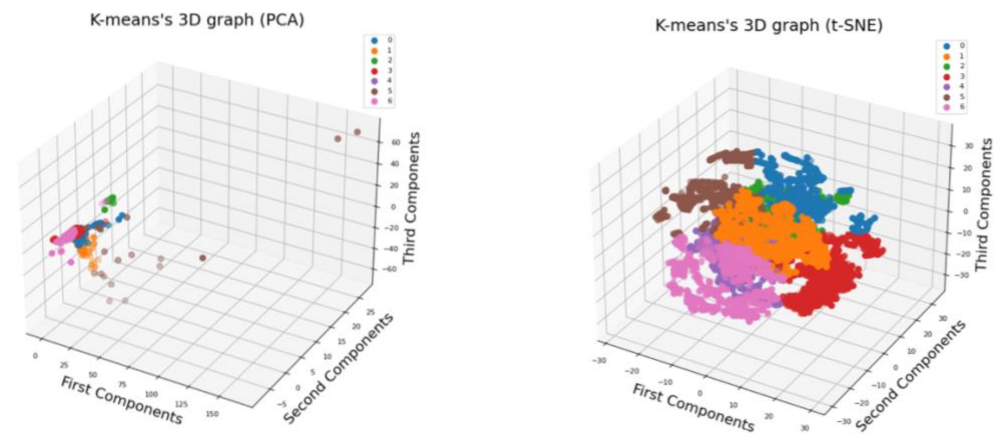


Figure 4.10: The clustering graph in K-means with three principal components, with the use of PCA (Principal Component Analysis) (figure on left) and t-SNE (T-Distributed Stochastic Neighbor Embedding) figure on right)

For Hierarchical Clustering, it mainly involved of two strategies: agglomerative and divisive, which referred to “bottom-up” and “top-down” methods respectively [44]. In this project, the agglomerative Hierarchical Clustering is used with the measurement of distance using Ward’s method and squared Euclidean distance (which is one of the linkage methods). One of the features of Agglomerative Hierarchical Clustering (AHC) is the output of the dendrogram, which describe the algorithm’s procedures as tree structure (with the technique of tree traversal) [44]. The dendrogram of the AHC from the model’s performance is shows in **Figure 4.11**.

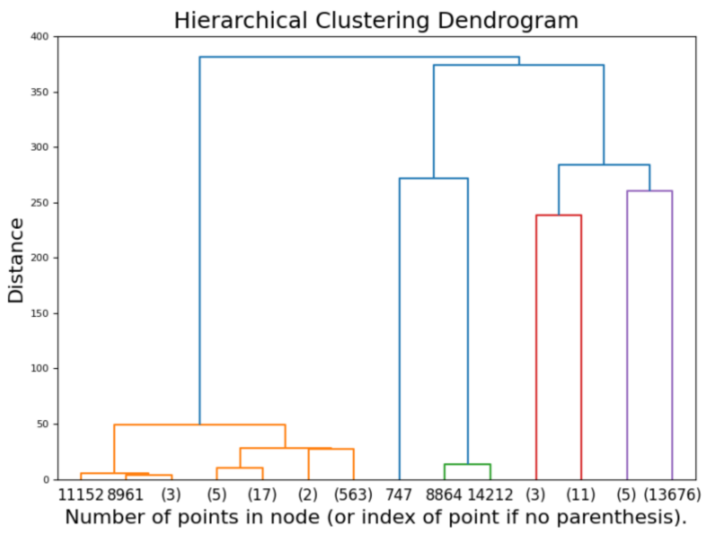


Figure 4.11: The Dendrogram for Hierarchical Clusters with number of points in node and corresponding distance, in range of three levels

The percentage of anomaly accounts is computed in each clusters under Hierarchical Cluster and K-means algorithms, which is displayed in **Figure 4.12** and **Figure 4.13** respectively. for Hierarchical Cluster, 90% of the datapoints are within one clusters. While for clusters of K-means, it indicates that there are 2 clusters contains all 100% labelled anomaly accounts, followed by a clusters with more than 50% of these labelled accounts.

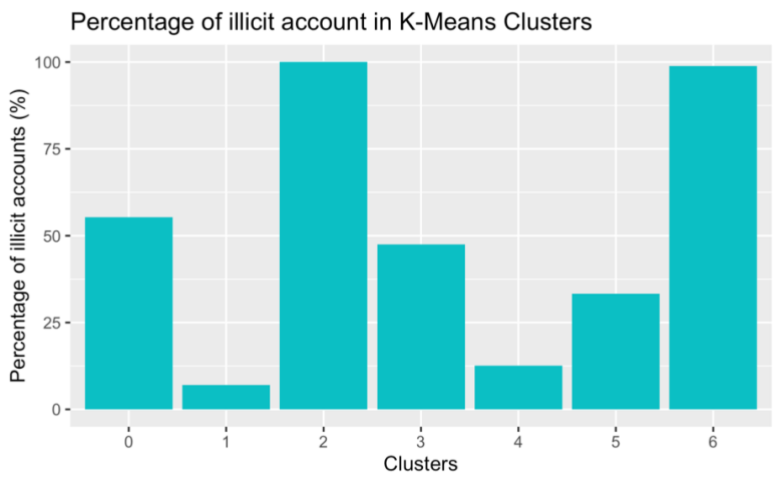


Figure 4.12: The percentages of anomaly accounts among different k-means clusters, with 7 clusters

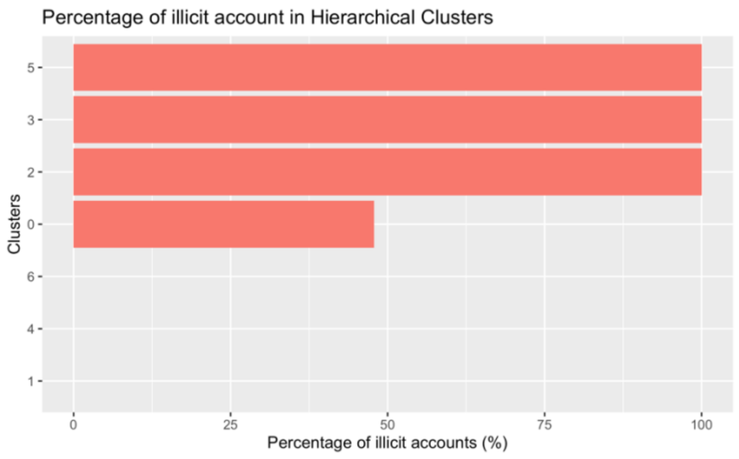
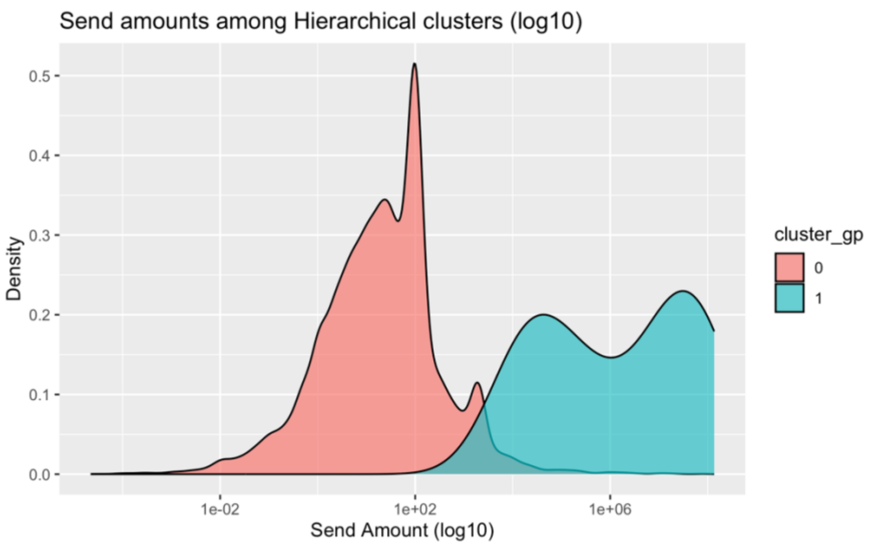


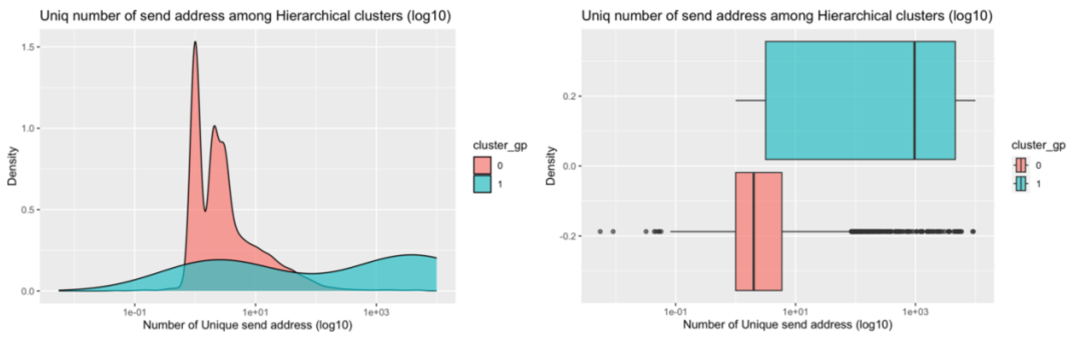
Figure 4.13: The percentages of anomaly accounts among different Hierarchical clusters, with 7 clusters

Although Hierarchical Cluster achieved the highest performance among the Silhouette score and Davies-Bouldin index, majority of the datapoints are within the single clusters. In comparison of the clusters, the minority data points in all other clusters are denoted as group 1 and the majority datapoints is denoted as group 0. The distribution of send amounts and number of unique send addresses are displayed in **Figure 4.14** and **Figure 4.15**



<u>**Figure 4.14**: The distribution of the send amounts among Hierachical clusters in density plot. The cluster group 0 indicate the single cluster contains more than 90% datapoints, while the clutser groups contains all the data from another clusters</u>

In the representation as box plot (**Figure 4.15**, it indicates there are considerable number of outliers for the group 0 (contains most of the datapoints), which may point out that relative low effectiveness in visuals and analyze the features in form of box plotting.



<u>**Figure 4.15**: The distribution of the number of unique send address among Hierachical clusters in density plot and box plot respectively. The cluster group 0 indicate the single cluster contains more than 90% datapoints, while the clutser groups contains all the data from another clusters</u>

The four other features regarding time, number of zero transaction, and transaction are displays in **Figure 4.16**. It indicates the minority clusters may have varies feature value as compared to the majority of data points in the single clusters.

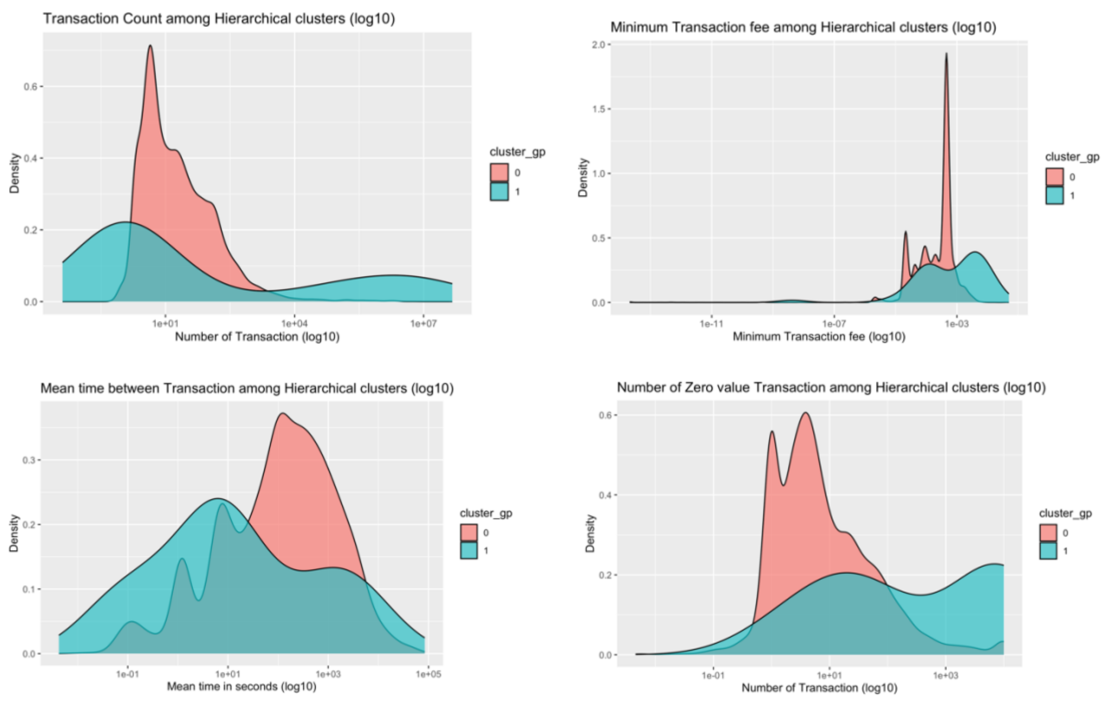


Figure 4.16: The distribution of the features for transaction count, minimum transaction fee, mean time between transaction and number of zero value transaction in the four figure respectively

Although the accounts are unlabeled, it is possible that these account change their behaviour between the data collection phrase or not revealed from the anomaly account database or certain sources. It may influence the performance of the clustering for grouping the datapoints into different categories.

4.3.3 Anomaly Outlier Detection

Concurrently, the anomaly outlier detection algorithms are being experimented using the Isolation Forest (IForest) and OneClassSVM. These algorithms detect the outliers by establishing the decision border (or boundary), where the anomaly data is recognized the datapoints align outside the boundary that the models learn from the training data. In comparison, the OneClassSVM having the higher specificity from 70.4% to 97.8% and FPR of 2.2% for the hyperparameter of the contamination fraction from 0.4 to automatic (default as 0.01), while having lower recall that less than 70%, while the IForest having a lower specificity of 70.9% to 92.3% and much lower recall. The contamination fraction decides the relative outlier fraction for the rough percentage of the anomaly datapoints in the datasets. While in the testing, the lower the contamination fraction, the higher specificity and lower FPR.

In the outlier detection, the anomaly values are mainly involving of the extreme values. For example, for the values near to the lowest or the highest values from the features. The **Figure 4.17** indicate the histogram for the anomaly datapoints detected in the IForest for the number of unique receive addresses.

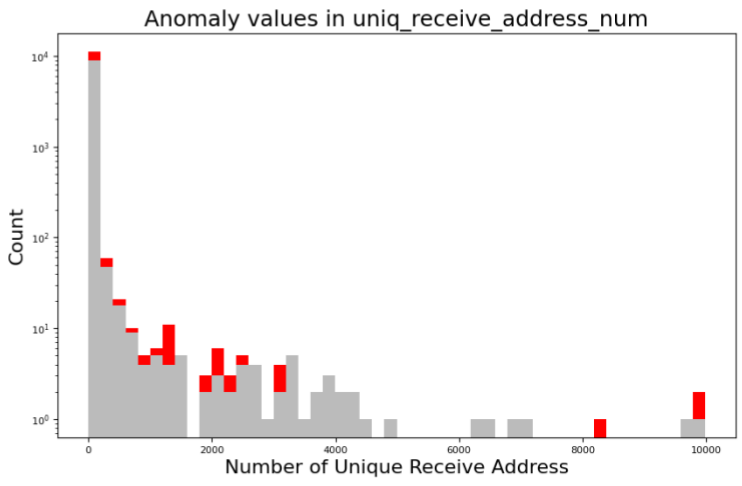


Figure 4.17: The histogram for the distribution of the anomaly values in number of unique receive address. The red part of the bar indicates the detected anomaly datapoints while the grey color indicate other datapoints

4.3.4 Comparison of supervised and unsupervised learning

In the unsupervised learning, the algorithms for clustering and the detection of anomaly datapoint with the utilization of decision boundary are having lower performance than the supervised learning in **Section 4.2**. However, the limitation of the unsupervised algorithms is primarily regarding the lack of ground-truth labels. In the unsupervised model evaluation in **Section 4.3**, it depends on the positive recognized anomaly labels (i.e., the labels for anomaly accounts) while attach less importance on the unlabeled normal accounts. Although the unsupervised learning such as OneClassSVM achieved a higher specificity and lower FPR at a lower outlier fraction (1-2% of the datasets). It may indicate certain amount of the anomaly accounts may not reveals their anomaly behaviours with existing features.

In the improvement of the performance in detecting the anomaly accounts in blockchain, the unsupervised learning methods may provide more information on the anomaly behaviour of the accounts. Meanwhile, it may be vital for cross validate the anomaly accounts with the unsupervised learning for the datasets for further improvement of supervised learning. Considering the performance, reliability, and benefits of models, the LightGBM algorithms is selected as the base model for the Anomaly Account Detection in the web application.

4.4 Visualization and Application

4.4.1 Visualized Money flow graph

The transaction visualizer is consisting of two parts, the collection of transaction data and visualized using the SigmaJS while layout is constructed by the ForceAtLas2 algorithm. In collection of transaction data, the blockchain explorers can only retrieved the latest 10,000 transaction from the single address. In addition, the visualizer may not help the user to interpret the transaction pattern in the visualized graph when all the transaction data are taken into consideration. Thus, the visualizer retrieves the 2 to 3 layers of the query address with the latest 100 transaction, with the latest 20 unique addresses that transacted with the user's query address.

The features of the transaction visualizer are the dynamic size of the address nodes, the display of associated address with transaction on pointer hover (see **Figure 4.18**), and the details of the specific address on pointer clicks (see **Figure 4.19** and **Figure 4.20**), which also provides external links to blockchain explorers for external informations.

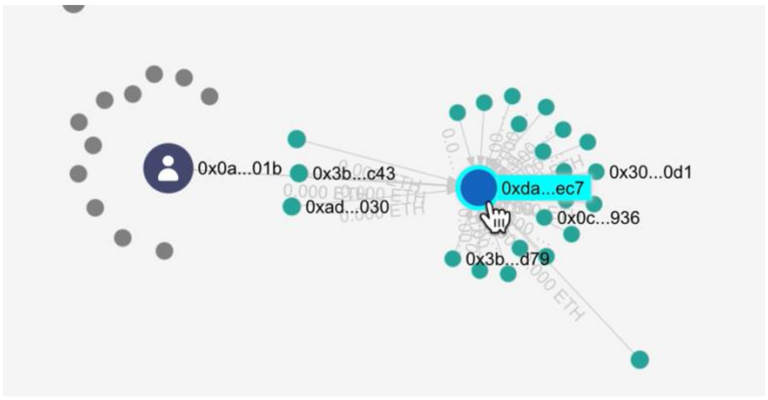
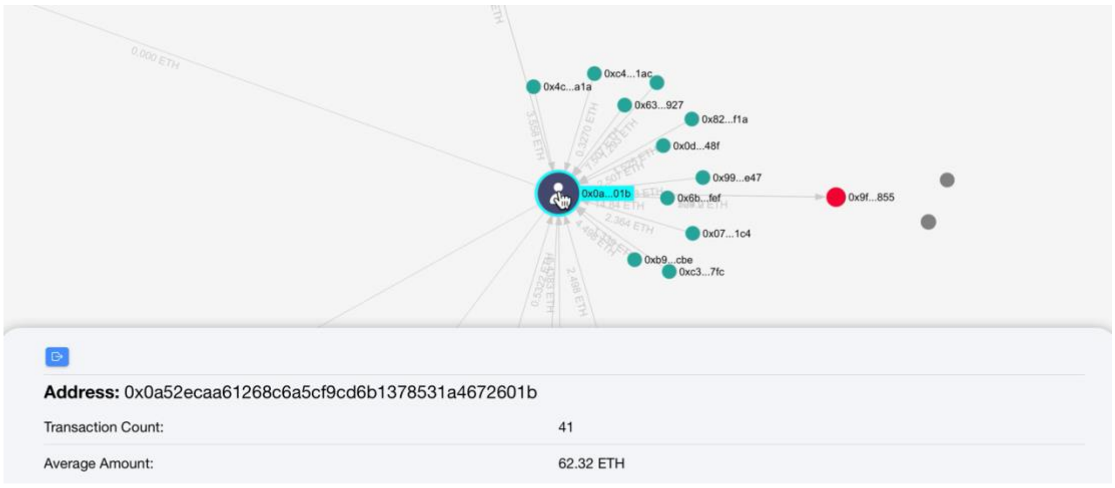


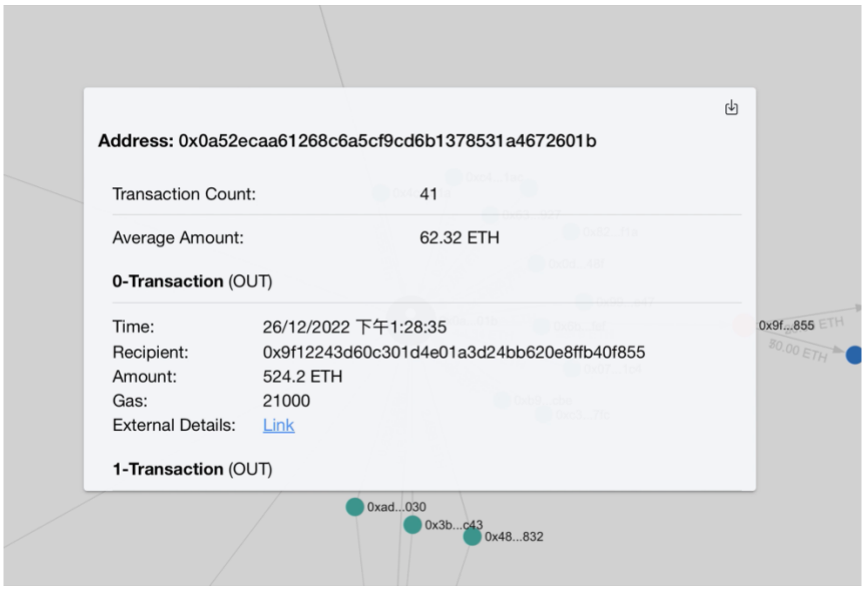
Figure 4.18: The screenshot of the visualized meony flow graph when mouse hover onto the specific address



Address: 0x0a52ecaa61268c6a5cf9cd6b1378531a4672601b

Transaction Count:	41
Average Amount:	62.32 ETH

Figure 4.19: The screenshot of the transaction visualizer on pointer clicks. The bottom panel show the details of the specific clicked address



Address: 0x0a52ecaa61268c6a5cf9cd6b1378531a4672601b

Transaction Count:	41
Average Amount:	62.32 ETH

0-Transaction (OUT)

Time:	26/12/2022 下午1:28:35
Recipient:	0x9f12243d60c301d4e01a3d24bb620e8ffb40f855
Amount:	524.2 ETH
Gas:	21000
External Details:	Link

1-Transaction (OUT)

Figure 4.20: The screenshot of the transaction visualizer on pointer clicks with more details, which showing summary of each transaction and external details that directed to external blockchain explorer

4.4.2 Basic design of the Main Application

The application developed by Ionic and Angular is named as "InnerChain". The application, "InnerChain" is mainly for the blockchain forensics in this project which integrating the anomaly account detection and visualized money flow graph. The main page for user to input the query Ethereum address, and the pages for showing the risk recognition, related analytics, and the money flow graph is shows in **Figure 4.21** to **Figure 4.24** respectively.

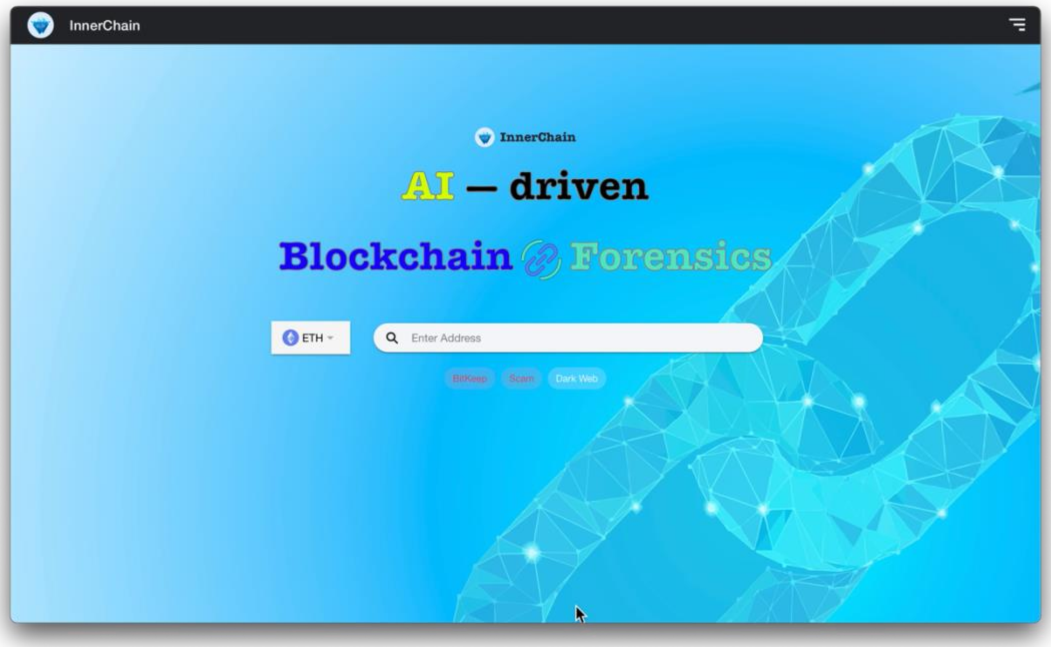
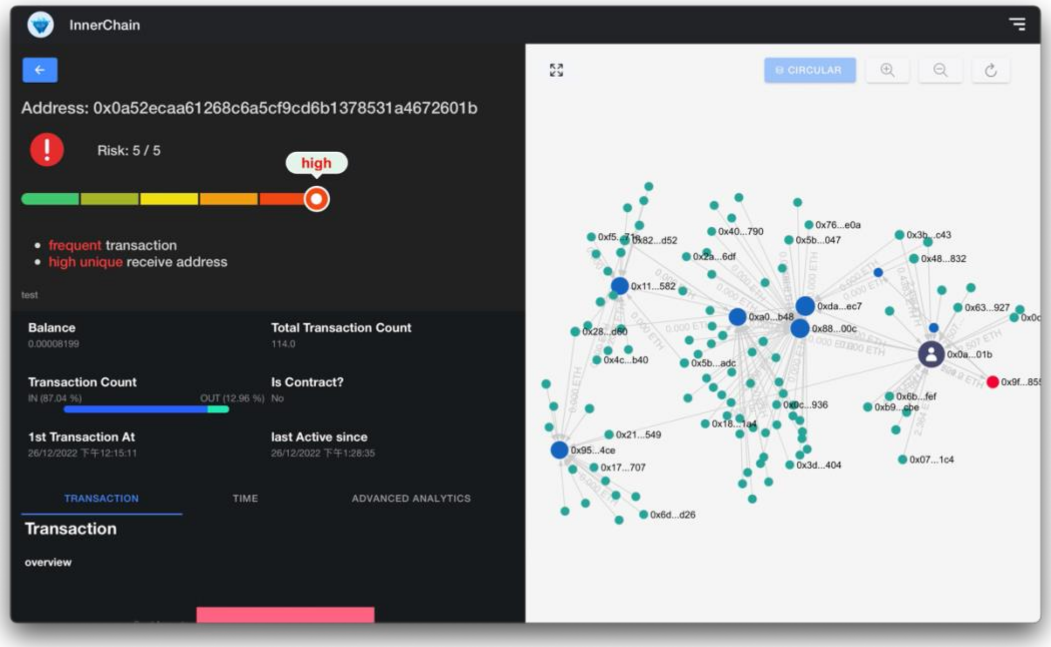


Figure 4.21: The screenshot of the main query page of the application (named "InnerChain"). The screenshot shows the input area for user to enter the Ethereum account address, while the demo anomaly account such as account recognized as scamming or phishing are provided below the input for user

In the results page (in **Figure 4.22**), the risks of the user's input address are mapped into 5 levels, where lowest level 1 indicate the lower risks (where it does not necessary indicate the address is absolutely safe or normal) to highest level 5. The mapping is established on the probability of the anomaly detection models built and utilized in the application. Meanwhile, the application also display the significant features that being the factors of the model decision for the corresponding risk levels, for example, the **Figure 4.23** points out the account is having frequent transaction and high unique receive address which is produced from the model output.

Moreover, the features of the accounts (i.e., the analytics regarding the transaction, time, and other advanced attributes) are displayed on the result page.



Address: 0x0a52ecaa61268c6a5cf9cd6b1378531a4672601b

Risk: 5 / 5

high

- frequent transaction
- high unique receive address

test

Balance
0.00008199

Total Transaction Count
114.0

Transaction Count
IN (87.04 %) OUT (12.96 %)

Is Contract?
No

1st Transaction At
26/12/2022 下午12:15:11

last Active since
26/12/2022 下午1:28:35

TRANSACTION TIME ADVANCED ANALYTICS

Transaction

overview

Figure 4.22: The screenshot of the query result page, the left side of the page showing the risk recognition results with machine learning in the form of risk probability out of 5. The remaining part display the account related statistics and certain graph to show the distribution and differences of transaction related features. Where the right side showing the money flow graph visualized by the application

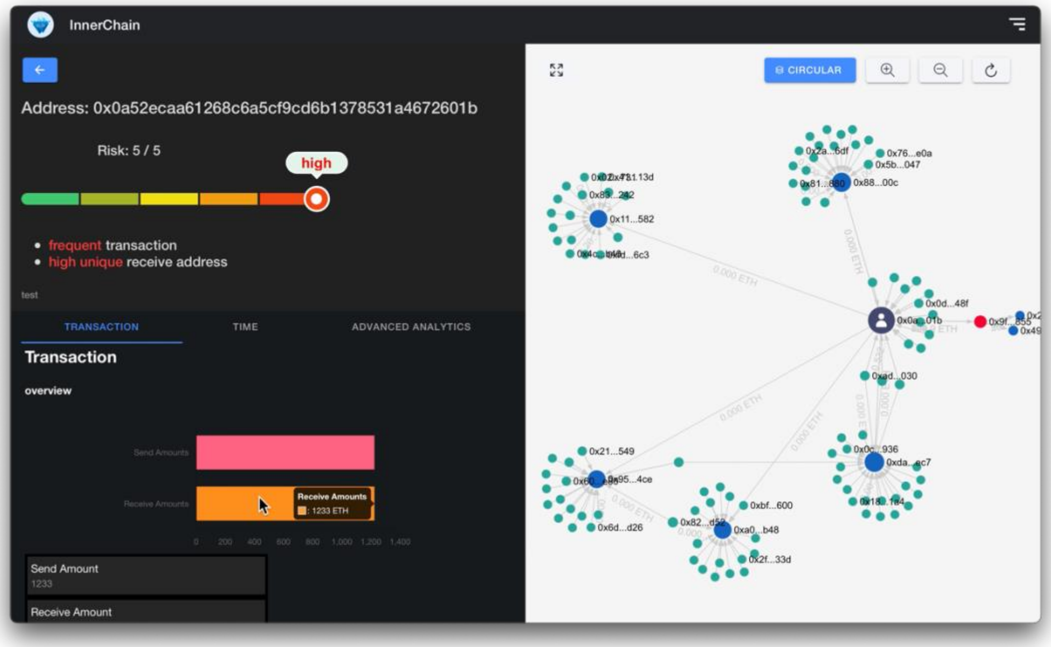


Figure 4.23: The screenshots showing the visualization of data as bar chart and interactive labels with user interaction

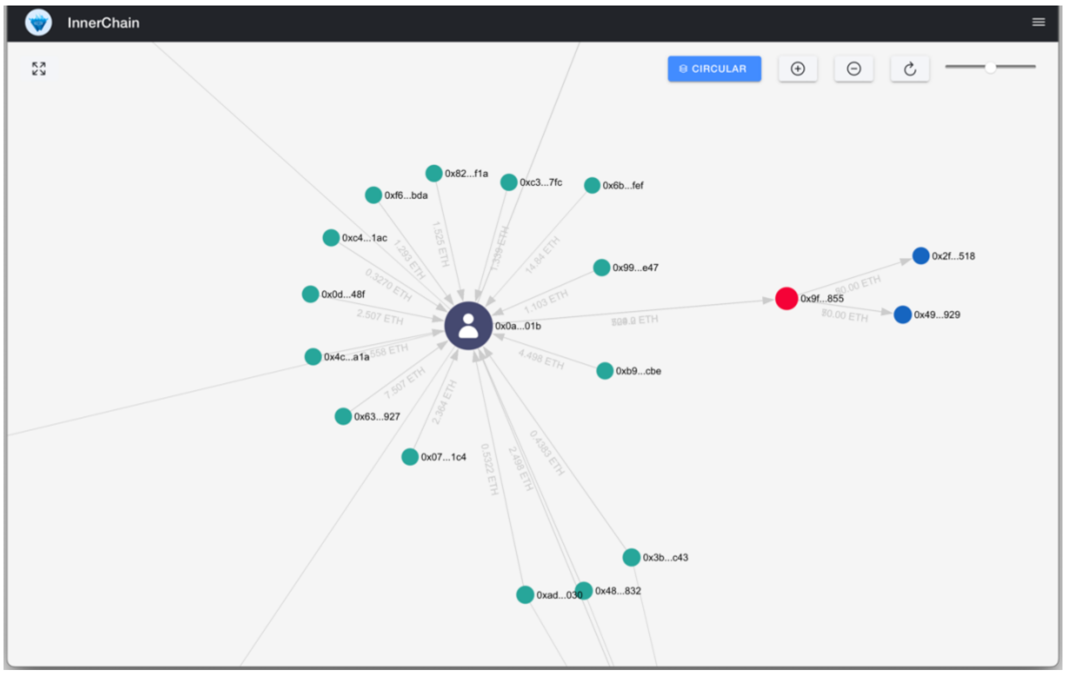


Figure 4.24: The screenshot for the Maximized money flow graph. The account address from user query are colored in black and white icon, the transaction are specified by the directed edges with the different colours of the address nodes

In the application, the user can maximize the panels for money flow graph, in addition, zooming in or out onto the graph to view the details of the relations between different account and respective transaction (with the important features or analytics), (as in **Figure 4.24**). For instance, the black and white nodes with person image icon indicate the input account address from the user, while the other nodes with different colors are connected with the input account with various directed edges.

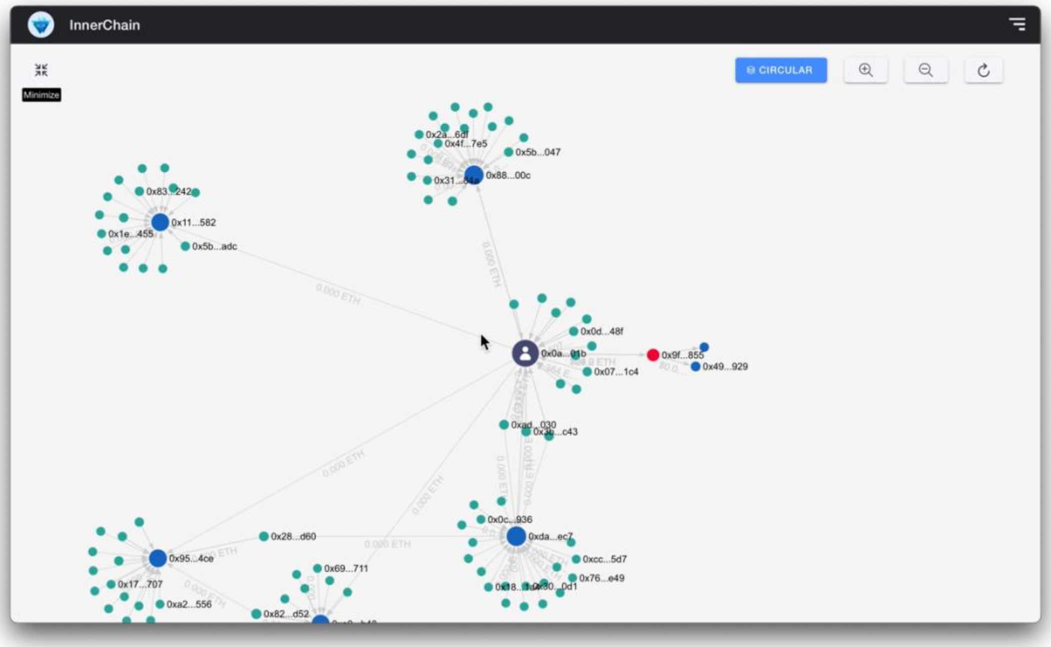


Figure 4.25: The screenshot for the money flow graph in minimized form. It shows the different colors of nodes, while provide layout options for user

5 Future Works

The current progress is on schedule with the completion of all the proposed works and stages, including the anomaly account detection in supervised and unsupervised learning, the transaction visualizer, and the integration into the forensics application.

In current works, the performance of the unsupervised learning is relatively low as compared with the supervised learning methods. It may be related to the data preparation phrase, where the unlabeled account may not be normal while hidden the performance of the unsupervised detection. Hence, for future work, it may be focus on the quality of anomaly detection and integration with the Realtime database. In particular, the application would connect to the Realtime database storing the anomaly account reported by user in different blockchain explorers when user query on specific address. Simultaneously, the report functions could be provided for retraining of the model based on the reasons and of user's comments.

On the other hand, for the theoretical part in machine learning, it is observed the variation between the supervised and supervised methods. Consequently, there may be essential to revise the dataset with in-depth analysis. For example, in terms of collect the data with more layers, (i.e., the addresses transacted with the account in transaction list of the input address). Furthermore, the limitation of the supervised learning should be focused, where they cannot foresee unseen anomaly behaviors when the account behavior changes. While the blockchain may develop and change dynamically, which potentially affect the model's performance. Therefore, more research will be conducted in the future for investigate the possibility of improvement of unsupervised methods. For instance, the unsupervised methods may be utilized to cross validate the anomaly accounts that reported from several websites or blockchain explorers. Additionally, the possibility of model built in this project for another blockchain using account-based model similar to Ethereum should be investigated for wider application of this project.

6 Conclusion

In this report, it has introduced the proposing of the detection of anomalous accounts in Ethereum with the utilization of machine learning techniques, and the visualization of transaction history in order to mitigate and recognize the behavior of the cybercrimes in blockchain. In the sight of increasing occurrences of cybercrimes on the blockchain such as phishing and fake ICO, this project provides the practical application that increase the accessibility for the blockchain forensics which recognizing malicious accounts in the Ethereum blockchain. In addition, visualise the account transaction history would be beneficial to understand pattern and behaviour in a more intuitive way.

The project has investigated and research on the performance and efficiency on the Anomaly Account detection for supervised and unsupervised respectively. The collection of data is primarily in two-part, collection of labelled account, and retrieving of latest transactions from each account. Total of 13941 accounts are from the authoritative open-source database and certain academic resources, where they transaction are collected from the blockchain explorers. Upon the challenges of data imbalance where labelled anomaly account is less than unlabeled account for more than 16%, which cause biased model evaluation scores, the sampling techniques (SMOTE and TomekLinks) are applied. All the models are performed with higher recall and specificity in about 1 to 8%.

In the first stage: the Anomaly Account Detection, for the supervised approaches, it has experimented in the machine learning algorithms: LR, RF, KNN, LR, XGBoost, LightGBM, and stacking of five ML models (RF, KNN, LR, DT, SVM) using the k-fold cross validation in model training. The baseline model has reached the average accuracy of 84.66%, while LightGBM achieved the best performance as about 98% for average accuracy, recall, and specificity with average false positive rate (FPR) of 1.86% and false negative rate (FNR) of 3.78%. In general, the ensemble tree-based algorithms (RF, LightGBM, XGBoost) are having higher performance among the tested algorithms. Although the result is fairly satisfactory comparing to the related studies [3], [9], [15], the amount of test data may not be sufficient to evaluate the performance of the model in the substantial amount of account in the public Ethereum blockchain.

Meanwhile, in tackling the reliability of unlabeled account as normal accounts, it has utilized the supervised learning method. It is mainly in two ways, for clustering and anomaly outlier detection. The former involving of the K-means, HC, DBSCAN, HDBSCAN, and Gaussian Mixture Model (GMM), while the latter contains the Isolation Forest, and OneClassSVM. In overall, the performance of the unsupervised learning is relatively lower than the supervised learning. The dimension reduction strategies are being used, which are the PCA and the t-SNE, which effectively improve the performance of the clustering and anomaly outlier detection.

The best clustering approach, HC have reached the silhouette score of 0.615, where 90% of the datapoints are within single clusters. The OneClassSVM have achieve the specificity of 97.8% and FPR of 2.2% in detecting the outlier, it is mainly composed of the extreme values in the features (i.e., extremely low or high). There are several factors affecting the outcomes of the unsupervised learning which not using the ground-truth labels, for instance, the depth of data features, the quality of the datasets, and the unbalanced datasets.

In the second stage, the transaction visualizer is implemented using the SigmaJS library to draw the money-flow graph on the application and ForceAtLas2 algorithms for the dynamic layout of the graph like the force acting on the different address. While the address is being the nodes, and transaction and important features as directed edges of the graph. In brief, the visualizer provided several features: dedicated color for types of address (i.e., send or receive only), show the associated nodes in transaction for specific account when user hovered over it, and display more transaction details and analytics for the account on click. It aims to provide money flow graph to help user to interpret the hidden pattern of the transaction with the account addresses.

In the last stage, the application combining both the anomaly account detector and the transaction visualizer. The user can input any Ethereum account address to recognize the associated risk and the money flow graph. Furthermore, the underlying APIs for retrieve the account’s transaction data and the models are available in open-source approaches.

For the sake of handling the dynamic behavior of account in blockchain and the rising trend of blockchain-related payment and application, more effort should be taken for the blockchain forensics. For instance, potential of anomaly recognizer model in other blockchain and improvement of unsupervised learning methods. With more research on blockchain-related forensics, it is expected more findings would discover the hidden nature of anomaly behavior in blockchain.

7 Documentation and Source files

The backend APIs hosted on the Microsoft Azure is at https://innerEthereum.azurewebsites.net, owing to limited bandwidth, the first fetching of the services may take around 5 minutes depending on the network stability. On the other hand, the datasets, the models in model training are available on the GitHub webpage https://github.com/Darwin-Chow/fyp23012_

References

[1] U. Hacioglu, *Blockchain Economics and Financial Market Innovation: Financial Innovations in the Digital Age,* Springer Nature, 2019.

[2] Ciphertrace, "Cryptocurrency crime and anti-money laundering," Ciphertrace, Mar 2023. [Online]. Available: https://ciphertrace.com/wp-content/uploads/2023/03/Ciphertrace-CAML-Report-Q3_FINAL.pdf. [Accessed 23 Oct 2023].

[3] H. Han, R. Wang, Y. Chen, K. Xie, and K. Zhang, "Research on abnormal transaction detection method for blockchain," *International Conference on Blockchain and Trustworthy Systems,* pp. 223-236, 2022.

[4] Chainalysis, "The 2023 Crypto Crime Report," Feb 2023. [Online]. Available: https://go.chainalysis.com/rs/503-FAP-074/images/Crypto_Crime_Report_2023.pdf. [Accessed 23 Oct 2023].

[5] N. Tovanich, N. Heulot, J.-D. Fekete and P. Isenberg, "Visualization of Blockchain Data: A Systematic Review," *IEEE Transactions on Visualization and Computer Graphics,* vol. 27, no. 7, p. 3135–3152, 2021.

[6] S. Al-Emari, M. Anbar, Y. K. Sanjalawe and S. Manickam, "A labeled Transactions-Based dataset on the Ethereum network," pp. 61–79. doi: 10.1007/978-981-33-6835-4_5, 2021.

[7] Ethereum.org, "INTRO TO ETHEREUM," 13 Apr 2023. [Online]. Available: https://ethereum.org/en/developers/docs/intro-to-ethereum/. [Accessed 9 Jan 2024].

[8] Ethereum.org, "ETHEREUM ACCOUNTS," 31 Jul 2023. [Online]. Available: https://ethereum.org/en/developers/docs/accounts/. [Accessed 9 Jan 2024].

[9] S. Dolev, V. Kolesnikov, S. Lodha and G. Weiss, "Detecting Malicious Accounts on the Ethereum Blockchain with Supervised Learning," *CSCML,* vol. 12161, p. 94–109, 2020.

[10] R. Agarwal, S. Barve and S. K. Shukla, "Detecting malicious accounts in permissionless blockchains using temporal graph properties," *Applied Network Science,* vol. 6, no. 9, 2021.

[11] Ethereum.org, "PROOF-OF-STAKE (POS)," 26 Sep 2023. [Online]. Available: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/. [Accessed 10 Jan 2024].

[12] Ethereum.org, "GAS AND FEES," 19 July 2023. [Online]. Available: https://ethereum.org/en/developers/docs/gas/. [Accessed 2 Jan 2024].

[13] S. Farrugia, J. Ellul, and G. Azzopardi, "Detection of illicit accounts over the Ethereum blockchain," *Expert Systems With Applications,* vol. 150, p. 113318.

[14] Wu, J. et al., "Who are the phishers? Phishing scam detection on Ethereum via network embedding," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 52, no. 2, p. 1156–1166, Feb 2022.

[15] J. Liu, C. Yin, H. Wang, X. Wu, D. Lan, L. Zhou and C. Ge, "Graph Embedding-Based Money Laundering Detection for Ethereum," *Electronics,* vol. 12, no. 14, pp. 3180-, 2023.

[16] C. WANG, X. DAI, J. XIAO, C. LI, M. WEN, B. ZHOU and H. JIN, "Demystifying Ethereum account diversity: observations, models and analysis," *Frontiers of Computer Science,* vol. 16, no. 4, pp. 164505-, Aug 2022.

[17] T. Chen, Z. Li, Y. Zhu, J. Chen, X. Luo, J. Lui, X. Lin and X. Zhang, "Understanding Ethereum via Graph Analysis," *ACM Transactions on Internet Technolog,* vol. 20, no. 4, pp. 1-32, 2020.

[18] Etherscan, "Etherscan," [Online]. Available: https://etherscan.io. [Accessed 2 Jan 2024].

[19] J. S. Tharani, E. Y. A. Charles, M. P. Z. Hóu and V. Muthukkumarasamy, "Graph Based Visualisation Techniques for Analysis of Blockchain Transactions," *IEEE 46th Conference on Local Computer Networks (LCN),* pp. 427-430, 2021.

[20] D. McGinn, D. Birch, D. Akroyd, M. Molina-Solana, Y. Guo and W. Knottenbelt, "Visualizing Dynamic Bitcoin Transaction Patterns," *Big Data,* vol. 4, pp. 109-119, 2016.

[21] M. Alloghani, D. Al-Jumeily, J. Mustafina, A. Hussain and A. J. Aljaaf, "A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science," *Supervised and Unsupervised Learning for Data Science,* pp. 3-21, 2020.

[22] CryptoScamDB LLC, "CryptoScamDB," [Online]. Available: https://cryptoscamdb.org. [Accessed 8 Nov 2023].

[23] Kaggle, "Ethereum Fraud Detection Dataset," 3 Jan 2021. [Online]. Available: https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset. [Accessed 8 Nov 2023].

[24] F. L. Gewers, G. R. Ferreira, H. F. D. Arruda, F. N. Silva, C. H. Comin, D. R. Amancio and L. D. F. Costa, "Principal Component Analysis: A Natural Approach to Data Exploration," *ACM Computing Surveys,* vol. 54, no. 4, pp. 1-34, 2021.

[25] M. Verleysen and D. François, "The Curse of Dimensionality in Data Mining

and Time Series Prediction," *COMPUTATIONAL INTELLIGENCE AND BIOINSPIRED SYSTEMS, PROCEEDINGS,* vol. 3512, pp. 758-770, 2005.

[26] B. Melit Devassy and S. George, "Dimensionality reduction and visualisation of hyperspectral ink data using t-SNE," *Forensic Science International,* vol. 311, pp. 110194-110194, 2020.

[27] C. Shi, B. Wei, S. Wei, W. Wang, H. Liu and J. Liu, "A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm," *EURASIP Journal on Wireless Communications and Networking,* vol. 2021, no. 31, 2021.

[28] K. R. Shahapure and C. Nicholas, "Cluster Quality Analysis Using Silhouette Score," *IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA),* pp. 747-748, 2020.

[29] F. Ros, R. Riad and S. Guillaume, "PDBI: A partitioning Davies-Bouldin index for clustering evaluation," *Neurocomputing (Amsterdam),* vol. 528, p. 178–199, 2023.

[30] R. Severino, "The Data Visualisation Catalogue," [Online]. Available: https://datavizcatalogue.com/methods/network_diagram.html. [Accessed 23 Oct 2023].

[31] M. Jacomy, T. Venturini, S. Heymann and M. Bastian, "ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software," *PloS One,* vol. 9, no. 6, p. e98679–e98679, 2014.

[32] Sciences-Po médialab and OuestWare, "Sigma.js," [Online]. Available: https://www.sigmajs.org. [Accessed 20 Oct 2023].

[33] M. Bach, A. Werner and M. Palt, "The Proposal of Undersampling Method for Learning from Imbalanced Datasets Ima,∗balanced Datasetas,∗ Małgorzata Bach , Aleksandra Werner , Mateus," *Procedia Computer Science,* vol. 159, pp. 125-134, 2019.

[34] R. M. Aziz, M. F. Baluch, S. Patel, and P. Kumar, "A Machine Learning based Approach to Detect the Ethereum Fraud Transactions with Limited Attributes," *Karbala International Journal of Modern Science,* p. 139–151, May 2022. doi: 10.33640/2405-609x.3229.

[35] M. Christ, N. Braun, J. Neuffer and A. W. E. B. Kempa-Liehr, "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)," *Neurocomputing (Amsterdam),* vol. 307, pp. 72-77, 2018.

[36] M. Butwall, "Data Normalization and Standardization: Impacting Classification

Model Accuracy," *International Journal of Computer Applications,* vol. 183, no. 35, pp. 6-9, Nov 2021.

[37] H. A. Ahmed, P. J. Muhammad Ali, A. K. Faeq and S. M. Abdullah, "An Investigation on Disparity Responds of Machine Learning Algorithms to Data Normalization Method," *ARO-THE SCIENTIFIC JOURNAL OF KOYA UNIVERSITY,* vol. 10, no. 2, pp. 29-37, 2022.

[38] D. Singh and B. Singh, "Investigating the Impact of Data Normalization on Classification Performance," *Applied Soft Computing,* vol. 97, p. 105524, 2020.

[39] Ahsan, M., Mahmud, M., Saha, P., Gupta, K., & Siddique, Z., "Effect of Data Scaling Methods on Machine Learning Algorithms and Model Performance," *Technologies (Basel),* vol. 9, no. 3, pp. 52-, 2021.

[40] R. Silhavy, P. Silhavy and Z. Prokopova, "The Comparison of Machine-Learning Methods XGBoost and LightGBM to Predict Energy Development," *Computational Statistics and Mathematical Modeling Methods in Intelligent Systems,* vol. 1047, p. 208–215, 2019.

[41] Y. Ju, G. Sun, Q. Chen, M. Z. H. Zhang and M. U. Rehman, "A Model Combining Convolutional Neural Network and LightGBM Algorithm for Ultra-Short-Term Wind Power Forecastin," *IEEE Access,* vol. 7, p. 28309–28318, 2019.

[42] BigQuery, Google Cloud, "Ethereum Cryptocurrency," [Online]. Available: https://console.cloud.google.com/marketplace/product/ethereum/crypto-ethereum-blockchain. [Accessed 2 Jan 2024].

[43] Statista Research Department, "Daily active Ethereum (ETH) addresses up until November 9, 2022," 26 Sep 2023. [Online]. Available: https://www.statista.com/statistics/1278174/ethereum-active-addresses/. [Accessed 4 Jan 2024].

[44] S. Miyamoto, "Theory of agglomerative hierarchical clustering," *Springer,* 2022.

[45] S. Dyson, W. J. Buchanan, and L. Bell, "The challenges of investigating cryptocurrencies and blockchain related crime," *The Journal of the British Blockchain Association,* vol. 1, no. 2, pp. 1-6, Dec 2018.

[46] L. Yang, Q. Zhu, J. Huang and D. Cheng, "Adaptive edited natural neighbor algorithm," *Neurocomputing (Amsterdam),* vol. 230, p. 427–433, 2017.

[47] S. Pothuganti, "Review on over-fitting and under-fitting problems in Machine Learning and solutions," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering,* vol. 7, no. 9, pp. 3692-3695, Sep 2018.

[48] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper,* pp. 1-32, 2014.

[49] D. Meng and Y. Li, "An imbalanced learning method by combining SMOTE with Center Offset Factor," *Applied Soft Computing,* vol. 120, pp. 108618-, 2022.

[50] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson and C. E. Leiserson, "Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics," *arXiv (Cornell University),* Jul 2019.