Transaction Graph Analysis for Bitcoin Address Classification: Traditional Supervised Machine Learning and Deep Learning Methods

Seyedarash Saeidimanesh

A Thesis

In the Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of

Master of Applied Science (Information Systems Security)

at Concordia University

Montréal, Québec, Canada

March 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Seyedarash Saeidimanesh**

Entitled: **Transaction Graph Analysis for Bitcoin Address Classification: Traditional Supervised Machine Learning and Deep Learning Methods**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Information Systems Security)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. A. Youssef

_____ Examiner
Dr. J. Clark

_____ Supervisor
Dr. I. Pustogarov

Approved by _____
Dr. J. Yan,
Graduate Program Director

_____
Dr. M. Debbabi,
Dean of the Gina Cody School of Engineering and Computer Science

Abstract

Transaction Graph Analysis for Bitcoin Address Classification: Traditional Supervised Machine Learning and Deep Learning Methods

Seyedarash Saeidimanesh

In this thesis, we consider the problem of Bitcoin address classification and clustering, common in the domains of law enforcement and regulatory compliance. We build a machine learning-based classification framework which is able to attribute a Bitcoin address to one of the predefined classes or to a specific company. We consider five distinct classes for coarse-grained classification: cryptocurrency exchanges, online marketplaces, mining pools, fundraising/charity platforms, and gambling; and 180 companies for fine-grained classification. Classes and the companies were selected so that they represent a broad spectrum of entities and activities within the Bitcoin ecosystem.

This thesis has three main contributions. First, due to the lack of publicly available datasets suitable for testing machine-learning classification algorithms, we create our own labeled dataset consisting of 3M Bitcoin addresses (from 2016-2022), with each Bitcoin address assigned a ready-to-use vector of carefully crafted features. Second, using this dataset, we conduct a comparative analysis of different machine-learning techniques and features for classification. Finally, we develop two types of classifiers: based on the Boosted tree algorithm and the neural network-based classifier. Both are able to attribute a Bitcoin address to one of the predefined classes/companies.

Our binary classification model achieves an F1 score of 76% using the Boosted tree algorithm, while our deep learning model achieves a 90% F1 score for multi-class classification with an accuracy of 92% and 28% higher than related work correspondingly. We achieve 67% accuracy for linking Bitcoin addresses to one of the 180 companies with our deep-learning model.

Acknowledgments

I am profoundly honored to express my heartfelt appreciation to my distinguished supervisor, Dr. Ivan Pustogarov, for his unwavering and invaluable support that has been instrumental in the successful completion of my master's degree. Throughout this academic journey, I have had the privilege of benefiting from Dr. Pustogarov's profound wisdom, profound knowledge, and exceptional mentorship. His remarkable dedication to my growth and development as a researcher and scholar has left an indelible mark on my academic and professional life. I am deeply grateful for his patience, unwavering motivation, and the wealth of expertise he has shared with me. Working under the close guidance of Dr. Pustogarov has been a truly enriching experience. His ability to inspire with innovative ideas, provide constructive comments, offer valuable suggestions, and share profound insights has significantly contributed to the refinement and improvement of this thesis. Dr. Pustogarov's mentorship has been an anchor in navigating the complexities of my research, and I consider myself exceptionally fortunate to have had the opportunity to collaborate with him.

Contents

List of Figures **vii**

List of Tables **ix**

Introduction **1**

1 Background **6**

1.1 Bitcoin 6

1.2 Bitcoin transactions graph behavior 9

1.3 Feature explanation 10

1.3.1 Pagerank 10

1.3.2 Component ID 11

1.3.3 K-core 12

1.4 Classifier methods 14

1.4.1 Support Vector Machine (SVM) classifier 14

1.4.2 Logistic Regression classifier 14

1.4.3 Decision tree classifier 15

1.4.4 Random Forest classifier 15

1.4.5 Boosted Tree classifier 16

1.4.6 Neural deep learning 16

2 Literature review **18**

3 Classification framework **27**

3.1 Creating dataset 27

3.1.1 Obtaining Bitcoin addresses 27

3.1.2 Enriching blockchain data 28

3.1.3 Calculating Bitcoin address features 28

3.2 Analysis existing methods 32

3.3 Our classifier and evaluation 33

3.4 Deep learning model 34

3.4.1 Tracking dark web UnlockDevices marketplace transaction 37

4 Implementation **40**

4.1 Block parser 40

4.2 Memory efficiency methods 41

4.3 Implementation steps 43

4.4 Project main challenges 46

4.5 Conclusion and Future work 47

Appendix A Our experiment results **48**

Bibliography **55**

List of Figures

Figure 1.1 Blockchain Structure[6] 7

Figure 1.2 Transaction propagation [4] 7

Figure 1.3 Transaction content [5] 8

Figure 1.4 Common and uncommon Forks [4] 9

Figure 1.5 Bitcoin address relationships look like graph behavior [7][8] 10

Figure 1.6 Illustration of the k-core decomposition[6] 13

Figure 1.7 Support Vector Machine classifier with a non-linear kernel 14

Figure 1.8 Logistic Regression Classifier. 15

Figure 1.9 The architecture of Gradient Boosting decision tree 16

Figure 1.10 Neural Network and Deep Learning Neural Network 17

Figure 2.1 Illicit F1-score obtained with Random Forest, for the standard 166 features and the new GuiltyWalker features before and after feature selection [13] 18

Figure 2.2 Building blocks of BitIodine [44] 22

Figure 2.3 Heuristics results in paper [34] 23

Figure 2.4 Common input ownership schematic diagram H1 [34] 23

Figure 3.1 PageRank formula 31

Figure 3.2 Repeated result illicit F1-score [31] 33

Figure 3.3 Shows the confusion matrix for multiclassification with neural deep learning method 36

Figure 3.4 Coefficients of each feature. 37

Figure 3.5 Tracking Bitcoin address in Dark web 38

Figure 3.6 Tracking Bitcoin address in Dark web 39
Figure 4.1 Implementation Steps 43
Figure 4.2 Neo4j output for Bitcoin transactions 44
Figure 4.3 Memory visualization interconnections 47

List of Tables

Table 2.1 Elliptic dataset. 19

Table 2.2 Evaluation metrics binary classification with SVM classifier [46] 22

Table 2.3 Confusion matrix binary classification with SVM classifier [46] 22

Table 2.4 Heuristic name in paper [34]. 23

Table 3.1 Bitcoin address feature calculation 29

Table 3.2 All the features that we calculate 30

Table 3.3 Comparison between different machine learning methods 33

Table 3.4 Evaluation metrics binary and multi-classification with boosted tree classifier in comparison with [46]. 35

Table 3.5 Confusion matrix binary and multi-classification with boosted tree classifier in comparison with [46]. 35

Table 3.6 Dataset that used with Boosted Tree classifier. 35

Table 3.7 Confusion matrix for Multi-classification with Boosted Tree Classifier 36

Table 4.1 Bitcoingraph block parser outputs 41

Table 4.2 TuriCreate common ML tasks 41

Table 4.3 Rusty block parser outputs 45

Table A.1 Fraudulent Bitcoin address classification results 54

Introduction

Bitcoin transaction clustering has gained significant attention due to its anti-money laundering (AML) and economic purposes. Clustering Bitcoin transactions into distinct classes aids in tracking stolen Bitcoins. Classification of Bitcoin transactions is essential for understanding its economic impact and identifying potential risks associated with its use. By categorizing transactions into different classes, researchers can discern patterns and behaviors, enabling them to develop more effective anti-money laundering measures.

Bitcoin's pseudonymous nature raises concerns about its potential for illicit activities, including money laundering and terrorist financing. Classification of Bitcoin transactions facilitates the identification and tracking of suspicious or high-risk transactions. By applying AML algorithms and techniques, regulatory authorities and financial institutions can detect and prevent illicit activities, ensuring compliance with AML regulations. Classification assists in identifying fraudulent activities within the Bitcoin ecosystem. By analyzing transaction patterns and characteristics, machine learning algorithms can detect anomalies and flag potentially fraudulent transactions. This aids in preventing scams, Ponzi schemes, and other fraudulent schemes that exploit Bitcoin transactions.

Most Bitcoin clustering studies relied on common-input ownership and change address heuristics for clustering. Previous studies provided numerical results in terms of the number of clusters or the number of addresses within a class. Previous studies focused on licit and illicit Bitcoin address clustering. However, we developed a new method to classify Bitcoin addresses into binary and multi-class categories and linked the Bitcoin addresses to their real-world owners. We achieved good statistical results in the classification evaluation metrics.

Several limitations were identified in the previous studies, in addition, most of these studies

relied on closed-source datasets, such as the elliptic dataset. Previous studies often focused on old Bitcoin transactions, which can not reflect recent Bitcoin network behavior. One important gap in the previous studies was that they focused only on clustering techniques for small groups of Bitcoin addresses. These clustering methods were typically unsupervised and lacked the ability to provide meaningful statistical information about the Bitcoin addresses clustering.

However, our research classifies Bitcoin addresses into binary and multi-class categories and links Bitcoin addresses to their real-world owners. Our research provides better statistical and evaluation metric results than the previous works. In this study, five distinct classes are considered for classification: Crypto-currency exchanges, Online marketplaces, Mining pools, and Fundraising/charity platforms. These classes represent various types of entities and activities associated with Bitcoin addresses and transactions, allowing for a comprehensive understanding of the Bitcoin ecosystem. Our research involves feature engineering to identify important features with high coefficients, leading to significant advancements in both binary and multi-class classification. This method enables the precise identification of Bitcoin addresses associated with various categories, including Finance, Service, Gambling, Miscellaneous, and Pools. In addition, we have successfully linked Bitcoin addresses to their real-world owners through deep-learning techniques.

The proposed approach has three main contributions, first, we created our dataset. To create our dataset we collected a large number of recent bitcoin addresses and extracted different properties for them, and we labeled these addresses into different classes by using available data from various sources such as wallet explorer[1] and bitcoinwhoswho[2]. In particular, this dataset includes a collection of recent Bitcoin addresses, totaling 3M entries. This large dataset is created to address the shortcomings of conventional datasets, which were often considered closed sources or outdated. Second, we conducted a comparative analysis of different machine-learning techniques on our dataset to determine the most suitable one for the task, and also we did feature engineering to identify important features with high coefficients. Third we classified Bitcoin addresses into binary and multi-class categories and linked Bitcoin addresses to their real-world owners.

These methodological strategies, when combined, form the foundation of our research, which provides a framework for our findings and insights into Bitcoin address clustering and classification. Through this comprehensive approach, we aim to classify Bitcoin addresses into binary and

multiclass categories and link Bitcoin addresses to their real-world owners.

In summary, the proposed model has some features including a mixture of big data and machine learning. It employs a two-step clustering approach based on common-input-ownership heuristics to group data into distinct entities. The model creates a large dataset using memory-efficient methods with recent data. It involves feature engineering to identify the important features with high coefficients that consider Bitcoin graph behavior. The model conducts a comparative analysis of different machine-learning techniques on our dataset to determine the most suitable one for the task. Notably, the proposed model supports supervised machine learning classification, encompassing all Bitcoin addresses, rather than being limited to specific groups, such as licit and illicit purposes. The proposed model includes both binary and multi-classification of Bitcoin addresses and also employs deep learning techniques to link Bitcoin addresses to their real-world owners, the results achieved by this model demonstrate better results in accuracy and other performance metrics compared to prior research.

Application domains for Bitcoin address classification

The classification of Bitcoin transactions holds significant importance across various domains, encompassing finance, law enforcement, and regulatory compliance. The following key reasons underscore the significance of Bitcoin transaction classification.

Economic analysis, categorizing Bitcoin transactions provides valuable insights into fund flows within the Bitcoin network. By classifying transactions based on their purpose or nature, economists can analyze patterns, identify market behaviors, and gain a deeper understanding of Bitcoin's overall economic activity. This information aids in making informed investment decisions, tracking market sentiment, and predicting market movements.

Anti-Money Laundering (AML), Bitcoin transaction classification is useful for Anti-Money Laundering (AML) purposes. Bitcoin's pseudonymous nature raises concerns about its potential for illicit activities, including money laundering and terrorist financing. Classification of Bitcoin transactions facilitates the identification and tracking of suspicious or high-risk transactions. By applying AML algorithms and techniques, regulatory authorities and financial institutions can detect

and prevent illicit activities, ensuring compliance with AML regulations.

Fraud detection, classification of Bitcoin transactions can help with fraud detection. Classification assists in identifying fraudulent activities within the Bitcoin ecosystem. By analyzing transaction patterns and characteristics, machine learning algorithms can detect anomalies and flag potentially fraudulent transactions. This aids in preventing scams, Ponzi schemes, and other fraudulent schemes that exploit Bitcoin transactions.

Customer profiling, and classifying Bitcoin transactions enables the creation of customer profiles and segmentation based on transaction behavior. This profiling helps understand the preferences, spending habits, and transaction patterns of different user groups. Financial institutions and businesses can leverage this information to provide personalized services, targeted marketing, and enhanced customer experiences.

Risk management, classifying Bitcoin transactions helps assess and manage risks associated with Bitcoin-related activities. By categorizing transactions based on risk factors such as the source of funds, transaction size, or destination, risk assessment models can be developed to evaluate the level of risk associated with specific transactions or entities. This enables proactive risk management and mitigation strategies.

Regulatory compliance, governments and regulatory bodies are increasingly focused on regulating cryptocurrencies and ensuring compliance with existing financial regulations. Classifying Bitcoin transactions facilitates compliance monitoring and reporting. By identifying transactions falling under regulatory frameworks (e.g., cross-border transactions, large-value transactions), regulators can enforce compliance and detect potential instances of tax evasion or financial crimes.

Therefore, the classification of Bitcoin transactions plays an important role in enhancing the understanding of this decentralized financial system, promoting transparency, and facilitating the responsible and secure use of cryptocurrencies.

Outline

The thesis is structured as follows, in chapter 1 we discuss about background. This section reviews and discusses how Bitcoin works and explains some of the Bitcoin graph behavior features.

Chapter 2 is about the literature review. In this chapter, we present previous research and studies related to Bitcoin clustering and transaction analysis. We also highlight the advancements and limitations in this field. We present our proposed approach and methodology for the classification of Bitcoin addresses in chapter 3. We employ binary, multi-class classification, and denomination classification. In this chapter deep learning techniques and boosted tree classifiers are utilized. We discuss the findings of our research and compare them with existing studies in the domain of Bitcoin transaction classification. We also present the effectiveness of our approach and its contributions to the Bitcoin address classification. Chapter 4 is about implementation, and presents some technical details that we follow to design our dataset and model.

Chapter 1

Background

1.1 Bitcoin

Bitcoin was introduced by an individual or group using the pseudonym Satoshi Nakamoto in 2009 [3] that operates on a blockchain. The blockchain provides Bitcoin's public ledger. A public ledger is, an ordered and timestamped record of transactions. It helps to prevent cheating by preventing double spending and modification of previous transaction records. In the Bitcoin network, every full node stores a blockchain which includes only the blocks that it has verified. When multiple nodes have identical blocks in their blockchain, they are considered to be in consensus. Consensus rules are the validation rules these nodes follow to maintain consensus [4]. The Bitcoin consensus system forms an unalterable ledger where entries can only be added, it is the append-only ledger. Miners, who validate transactions ensure that transactions are valid [5].

A group of new transactions is gathered in the transaction data section of a block. Each transaction is duplicated and hashed, and the hashes are then paired, and the process repeats until a single hash remains, it is called the Merkle root of a Merkle tree. The block's header holds the Merkle root. Each block includes the hash of the previous block's header, creating a chain of blocks. This ensures that altering a transaction requires modifying not just its block but also all the following blocks. Transactions are linked and create a chain together, and Bitcoins move from transaction to transaction. In each transaction, the satoshis that previously received from earlier transactions will be spent. The input of one transaction is the output of a previous transaction. A single transaction

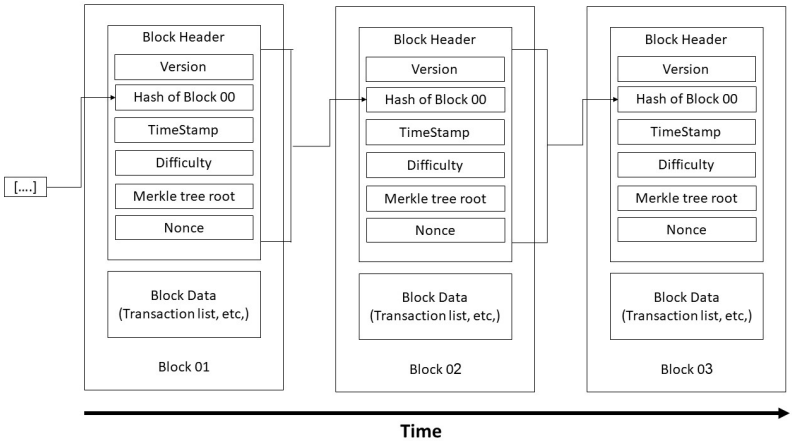

Figure 1.1: Blockchain Structure[6]

can have multiple outputs, this happens when sending to multiple addresses. However, each output of a particular transaction can only serve as an input once in the blockchain. This procedure prevents double spending in the Bitcoin blockchain. As each output from a particular transaction can only be used once in the blockchain; therefore, all transactions in the blockchain are categorized as either Unspent Transaction Outputs (UTXOs) or spent transaction outputs. [4].

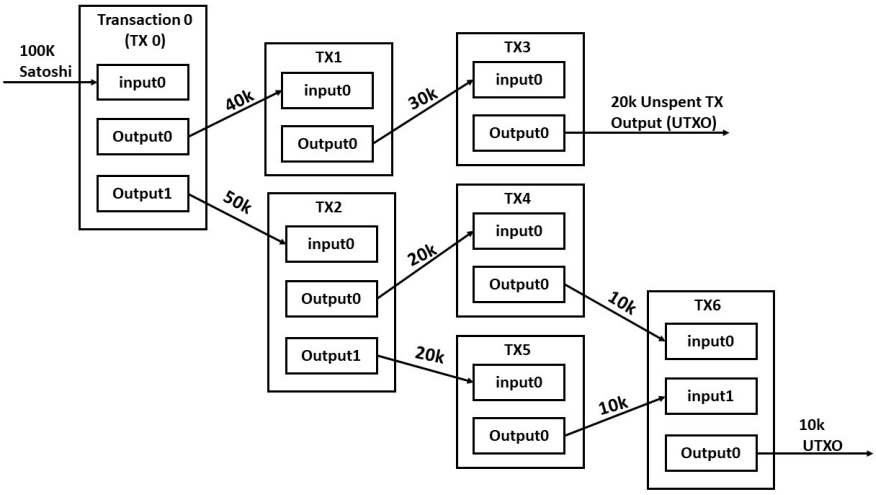

Figure 1.2: Transaction propagation [4]

Transaction has three parts, metadata, inputs, and outputs. Metadata includes the size of the transaction, the number of inputs, the number of outputs, and the hash of the entire transaction. The transaction inputs are organized into an array, and each input follows a consistent format. An input

includes information about a preceding transaction, such as its hash, serving as a hash pointer to the previous transaction. Additionally, the input specifies the index of the previous transaction's outputs that are being claimed. The transaction outputs form an array, each output has two fields that each has a value. The total of all output values must not exceed the total of all input values. If the output values' sum is less than the input values' sum, the difference represents a transaction fee awarded to the miner who validates and publishes the transaction [5].

Figure 1.3: Transaction content [5]

The private key is a secret, confidential, and long string of characters that is known only to the owner of a Bitcoin wallet, it serves as the digital signature of ownership and control over the associated Bitcoin. The private key is used to create digital signatures for Bitcoin transactions. It is used to prove that the person initiating the transaction has the right to do so. The public key is derived from the private key using a mathematical algorithm. It is a shorter string of characters, and it is publicly known and associated with the owner's Bitcoin address. The public key is used to verify the digital signatures created with the private key. It confirms that the transaction has been authorized by the private key holder.

Many anonymous peers work together on the network to maintain the blockchain. Some untrustworthy peers may want to modify past blocks, to prevent this each block needs to prove a significant amount of work was invested in its creation. Untrustworthy peers seeking to change previous blocks must exert more effort than honest peers who simply aim to add new blocks to the

blockchain [4]. The connection between blocks in the blockchain ensures that modifying any transaction in the block needs to change all the blocks that come after it in the sequence. So, changing a specific block increases the cost with every new block added to the blockchain, this shows the importance of the proof of work in the Bitcoin blockchain. Bitcoin's proof of work relies on the unpredictability of cryptographic hashes. It means if any person changes the data and recalculates the hash, he gets a new random number. Therefore, it is impossible to modify the data in a way that makes the hash number predictable.

Bitcoin miners compete to add new blocks to the blockchain by solving a complex math puzzle. If a miner succeeds, they can add their block to the blockchain, and it's identified by its block height. Sometimes, multiple miners create blocks at the same time, leading to a fork in the blockchain. When this happens, each participant on the network decides which block to accept. They usually pick the first one they see. Eventually, a miner produces another block that attaches to only one of the competing simultaneously-mined blocks, this makes that side of the fork stronger than the other side. Since multiple blocks can have the same height during a blockchain fork, block height should not be used as a globally unique identifier. Instead, blocks are usually referenced by the hash of their header.

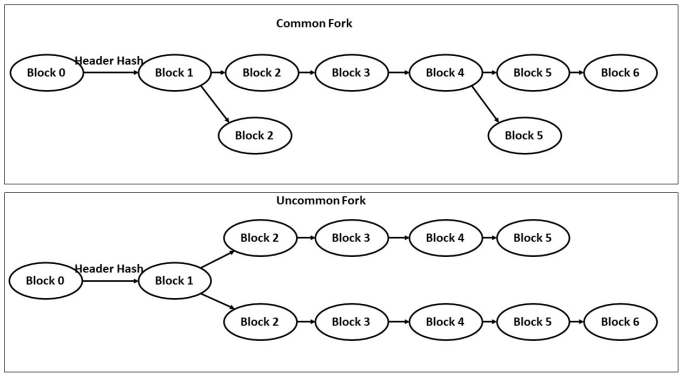

Figure 1.4: Common and uncommon Forks [4]

1.2 Bitcoin transactions graph behavior

Bitcoin transactions form a graph-like structure due to their interconnectivity and dependency on previous transactions. Each transaction contains information about the previous transaction it

refers to, identified by the ”prev hash” information, thereby creating a chain-like linkage between transactions. This chain of transactions is commonly referred to as the blockchain. As new transactions are added to the blockchain, they reference the outputs of previous transactions as their inputs, effectively forming a directed acyclic graph (DAG) known as the transaction graph. This graph's structure captures the flow of funds through the Bitcoin network, as the outputs of one transaction become the inputs of subsequent transactions. Transaction graph enables the verification of the entire transaction history, ensuring that each Bitcoin being spent originates from legitimate sources and has not been double-spent. This inherent transparency and immutability make Bitcoin's transaction graph an essential feature of its decentralized and secure nature.

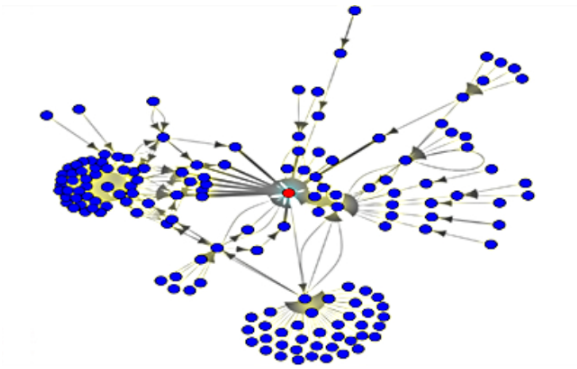

Figure 1.5: Bitcoin address relationships look like graph behavior [7][8]

1.3 Feature explanation

Bitcoin address graphical features such as PageRank, k core, and component ID are discussed in the following.

1.3.1 Pagerank

PageRank is an algorithm used to measure the importance or influence of nodes (web pages) in a directed graph, typically represented as a web graph. It was developed by Larry Page and Sergey Brin, the co-founders of Google, and is one of the foundational algorithms that fueled the success of Google's search engine. In a web graph, nodes represent web pages, and directed edges represent hyperlinks between those pages. The PageRank algorithm assigns a numerical score to each page,

which reflects the importance or popularity of the page within the entire web graph. The underlying idea is that a page is more important if it is linked to by other important pages. The original concept of PageRank can be understood through the following simplified explanation, each page in the web graph is initially assigned an equal score (e.g., 1.0). During each iteration of the algorithm, the score of each page is updated based on the scores of the pages linking to it. Pages with higher incoming scores contribute more to the score of the current page.

The process of updating the scores continues for a certain number of iterations or until convergence is reached. The final scores represent the PageRank of each page, with higher scores indicating higher importance or influence. PageRank is used by search engines to rank web pages in their search results. Pages with higher PageRank scores are considered more relevant or authoritative and are likely to appear higher in search results when relevant queries are made by users.

Although PageRank was initially developed for web graphs, the underlying concept of measuring the importance of nodes in a network has been extended to various other applications, such as social network analysis, recommendation systems, and citation networks.

1.3.2 Component ID

In the context of graphs, a ”component ID” refers to a unique identifier assigned to each connected component in the graph. A connected component is a subgraph in which every node is connected to at least one other node through a series of edges. In other words, all the nodes within a connected component can be reached from any other node in the same component by traversing the edges of the graph. An undirected graph consists of a set of nodes (vertices) connected by edges, where the edges have no direction. A connected component in an undirected graph is a subset of nodes where there is a path between any two nodes in the component. However, there may not be a path between nodes in different components.

When a graph contains multiple connected components, each component is assigned a unique component ID. The component IDs serve as a way to distinguish and group nodes belonging to the same connected component. For example, if a graph has three connected components, they might be labeled with component IDs 1, 2, and 3, respectively. Graph algorithms often use component IDs to perform various operations on connected components, such as finding all components, determining

the size of each component, identifying if two nodes are in the same component, and so on. Component IDs are particularly relevant in applications like network analysis, social network analysis, and community detection, where identifying and analyzing connected components can reveal important structural properties and patterns within the graph.

It is important to note that the specific representation and calculation of component IDs may vary depending on the graph data structure and the algorithms used for graph analysis. In some cases, component IDs may be integers, while in other cases, they could be strings or other data types, depending on the requirements of the analysis or the software library used for graph manipulation[9].

1.3.3 K-core

In graph theory, a k-core is a subgraph of a graph in which every node has a degree of at least k. In other words, a k-core is a maximal connected subgraph where each node has at least k neighbors within the subgraph. To form a k-core, the process involves iteratively removing nodes with degrees less than k until no more such nodes can be removed. The resulting subgraph is the k-core of the original graph. The k-core concept is commonly used in graph analysis and has various applications, including: Identifying central and highly connected regions.

Higher k-cores represent regions with denser connectivity and can be seen as the ”core” of highly connected nodes in the graph. K-cores can be used as a preprocessing step in community detection algorithms to find dense substructures in the graph. Displaying the k-core subgraphs can help in visualizing the most connected parts of the network. The k-core decomposition can be used to simplify the graph by removing less important nodes, and reducing its size while preserving its core structure.

The k-core decomposition is a useful tool to understand the hierarchical organization of networks and to reveal important substructures within them. It allows researchers to analyze and interpret complex networks in a more manageable and interpretable way. The k-core concept is closely related to the concept of ”degree centrality,” which measures the number of edges connected to a node. However, k-core goes beyond just the degree and provides a more nuanced view of node connectivity in the graph [10].

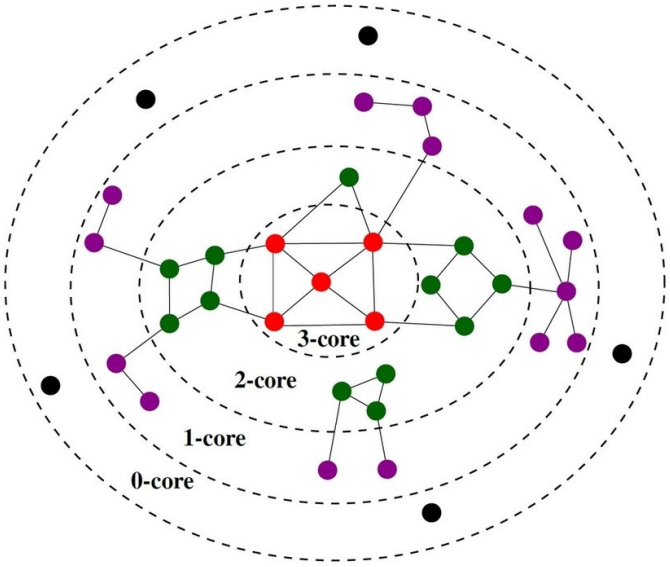

Figure 1.6: Illustration of the k-core decomposition[6]

In summary, Pagerank is a feature that measures the importance or influence of a node in a network or graph. It assigns a numerical value to each node based on the quantity and quality of incoming links. Nodes with higher Pagerank scores are considered more significant or central within the network. The algorithm takes into account both the number and quality of incoming links, giving more weight to links from highly ranked nodes. Pagerank is commonly used in fields like social network analysis and information retrieval to identify important nodes or entities in a network.

In graph theory, the component ID feature refers to a unique identifier assigned to each connected component within a graph. A connected component is a subgraph where every node is connected to at least one other node in the same subgraph. The component ID serves as a label or tag that distinguishes one connected component from another. By assigning a component ID to each connected component, we can identify and analyze the different clusters or groups of nodes within a graph. This feature helps in understanding the connectivity patterns, identifying isolated or disconnected nodes, and studying the overall structure and organization of the graph. Component ID can be useful in various graph-related tasks, such as community detection, network visualization, and analyzing the spread of information or influence within a network.

1.4 Classifier methods

We use different machine-learning methods and compare them with each other to find the best one.

1.4.1 Support Vector Machine (SVM) classifier

Support Vector Machine is a powerful and widely used supervised learning algorithm for classification and regression tasks. In the context of classification, it finds the optimal hyperplane that best separates different classes in the feature space. The main idea is to maximize the margin between the two classes, making SVM effective in dealing with both linearly separable and non-linearly separable datasets. The SVM classifier works by mapping input data points into a higher-dimensional feature space and then finding the optimal hyperplane that separates the data points. It can handle binary classification tasks as well as be extended to multi-class classification using techniques like one-vs-one or one-vs-all. SVM is known for its ability to handle high-dimensional data and handle overfitting well.

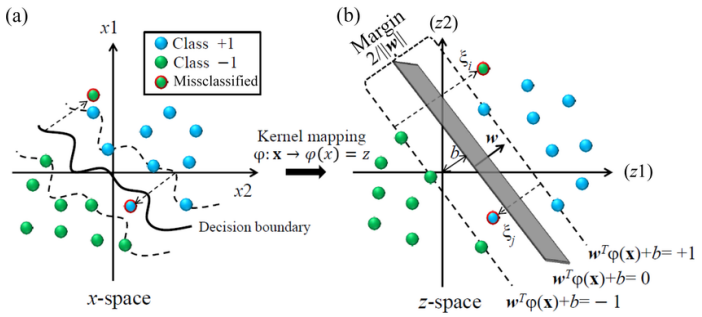

Figure 1.7: Support Vector Machine classifier with a non-linear kernel

1.4.2 Logistic Regression classifier

Logistic Regression is a linear classification algorithm used for classification tasks. Despite its name, it is primarily used for classification, not regression. It predicts the probability of an instance belonging to a particular class by applying the logistic function (sigmoid) to a linear combination of input features. In logistic regression, the model estimates the probability of an instance belonging to the positive class (e.g., class 1) given its features. If the probability is above a specified threshold

(usually 0.5), the instance is classified as positive; otherwise, it is classified as negative (e.g., class 0). Logistic regression is simple, interpretable, and efficient, but it may not perform well on complex, non-linear datasets.

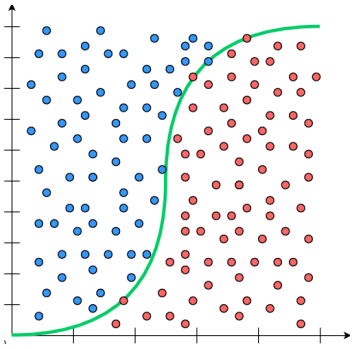

Figure 1.8: Logistic Regression Classifier.

1.4.3 Decision tree classifier

A decision tree is a non-linear and hierarchical tree-like structure used for both classification and regression tasks. In the context of classification, the tree is built by recursively splitting the data into subsets based on the features, aiming to create pure leaf nodes (each containing instances of a single class). At each node, the decision tree algorithm selects the feature that best separates the data based on some impurity measure (e.g., Gini impurity or entropy). The process continues until a stopping criterion is met, such as a maximum tree depth or the minimum number of samples in a leaf node. Decision trees are easy to interpret and visualize, but they can be prone to overfitting, especially when the tree becomes deep.

1.4.4 Random Forest classifier

Random Forest is an ensemble learning method that builds multiple decision trees during training and combines their predictions to make a final classification decision. Each decision tree is constructed using a random subset of the training data and a random subset of the features. This randomness helps reduce overfitting and improve generalization. In the classification task, the final prediction is made by taking a majority vote among the predictions of individual decision trees. Random Forest is more robust than a single decision tree and performs well on a wide range of

tasks. It is known for its ability to handle high-dimensional data and avoid overfitting.

1.4.5 Boosted Tree classifier

Boosted Tree is another ensemble learning method that combines weak learners (typically shallow decision trees) into a strong predictive model. The algorithm works in an iterative manner, with each new tree correcting the errors of the previous ones. During training, the algorithm assigns higher weights to misclassified instances, which means the subsequent trees focus more on these misclassified instances. The final prediction is obtained by aggregating the predictions of all the weak learners. Boosted Trees are powerful and often provide superior accuracy compared to individual decision trees or Random Forests, but they may be more prone to overfitting and can be computationally expensive.

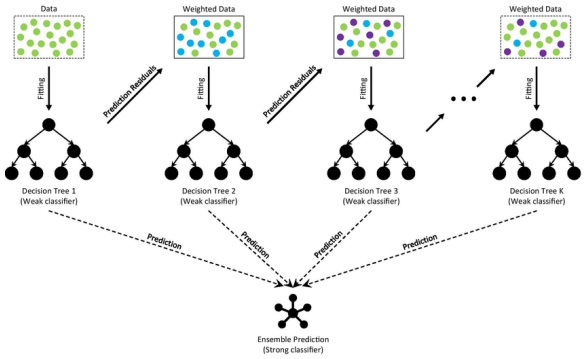

Figure 1.9: The architecture of Gradient Boosting decision tree

1.4.6 Neural deep learning

Often referred to simply as deep learning, is a subfield of artificial intelligence (AI) and machine learning that focuses on training artificial neural networks to perform complex tasks. It is inspired by the structure and function of the human brain's neural networks, where neurons process and transmit information to make decisions and perform various functions. At the core of deep learning are Artificial Neural Networks (ANNs), which are computational models consisting of interconnected layers of nodes called neurons. These neurons are organized in layers: an input layer, one or more hidden layers, and an output layer. Each neuron in a layer receives input data, processes it using mathematical operations, and passes the output to the next layer. The term "deep" in deep learning

refers to the use of multiple hidden layers in neural networks. The depth of a neural network allows it to learn and represent hierarchical patterns and features in the data, making it capable of handling more intricate and abstract relationships in the input data. The process of deep learning involves two fundamental phases: training and inference[11].

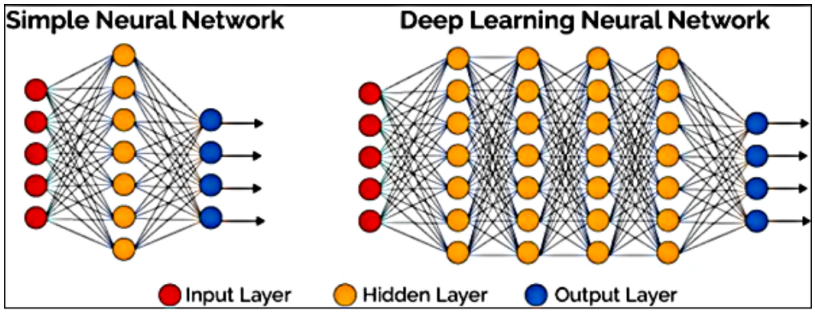

Figure 1.10: Neural Network and Deep Learning Neural Network

Chapter 2

Literature review

In this chapter, some important research and papers that work on Bitcoin address classification and tracking will be discussed.

There are several research in the field of fraud detection and clustering Bitcoin addresses to distinguish between illicit and licit activities. Some clustering methods for fraud detection rely on elliptic dataset, as mentioned in [12]. Catarina [13] introduced a new approach called guilty walker methods, which utilize graph structure and past labels to enhance the performance of machine learning methods for money laundering detection. Additionally, a paper by Joana [14] presented an active learning solution for detecting money laundering in the Bitcoin blockchain when labeled data is scarce.

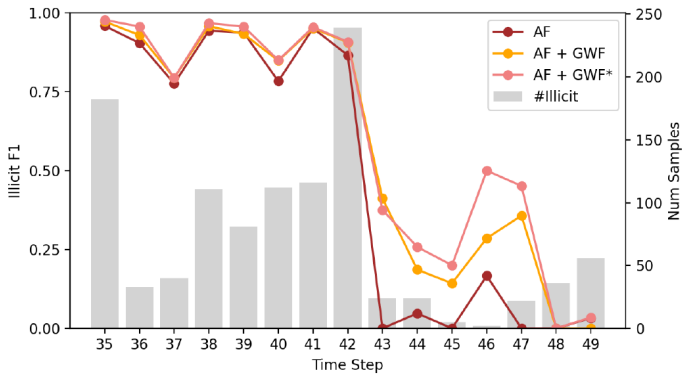

Figure 2.1: Illicit F1-score obtained with Random Forest, for the standard 166 features and the new GuiltyWalker features before and after feature selection [13]

Most Bitcoin fraud detection research uses the Elliptic dataset [15, 16, 17, 18, 19, 14, 20, 21,

22, 23, 24, 25, 26] and [27, 28, 18, 29, 13, 30, 31, 32]. This is an anonymized dataset that due to intellectual property issues, did not provide an exact description of all the features in the dataset. There is a time step associated with each node, representing a measure of the time when a transaction was broadcasted to the Bitcoin network. The time steps, running from 1 to 49, are evenly spaced with an interval of about two weeks. Each time step contains a single connected component of transactions that appeared on the blockchain within less than three hours between each other; there are no edges connecting the different time steps.

The dataset includes 203,769 node transactions and 234,355 directed edges, representing the flow of Bitcoin going from one transaction to the next. From the total number of transactions, 21% (42,019) are labeled as licit, 2% (4,545) as illicit, and the remaining 77% (157,205) are unknown. Besides the graph structure, the dataset has 166 anonymized features associated with each transaction. The first 94 relate to information about the transaction itself. Table 2.1 shows the Elliptic dataset contents. More than anonymized features the Elliptic dataset can not follow the temporal dynamics with the emergence or disappearance of new entities in the blockchain. For example at time step t = 43, which follows a dark market shutdown, none of the models performs well after this event as the figure 2.1 shows, the Elliptic dataset can not work well in sudden changes in the underlying behavior of a system [33].

ID	**Feature 1**	**Feature 2**	**Feature 3**	**Feature 4**	**Feature 165**	**Feature 166**
230425980	-0.17146	-0.18466	-1.2013	-0.12061	-0.1206	-0.1197

Table 2.1: Elliptic dataset.

Wai [29] proposed Inspection-L, which is a graph neural network (GNN) framework based on self-supervised Deep Graph Infomax (DGI) and Graph Isomorphism Network (GIN). Supervised learning algorithms like Random Forest (RF) are employed to detect illicit transactions for Anti-Money Laundering (AML) purposes. In another work, Mark [18] utilized Graph Convolutional Networks (GCN) for financial forensics, and explored binary classification tasks for predicting illicit transactions using logistic regression, random forest, multilayer perceptron, and GCN. To improve upon existing methods, combinations with GCN have been investigated. Furthermore, Aldo [30] proposed EvolveGCN, a model that adapts GCN along the temporal dimension without relying on

node embeddings. However, most prior research on Bitcoin address clustering for AML primarily utilized pre-made datasets such as elliptic datasets, which lacked detailed headers and had only a few numerical columns.

He [34] analyzed the associations between Bitcoin transactions and addresses. He proposed an improved change address detection algorithm and compared it with the original algorithm to demonstrate its effectiveness. Additionally, the researcher employed the Louvain algorithm along with other heuristics to enhance the clustering method. Sarah [35] explores unique characteristics and employs heuristic clustering to group Bitcoin wallets based on evidence of shared authority. This approach considers multiple public keys observed in the blockchain into larger entities, with heuristics considering different public keys used as inputs to a transaction and change addresses.

Furthermore, Goldsmith [36] analyzed six subnetworks of bitcoin transactions known to be associated with prominent hacking groups. Additionally, Nerurkar [37] focuses on extracting nineteen features from the Bitcoin network and proposes a deep learning-based graph neural network model that utilizes spectral graph convolutions and transaction features, with the incorporation of graph convolutional networks (GCN) as well. Di [38] designed a system to visually analyze the flow of Bitcoin transactions within the blockchain. This tool incorporates the concept of purity, enabling analysts to quickly and effectively identify instances where Bitcoins are being mixed in a suspicious manner, allowing for a better understanding of the timing and methods used in such transactions.

Zheng [39] utilized a novel heuristic method to cluster the incidence relationships between Bitcoin addresses, and employed an enhanced Louvain algorithm to validate the incidence relationships among users, but the heuristic method employed in this research may have some errors when determining change addresses. Ron [40] focused on examining the common patterns observed in user behavior, specifically related to the acquisition, expenditure, and balance of bitcoins in their accounts. Additionally, the research investigates how users transfer bitcoins between different accounts as a means of safeguarding their privacy.

Fleder [41] created a system to extract Bitcoin addresses from public forums. Additionally, a method was developed to link users to transactions using incomplete transaction data. The researchers also introduced a graph-analysis framework that enabled the tracing and clustering of user activity. By combining publicly available information from web-scraped forums and Bitcoin's

transaction ledger, the study demonstrated that the Bitcoin transaction network is not completely anonymous. Moreover, the researchers were able to connect Bitcoin forum users to the original Silk Road nodes.

Tironsakkul [42] introduced a novel approach known as Address Taint Analysis, which builds upon existing techniques used in transaction-based taint analysis. Their primary focus was on tracking Bitcoins that had undergone mixing through a mixer service. They also investigated the potential benefits of combining address taint analysis with address clustering and backward tainting. To minimize false-positive outcomes, they put forward a set of filtering criteria that took into account the characteristics of withdrawn transactions. Furthermore, the researchers identified two potential improvements for both address taint analysis and filtering criteria. Firstly, they recommended the utilization of external information to enhance the accuracy of tracking results. Secondly, they proposed the incorporation of more sophisticated address clustering methods to bolster the effectiveness of the analysis.

Wu [43] introduced a framework called FABT, which aims to facilitate the forensic analysis of Bitcoin transactions through the identification of suspicious Bitcoin addresses. The framework establishes a formalized approach for analyzing transaction patterns based on a wide range of features, enabling the examination of various clues within a given case. Spagnuolo [44] introduced BitIodine, a modular framework, which is designed to parse the blockchain and perform various tasks related to address clustering, user classification, and information visualization. The framework utilizes two heuristics, namely Multi-input transactions and change address guessing, to identify addresses that are likely to belong to the same user or group of users. It then proceeds to classify and label these users, while also visualizing the extracted information from the Bitcoin network in a comprehensive manner.

Shojaeinasab [45] addressed the issue of losing traceability of funds in Bitcoin and other cryptocurrency networks due to the utilization of mixing services. The researcher develops techniques to trace the transactions and addresses associated with these services, as well as identify the addresses involved in the circulation of both illicit and clean funds.

There are many machine learning methods such as support vector machines, decision tree, logistic regression, and neural networks (deep learning). These classifiers each have their strengths

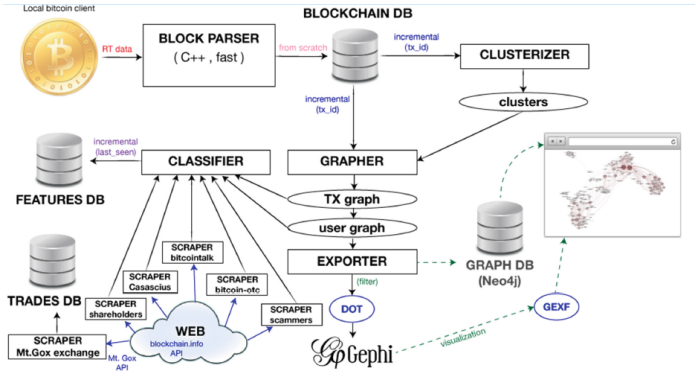

Figure 2.2: Building blocks of BitIodine [44]

and weaknesses, and their effectiveness can vary depending on the nature of the data and the specific problem at hand. Archie [46] provided an SVM model for binary classification of bitcoin addresses with a small number of bitcoin addresses 114000 samples that provided the below results.

Classifier	Length	F1 Score	Precision	Recall	Accuracy	AUC
Service_binary_SVM[46]	114K	0.727	0.907	0.907	0.606	

Table 2.2: Evaluation metrics binary classification with SVM classifier [46]

Classifier	False Negative	False Positive	True Positive	True Negative
Service_binary_SVM[46]	6K	1.4K	10.6K	6.5K

Table 2.3: Confusion matrix binary classification with SVM classifier [46]

Based on our results, the SVM model can not generate good results with a large number of bitcoin addresses, Archie [46] just worked on binary classification with old and small amounts of bitcoin addresses. He [34] introduced five heuristic methods for Bitcoin address clustering. Researchers used the address 18yVghac MaDU 8SzG48 7h2e Qvj2S aUCbtXj as an example, that applied the heuristics H1, H1 + H2, H1 + H2 + H3, and H1 + H2 + H3 + H4 iteratively. Through experiments, they found that under different heuristic conditions, the number of addresses obtained was 35, 39, 120, and 6,469, respectively. These addresses were divided into 74,286 entity nodes across 1,247 communities. However, He [34] did not provide any statistical results such as accuracy and precision, the research solely presented numerical outcomes regarding the number of clusters and communities formed as figure 2.3 shows. Researchers provided 6 common Heuristics as table 4.1.

Algorithm	Name
H1	Common-input-ownership heuristic
H2	Change address detection heuristic
H3	Coinbase transaction mining address clustering heuristic
H4	Multiple mining pool address clustering heuristic
H5	Mixing transaction recognition heuristic
H6	Louvain community detection algorithm

Table 2.4: Heuristic name in paper [34].

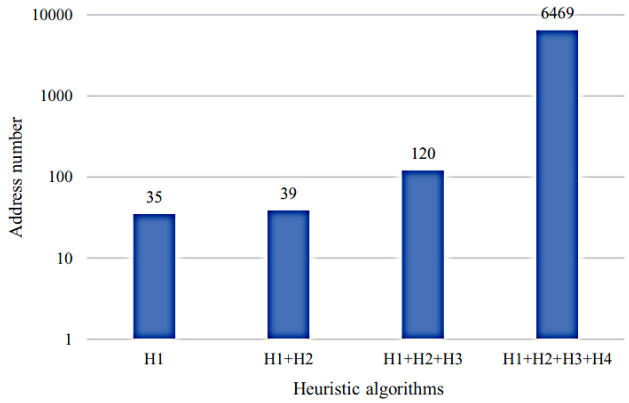

Figure 2.3: Heuristics results in paper [34]

Heuristic algorithm H1 [34] is an address clustering technique that leverages the concept of common-input-ownership within the Bitcoin transaction framework when multiple addresses are utilized as inputs for a transaction. H1 considers that these input addresses can be logically grouped together, indicating a shared control by a singular transaction entity. The accuracy of address clustering achieved through H1 can theoretically attain 100%.

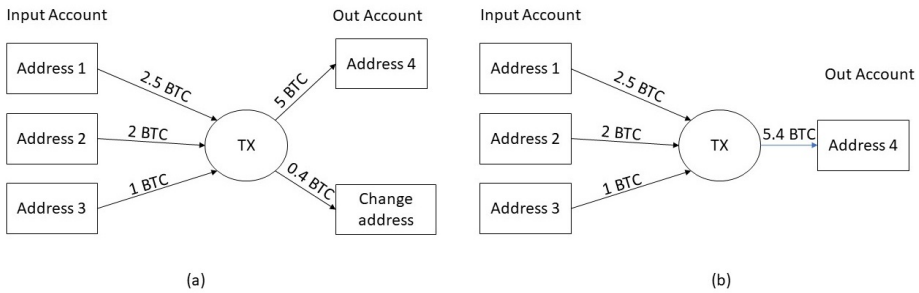

Figure 2.4: Common input ownership schematic diagram H1 [34]

Another address clustering strategy is the change address detection heuristic algorithm. The

investigations conducted by Androulaki [47] and Meiklejohn [35] show the importance of change addresses in user privacy within the Bitcoin ecosystem. These findings show the role of change addresses in enhancing the anonymity infrastructure.

Heuristic algorithm 2 (H2) [34] is used for identifying change addresses in Bitcoin transactions. Change addresses are addresses used to receive the remaining funds in a transaction after the intended payment is made. The existing methods for identifying change addresses in Bitcoin transactions rely on four specific conditions including the address only being used once as an output and not being part of a Coinbase transaction. He [34] suggests two scenarios for identifying change addresses. In the first scenario, if a transaction has only two output addresses and one of them appears only once while the other has significantly more funds, the one-time address is considered the change address. In the second scenario, if a transaction has more than two output addresses, the change address is identified based on the four conditions proposed by Meiklejohn [35]. This new approach aims to improve the accuracy of change address detection and facilitate the analysis of Bitcoin transactions and their connections. Androulaki [47] investigated the importance of change addresses in user privacy within the Bitcoin ecosystem. These findings show the role of change addresses in enhancing the anonymity infrastructure.

Heuristic algorithm H3 focuses on clustering output addresses of Coinbase transactions, which are rewards given to miners when they successfully add a new block to the blockchain. Miners combine their computational power under the control of a pool owner. Miners in these pools calculate hash values, and the rewards are distributed by the pool owner. This trend led to more miners joining mining pools for stability and reduced costs. The key insight of this algorithm is that the output addresses of Coinbase transactions, which are responsible for rewarding miners, are typically controlled by the same entity. This assumption is based on the fact that in mining pools, the pool owner controls the payouts to individual miners. This heuristic helps in clustering output addresses of Coinbase transactions, facilitating the analysis of mining activities on the Bitcoin blockchain.

Heuristic algorithm H4 is based on the number of output addresses and is designed for clustering multiple mining pool addresses. The key distinction between H4 and H3 lies in the clustering target. H4 focuses on grouping addresses related to various mining pools, and its heuristic rule is as follows, if a transaction contains more than 100 output addresses, and at least one of these

addresses is recognized as belonging to a specific mining pool, then all the output addresses in the transaction are attributed to the owner of that particular mining pool as Lewenberg [48] and Zheng [49] discussed.

Heuristic algorithm H5 focuses on identifying mixing transactions in the Bitcoin blockchain. Mixing transactions is a way to obscure the source of funds and increase privacy in Bitcoin transactions, which can be used for legitimate purposes but are also associated with potentially illicit activities. H5 aims to recognize these mixing transactions using a heuristic approach, if there are more than four input addresses and output addresses of a transaction, there will be mixing transactions in the transaction [50].

Heuristic algorithm H6 leverages the Louvain community detection algorithm to identify relationships between entities in the Bitcoin blockchain. The Louvain algorithm is known for its efficiency in discovering community structures within large networks. As Blondel [51] mentioned, it can provide a hierarchical view of the community structure. In the context of transaction networks, the Louvain algorithm is employed to divide the network into distinct communities. H6 extends the work of previous heuristic algorithms (H1 to H4), which clustered address groups. With the information obtained from these earlier heuristics, H6 goes a step further by using the Louvain community detection algorithm to uncover relationships between different entities in the Bitcoin transaction network. This can help identify group activities connections, and interactions among various entities, including the possibility of intermediary roles.

Additionally, Meiklejohn [35] introduced two heuristics for Bitcoin address clustering. Heuristic 1 states that if two or more addresses are inputs to the same transaction, they are controlled by the same user. By applying Heuristic 1, they started with 12,056,684 public keys and ended up with 5,577,481 distinct clusters. Heuristic 2 focused on the one-time change address, which is controlled by the same user as the input addresses. Applying Heuristic 2 resulted in 3,384,179 clusters.

These studies relied on heuristics that primarily focused on common input ownership and change addresses for clustering. They provided numerical results in terms of the number of clusters or addresses within a class. Most of the studies focus on finding illicit activity based on closed-source datasets such as elliptic datasets. These datasets have old and small amounts of Bitcoin addresses that are not useful for Bitcoin classification in the real world. There is no reliable research about

Bitcoin address classification in the real world without focusing on illicit detection. Our research focuses on graph behavior features and machine learning methods. Our research classifies Bitcoin addresses into binary and multi-class categories and links Bitcoin addresses to their real-world owners by using our own dataset and provides better statistical and evaluation metric results than the previous works.

If we classify Bitcoin addresses we can find very useful information about the addresses to examine the privacy implications associated with the use of Bitcoin in commercial transactions.

Chapter 3

Classification framework

3.1 Creating dataset

Available datasets are not open source, they do not have information about the features and they do not have enough Bitcoin address samples also they are not updated. These problems with the available datasets made us create our dataset. Creating the dataset has three phases as downloading Bitcoin addresses through Bitcoin core and web scraping for labeling Bitcoin addresses and creating features for each Bitcoin address that we labeled.

3.1.1 Obtaining Bitcoin addresses

To work with Bitcoin data we have to download the Bitcoin blockchain on our local computer, there are large amounts of data in the Bitcoin blockchain that contains input, output addresses, transactions, blocks, and other information, and also there are some links between transactions that we need to explicit these links to consider graph features of Bitcoin addresses. Therefore, we need Bitcoin block parsers to explicit links in these large amounts of transactions. There are different block parsers in GitHub but most of them require a long time to parse this large amount of data; however, the Bitcoingraph parser can do this task in a short period of time so we use this block parser for our task [52].

3.1.2 Enriching blockchain data

After downloading our Bitcoin addresses, we need to label these Bitcoin addresses in different classes, to test machine-learning classification algorithms, so we have to find tagged Bitcoin addresses through different websites such as Wallet Explorer, Bitcoinwhoswho, and Bitcoinabuse [53]. There are a lot of tagged historical Bitcoin addresses on these websites that are distributed in different parts and pages which need preprocessing to use. Therefore, we can not take them manually or use the wget command to download website pages. We have to write a Python code that automatically gathers Bitcoin addresses and preprocesses the available tags in different parts of Wallet Explorer, Bitcoinwhoswho, and Bitcoinabuse websites. Then the code tags Bitcoin addresses through these tags and saves them with their category. We consider five distinct classes: crypto-currency exchanges, online marketplaces, mining pools, fundraising/charity platforms, and gambling; and 180 companies for fine-grained classification. This procedure is called web scrapping or web crawling which is a common method for gathering data from websites. For web scraping, we must use Regex for pattern matching and extracting specific information from HTML.

3.1.3 Calculating Bitcoin address features

To complete the dataset we need to calculate different features for each Bitcoin address. We choose Bitcoin address features based on class behaviors and Bitcoin graph behaviors. For example, transactions in the crypto-currency exchanges class often have a large number of connections that cause the exchange transactions to have more **in degree** and **out degree** than other categories. Since **component ID** refers to the connected component to which a node belongs, exchange transactions based on their behavior usually have more **component ID**, so these kinds of features are useful for our classification. Mining pools usually involve large transactions, as they collect rewards from many miners. Therefore, transactions associated with pools might have higher **values**. Since mining pools might have transactions spread across various block heights as they continuously receive contributions from miners; therefore, the **height** of bitcoin addresses in the blockchain would be a good feature for classification. Mining pools may exhibit characteristics of a highly connected subgraph (high **k-core**) as many miners contribute to the same pool so the **k-core** and **core id** would

be good features for classification purposes. For online marketplaces, the **delta** might vary more as they receive payments and spend funds for different activities so it can be a good feature for classification as well.

Class	Feature
Exchange	High in degree and out-degree
Exchange	High Component ID
Mining pool	High values
Mining pool	height is important
Mining pool	High k-core
Online marketplace	High Delta

Table 3.1: Bitcoin address feature calculation

We calculate 30 features such as PageRank, component ID, k-core, entity ID, value, delta, in degree, out-degree, core id, avg value, in degree std, out degree std, and height as table 3.2 shows. For calculating features we use Python and the Turicreate library built-in functions. Turicreate is a library for working on graphs for machine learning purposes. For example, for calculating in degree feature we calculate the number of transactions incoming to that address using the transaction history of the entire Bitcoin blockchain. Since we are working on a huge amount of data like Bitcoin transactions we need to use special big data tools such as SFrame, which is a data structure for processing large-scale data efficiently. We merge the output files from the block parser into a single SFrame.

Since Bitcoin transactions form a graph, we need to consider all the blocks together. It is not feasible to calculate features for small block numbers and merge them into a larger block number as graph features such as PageRank, component ID, and core ID are inherently tied to the specific block numbers. For example, if we use block 100000 110000 and block 110000 120000 separately, the resulting features for each address are calculated independently. Merging block 110000 120000 with the features of block 100000 110000 would yield different results compared to directly calculating the features for block 100000 120000, which is more accurate. We use a large block number range, specifically from block height 400000 to 440000, which contains a total of 3M Bitcoin addresses. This block range allows for more accurate feature calculation.

The K-core decomposition recursively removes vertices from the graph with degree less than

Feature	Description
PageRank	PageRank for each vertex (Address) in the graph
Component ID	component ID corresponding to each vertex (Address) in the graph
Entity ID	calculate the entity ID for each Bitcoin address using the common input ownership
Value	amount of bitcoin that bitcoin address contains
Delta	total changes in PageRank during the last iteration for each vertex (Address)
In degree	number of edges directed into that vertex or number of incoming edges to a particular vertex
Out degree	number of edges directed out from that vertex or number of outgoing edges from a particular vertex
Core ID	measure of global of vertex centrality
avg value	average value of the bitcoin amount that bitcoin address contains
In degree std	standard deviation of in degree feature
Out degree std	standard deviation of out degree
Height	block number of the transaction that has the bitcoin addresses
Shortest path	shortest directed path distance from a single source vertex to all other vertices
Triangle	counts the number of triangles in the graph and for each vertex
Kmax	the maximum core id assigned to any vertex
Kmin	the minimum core id assigned to any vertex
Color ID	color ID assigned to each vertex represents a numerical label identifying the color group to which the vertex belongs
txsize	the size of this transaction (tx) in bytes
Min sent	minimum BTCs sent in a tx by inputs
Max sent	maximum BTCs sent in a tx by inputs
Var sent	variance of BTCs sent in a tx by inputs
txfee	The fee paid by this transaction
Var received	variance of BTCs received in a tx by outputs
Time	date and time of the transaction
Max received	maximum BTCs received in a tx by outputs
Total received	total BTCs received in a tx by outputs
txinput val	total BTCs sent by inputs in a tx
txoutput val	total BTCs sent to outputs in a tx
Total sent	total BTCs sent in a tx by inputs

Table 3.2: All the features that we calculate

k. The value of k where a vertex is removed is its core ID (vertex's core id). The algorithms iteratively remove vertices that have less than k neighbors recursively. The algorithm guarantees that at iteration k+1, all vertices left in the graph will have at least k+1 neighbors. The vertices were removed at iteration k is assigned with a core ID equal to k.

In the context of a graph, such as a Bitcoin graph, each node represents a Bitcoin address, and the directed edges between nodes represent the amount of Bitcoin sent or received. The basic

idea behind PageRank is to assign a numerical score to each node in the graph, representing the likelihood that a random walker will arrive at that node. The score is computed iteratively based on the structure of the graph.

Here is a simplified explanation of how the PageRank algorithm works. Initially, all nodes are assigned an equal PageRank score. This score could be set to 1/N, where N is the total number of nodes in the graph. The PageRank scores are updated iteratively based on the graph structure. The basic idea is that a node's importance is determined by the importance of the nodes that link to it. The more incoming links a node has, and the more important those linking nodes are, the higher the node's PageRank.

For example, consider a graph where each node represents a Bitcoin address and directed edges represent transactions between addresses. The weight of each edge could represent the amount of Bitcoin transferred in that transaction. Address A has outgoing transactions to B, C, and D. Address B has outgoing transactions to C. Address C has outgoing transactions to A and D. A simplified version of the PageRank algorithm to determine the influence of each address in this Bitcoin transaction graph will be discussed. Assign an equal PageRank score to each address. For simplicity, let's start with equal scores (1/N, where N is the total number of addresses). Update the PageRank scores based on the incoming transactions. The new PageRank score for each address is proportional to the sum of the PageRank scores of the addresses that sent Bitcoin to it. Introduce a damping factor to model the likelihood that Bitcoin users will continue to make transactions. Typically, this is a value between 0 and 1 (e.g., 0.85). Repeat the iteration until the PageRank scores converge.

$$PAGE\_RANK(i) = d + (1 - d) \sum_{j \epsilon N(i)} \frac{PAGE\_RANK(j)}{Out\_degree(j)}$$

Figure 3.1: PageRank formula

3.2 Analysis existing methods

There are many machine learning methods such as support vector machines, decision tree, logistic regression, and neural networks (deep learning). These classifiers each have their strengths and weaknesses, and their effectiveness can vary depending on the nature of the data and the specific problem at hand. Logistic regression is a popular linear model for binary classification, it works well when the relationship between the features and the target variable is approximately linear. However, Logistic regression has problems with complex and non-linear relationship datasets such as Bitcoin datasets. SVM inherently is a binary classifier that can not work well for large datasets, also training an SVM can be computationally expensive, especially for large datasets. SVM is so sensitive to noise and outliers. Random Forests consume a large amount of memory, particularly when dealing with a large number of trees or deep trees. Training decision trees can be computationally expensive, especially for very large datasets, Random Forest can not work with imbalanced datasets such as ours since can be biased toward the dominant class.

We conducted a comparative analysis of different machine-learning techniques on our dataset to determine the most suitable one for the task, which that shows the boosted tree works well for binary classification purposes in our task. The boosted tree has some advantages in comparison with other methods in our task. For example, the boosted tree has high predictive accuracy that can handle complex relationships in the dataset and also can handle missing data well. Boosted trees are flexible and don't make strict assumptions about the distribution of the data and we can use different loss functions with that. Table 3.3 shows a comparison between different machine learning methods that the Boosted Tree classifier can provide better results for our work.

For a better understanding of research in the field of fraud classification with the Elliptic dataset, we simulated these papers [31, 18] and repeated their results as below figure 3.2 shows.

Classifier	F1 Score	Precision	Recall	Accuracy
SVM	0.34	0.58	0.22	0.62
Logistic Regression	0.32	0.58	0.20	0.61
XGBRegressor	0.30	0.51	0.20	0.61
GradientBoostingRegressor	0.29	0.52	0.20	0.57
AdaBoostRegressor	0.27	0.46	0.22	0.60
PoissonRegressor	0.33	0.44	0.15	0.52
LGBMRegressor:	0.34	0.57	0.25	0.63
SGDRegressor	0.28	0.40	0.19	0.53
OrthogonalMatchingPursuitCV	0.37	0.48	0.25	0.59
RANSACRegressor	0.20	0.52	0.24	0.47
LassoLars	0.35	0.59	0.25	0.65
k-Nearest Neighbours	0.29	0.52	0.23	0.59
Bagging Classifier	0.28	0.52	0.17	0.55
Random Forest	0.71	0.72	0.71	0.69
Boosted Tree	0.73	0.72	0.73	0.79

Table 3.3: Comparison between different machine learning methods

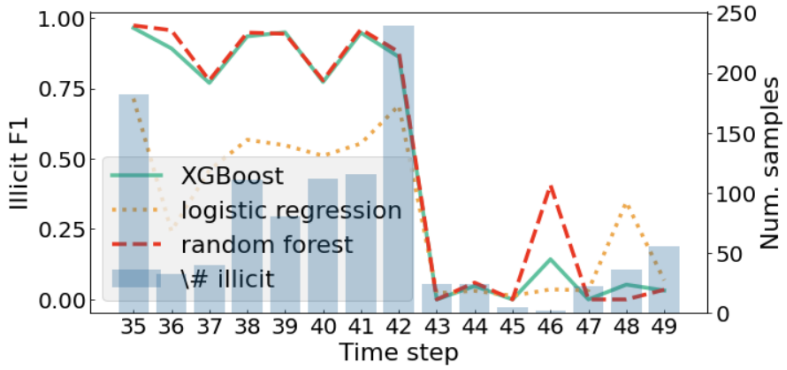

Figure 3.2: Repeated result illicit F1-score [31]

3.3 Our classifier and evaluation

In the binary classification of bitcoin addresses, we conduct a comparative analysis of different machine-learning techniques on our dataset to determine the most suitable one for the task that shows the boosted tree classifier works well for binary classification with our large dataset samples in comparison with Archie's work [46]. However, for multi-classification purposes, the boosted tree doesn't provide good results, which makes us use deep learning methods for multi-classification purposes. One of the main advantages of deep learning is its ability to automatically learn relevant

features from raw data and deep learning does not need manual feature engineering, which was common in traditional machine-learning approaches.

We also explore the coefficients of the features, as they represent the strength and direction of the relationship between each input feature and the output variable. A positive coefficient indicates that an increase in the input feature is associated with an increase in the output variable, while a negative coefficient indicates that an increase in the input feature is associated with a decrease in the output variable. The ”highest positive coefficient” refers to the input feature with the greatest positive impact on the output variable, while the ”lowest negative coefficient” refers to the input feature with the greatest negative impact on the output variable. Based on table 3.4a and 3.4b the most efficient features for binary classification are PageRank, in degree, out-degree,avg value, delta, and core id.

3.4 Deep learning model

For multi-classification and linking Bitcoin addresses to their real-world owners, we use the Multilayer Perceptron (MLP) deep learning model and implement it with the Keras library and Python program language. For the multi-classification purpose, our deep learning model has hidden layers and 10 features as input neurons and Adam optimizer that we achieve 90% as the F1 score and 92% as accuracy as figure 3.3 and for linking Bitcoin addresses to their owners we use the deep-learning model with hidden layers that each has different neurons. The input layer contains 15 features as the input features, also, this study consists of 180 real-world owners' Bitcoin addresses for linking Bitcoin addresses to their owners. We achieve an accuracy of 67% for linking Bitcoin addresses to their owners which is a good result in this regard.

The first row of the table 3.4 illustrates the outcomes of Archie's work [46]. The second and third rows of the table 3.4 illustrate the outcomes of our binary classification with the boosted tree classifier, where a specific Bitcoin address is categorized into either financial or non-financial, or into service-related or non-service-related classes. The last row of table 3.4 presents the outcomes of multiclass classification with the boosted tree, wherein a Bitcoin address is assigned to one of the six designated classes, namely finance, service, gambling, pool, miscellaneous, or charity. Table 3.5

compares our binary classification confusion matrix with the binary classification of Archie's work [46].

Based on the results in table 3.4 and multiclass confusion matrix with the boosted tree classifier, table 3.7 exhibits notable false positive and false negative rates. It indicates that the boosted tree classifier encountered challenges in accurately assigning Bitcoin addresses to the appropriate categories. It is apparent that conventional machine learning methods such as the boosted tree can not provide good performance for multi-class classification, particularly in terms of F1 score. as table 3.4 and table 3.5. To address these issues and improve the performance of the multi-classification model, we use deep learning for multi-class classification and linking Bitcoin addresses to their real-world owner purpose.

Classifier	**Length**	**F1 Score**	**Precision**	**Recall**	**Accuracy**	**AUC**
Service_binary_SVM[46]	114K	0.727	0.907	0.907	0.606	
Finance_binary_Boosted Tree	3.1M	0.73	0.72	0.73	0.790	0.87
Service_binary_Boosted Tree	3.1M	0.76	0.86	0.69	0.780	0.86
Multi_Boosted Tree	3.1M	0.47	0.74	0.76	0.73	

Table 3.4: Evaluation metrics binary and multi-classification with boosted tree classifier in comparison with [46].

Classifier	**False Negative**	**False Positive**	**True Positive**	**True Negative**
Service_binary_SVM[46]	6K	1.4K	10.6K	6.5K
Finance_binary_Boosted Tree	21K	22K	58K	106K
Service_binary_Boosted Tree	34K	12K	75K	87K

Table 3.5: Confusion matrix binary and multi-classification with boosted tree classifier in comparison with [46].

Classifier	**Length**	**Merged**	**T**	**F**
Finance_binary	3.1M	1.05M	322K	520K
Service_binary	3.1M	1.05M	442K	399K
Multi	3.1M	1.05M	-	-

Table 3.6: Dataset that used with Boosted Tree classifier.

Actual / Predicted	Finance	Service	Gambling	Pool
Finance	62565	15559	2396	0
Service	22217	0	0	0
Gambling	0	8426	5314	0
Pool	473	143	20	6

Table 3.7: Confusion matrix for Multi-classification with Boosted Tree Classifier

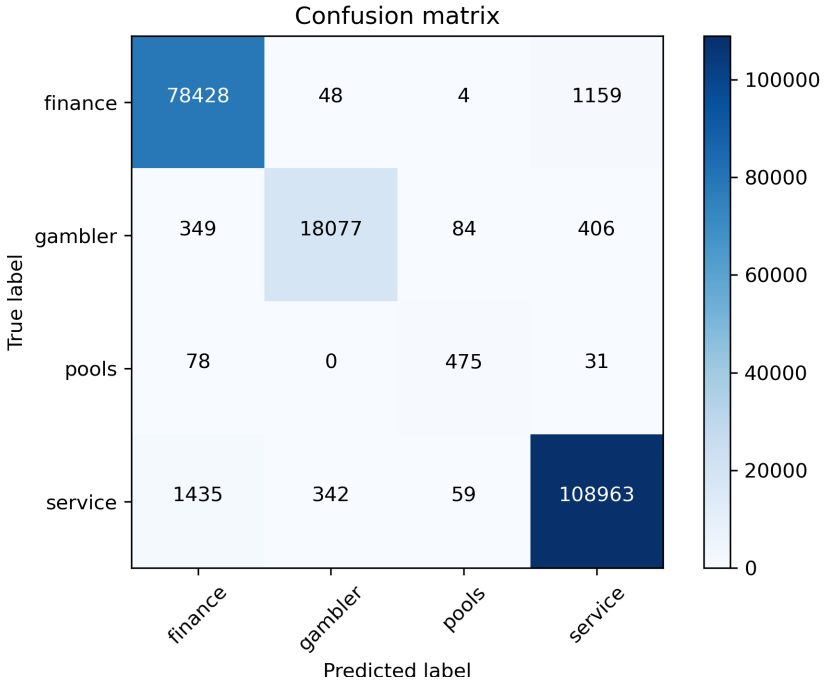

Figure 3.3: Shows the confusion matrix for multiclassification with neural deep learning method

The 3.4a and 3.4b show the coefficient comparison for different features to identify important features with high coefficients.

To test how well our model performs in identifying fraudulent Bitcoin addresses, we fed our model with some known Bitcoin addresses or suspected addresses associated with fraud or scams to determine their respective categories. In online scam alert websites, since their databases about fraudulent Bitcoin addresses are mostly created by user reports, they contain a lot of inactive Bitcoin addresses. An inactive Bitcoin address means this address has no activity yet. Therefore it is not possible to calculate their features such as in degree, out degree, and other graph features. So we should use Bitcoin addresses that have an activity in the blockchain. Our experiment results are in the appendix section. These Bitcoin addresses are Found in the DirtyHash [54] scam blacklist and

Service_SVM	Service_logistic
Highest Positive Coefficients	
Core_id	Core_id
0.0036	0.0199
Avg_value	Avg_value
0.0012	0.0034
Component_id	Component_id
0.0	0.0
Lowest Negative Coefficients	
delta	delta
-4.4436	-0.5642
Intercept	Intercept
-1.088	-2.215
pagerank	pagerank
-0.0013	-0.0002
Out_degree	out_degree
-0.0005	-0.0
In_degree	in_degree
-0.0002	-0.0

(a) Coefficients of each feature for Service models.

Finance_SVM	Finance_logistic
Highest Positive Coefficients	
(intercept)	(intercept)
1.0915	2.1556
pagerank	
0.0017	
in_degree	
0.0006	
transaction_count	
0.0001	
Lowest Negative Coefficients	
delta	delta
-82.344	-1.162
core_id	core_id
-0.0127	-0.1752
avg_value	avg_value
-0.0012	-0.0025
out_degree	pagerank
-0.0004	-0.0004
component_id	in_degree
-0.0	-0.0001

(b) Coefficients of each feature for Finance models.

Figure 3.4: Coefficients of each feature.

dark web websites that are labeled as Fraud.

3.4.1 Tracking dark web UnlockDevices marketplace transaction

Based on dark web information the UnlockDevices marketplace provided this Bitcoin address 3LnzwDcMdRFbVLG6B71e68ydQ4JYWaKUrE for payment. This Bitcoin address has 85 transactions, and 0.61625000 BTC was sent to 16csF6xeY 2bj1EZWk 3B8nGj9md vTWaNdpe in one of the transactions as below the figure 3.5, the UnlockDevices marketplace bitcoin address sent some Bitcoins in multiple transactions at different times to this address 16csF6xeY 2bj1EZWk 3B8nGj9md vTWaNdpe that figure 3.5 shows one of them.

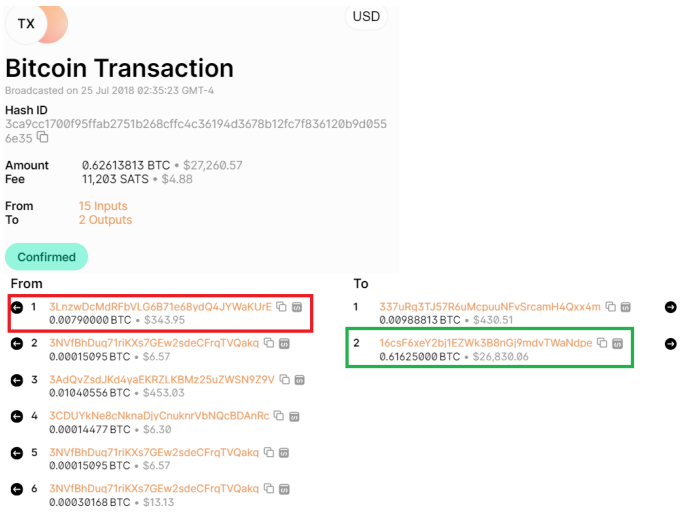

Figure 3.5: Tracking Bitcoin address in Dark web

Based on our investigation the 16csF6xeY2bj1EZWk3B8nGj9mdvTWaNdpe can be a mixer or an exchange since this Bitcoin address is in a wallet that has 90000 Bitcoin addresses. As we checked the majority of its payments are made into the Bitcoin address 1NDyJtN Tjmwk5 xPNhj gAMu4HD Higtobu1s it is more probable to be a mixer since the majority of its payments are made into the Bitcoin address 1NDyJtN Tjmwk5 xPNhj gAMu4HD Higtobu1s which belongs to the Binance exchange.

The Bitcoin address 1NDyJtNTjmwk5 xPNhjgAM u4HDHig tobu1s belongs to the Binance exchange as they announced that [55]. Our model classified this Bitcoin address as an exchange. As we checked most of the Bitcoin addresses in the wallet that contains 16csF6xeY 2bj1EZW k3B8nGj9md vTWaNdpe sent money to the Binance address as the final source of all of their transactions. The figure 3.6 shows the tracking in summary.

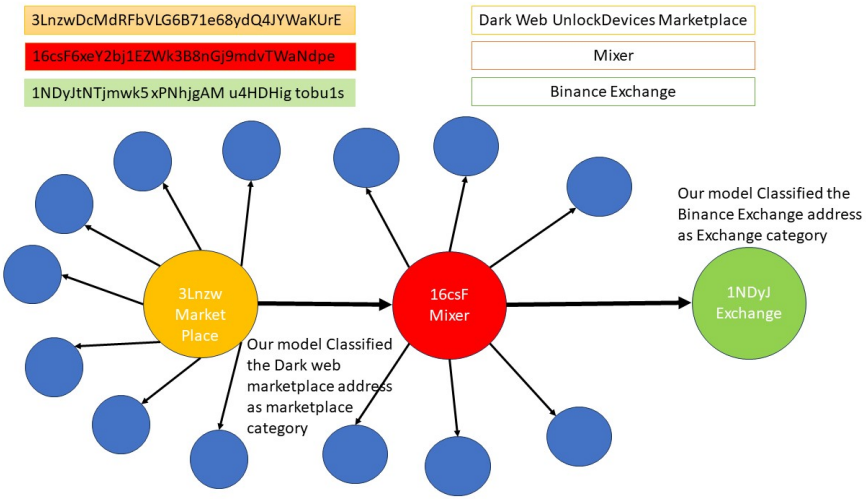

Figure 3.6: Tracking Bitcoin address in Dark web

In summary, the research project has successfully fulfilled its primary objectives of classifying Bitcoin addresses into binary and multi-class categories and linking Bitcoin addresses to their real-world owners. The research project provides better statistical evaluation metric results than the previous works.

Chapter 4

Implementation

Our project has some implementation details that are discussed in this chapter. The project is a mixture of big data and machine learning so we need to consider some specific approaches.

4.1 Block parser

We need Bitcoin block parsers to explore links in Bitcoin transactions. There are different block parsers in GitHub, for example, Rusty block parser [56]. Most of the block parsers like the Rusty block parser provide outputs that are only usable by relational databases. Based on our experience importing block parser outputs to relational databases takes a long time. Using Rusty block parser that works with relational databases can not show graph relationships in Bitcoin transactions. We use Bitcoingraph [52] block parser which is much faster than other block parsers. The Bitcoingraph block parser outputs are compatible with the graph databases that can show graph relations in Bitcoin transactions better than other block parsers. The table 4.1 shows Bitcoingraph block parser outputs.

We convert Bitcoingraph block parser outputs to SFrame. Since we are working on a huge amount of data like Bitcoin transactions we need to use big data data structures such as SFrame. To convert the dataset to SFrame we use the TuriCreate library [9] developed by Apple. TuriCreate simplifies the development of custom machine-learning models and converting data to SFrame, table 4.2 shows TuriCreate common ML tasks.

Output file	Description
addresses.csv	sorted list of Bitcoin addressed
blocks.csv	list of blocks (hash, height, timestamp)
transactions.csv	list of transactions (hash, Coinbase/non-Coinbase)
outputs.csv	list of transaction outputs (output key, id, value, script type)
rel-block-tx.csv	relationship between blocks and transactions (block-hash, tx-hash)
rel-input.csv	relationship between transactions and transaction outputs (tx-hash, output key)
rel-output-address.csv	relationship between outputs and addresses (output key, address)
rel-tx-output.csv	relationship between transactions and transaction outputs (tx-hash, output key)

Table 4.1: Bitcoingraph block parser outputs

ML Task	Description
Recommender	Personalize choices for users
Image Classification	Label images
Drawing Classification	Recognize Pencil/Touch Drawings and Gestures
Sound Classification	Classify sounds
Object Detection	Recognize objects within images
One Shot Object Detection	Recognize 2D objects within images using a single example
Style Transfer	Stylize images
Activity Classification	Detect an activity using sensors
Image Similarity	Find similar images
Classifiers	Predict a label
Regression	Predict numeric values
Clustering	Group similar datapoints together
Text Classifier	Analyze sentiment of messages

Table 4.2: TuriCreate common ML tasks

4.2 Memory efficiency methods

Memory usage in our project is very important since we are working with a huge amount of Bitcoin transactions. We use some techniques to optimize memory usage. There are 3 important big data structures Spark, Dask, and SFrame. We checked all of them in our project and Spark can not work well with the Bitcoin transaction dataset based on our results. We use SFrame and Dask dataframe structures. The other technique to optimize memory usage is deleting variables that are no longer used within the code. There are some interdependencies among certain variables deleting a single variable may not necessarily lead to the release of memory. To address this problem we use the memory profiler and memory visualization method. The memory profiler method calculates the memory usage of each code line. The memory visualization method explores how variables are

interrelated and linked to each other. Therefore, we can find interdependencies between variables that might hinder effective memory management. We also change the default location for storing libraries and temporary working files. This is so efficient for working with large datasets and large results in the output. Dask, SFrame, and Spark dataframes are data structures used in different distributed computing frameworks for processing large-scale data efficiently. Each of them has its own unique characteristics and usage scenarios.

Dask dataframe is part of the Dask library, which provides parallel computing capabilities for Python. It is designed to handle large datasets that do not fit into memory by breaking them down into smaller partitions that can be processed in parallel across multiple cores or machines. Dask leverages task scheduling to execute operations on these partitions in a distributed manner. Dask dataframe closely resembles the Pandas dataframe API, making it easy for users familiar with Pandas to work with larger-than-memory datasets. It supports most of the Pandas dataframe operations and provides scalability by distributing the computations.

SFrame is a data structure provided by the TuriCreate library (previously known as Graphlab Create) for scalable machine learning and data manipulation tasks. TuriCreate is primarily used for building machine learning models, and the SFrame is its core data structure for handling large datasets. SFrame is optimized for memory efficiency and fast data manipulation on large data. It can be thought of as a horizontally partitioned dataframe, where each partition holds a subset of the overall data. SFrame supports various data transformations, filtering, and joins, and it can handle datasets that are too large to fit into memory.

Spark is part of the Apache Spark framework, a distributed computing system for big data processing. Spark dataframe is an abstraction built on top of the distributed Resilient Distributed Dataset (RDD) providing a high-level API for working with structured and semi-structured data. Spark dataframe is designed for distributed data processing and fault tolerance. It supports various data operations, including filtering, aggregation, joins, and window functions. Spark optimizes data processing by performing distributed transformations and leveraging memory caching for iterative computations.

Each of these data structures has its strengths and usage cases, depending on the specific requirements and the scale of data processing tasks. Choosing the right one depends on factors like

the size of the dataset, the complexity of operations, and the existing ecosystem and tools that we are working with. We checked all three Spark, Dask, and SFrame, and finally, Dask and SFrame work better than Spark.

4.3 Implementation steps

Figure 4.1 shows the implementation steps that we followed in summary. As figure 4.1 shows the implementation steps start with running the Bitcoin core. By downloading Bitcoin core on our local computer we can run a full node on the Bitcoin network. Running a full node means we can have a complete copy of the entire Bitcoin blockchain on our local computer. Bitcoin full node is about 500 GB, the downloading time of this large amount of data relies on some factors such as internet speed, RAM, and operating system. Based on our experience, Linux works better than others. In the next step, we run the Bitcoingraph parser to find links that exist between transactions explicitly. There are some links between Bitcoin transactions and we need to find these links to consider graph features of Bitcoin addresses. For this purpose, we have to use block parsers. There are different Block parsers like Rusty and Bitcoingraph.

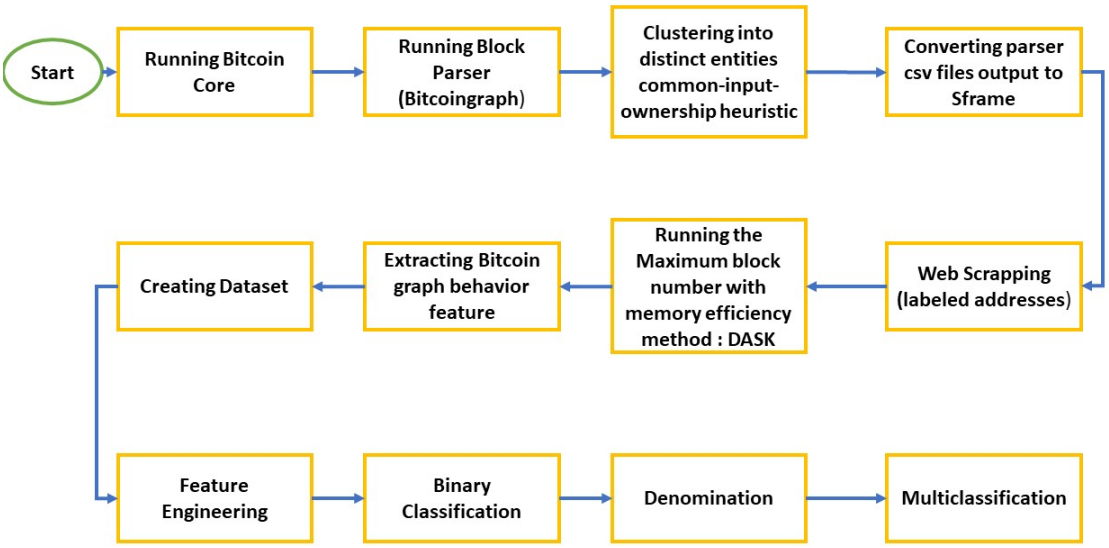

Figure 4.1: Implementation Steps

Some of the block parsers can not show the links between Bitcoin addresses explicitly and usually, they require a long time to parse this large amount of Bitcoin data. For example, Rusty Block Parser provides 4 files as output. To use these files we have to import them into the SQL database. However, it has some main problems such as it takes a long time to import its output files to the SQL database, and also it can not show the links between bitcoin transactions explicitly.

We choose the Bitcoingraph as it provides more practical data that is compatible with the Neo4J NoSQL database which helps to consider the graph behavior of Bitcoin transactions and links between Bitcoin transactions. Table 4.3 shows the Rusty block parser outputs, and also figure 4.2 shows the Noe4j output for Bitcoin transactions. Neo4j is a graph database management system that is designed to store, retrieve, and manage data in the form of nodes, relationships, and properties. It is particularly well-suited for applications where relationships between data points are crucial, making it a powerful tool for modeling and querying connected data. Neo4j can be used to model and explore the relationships between different elements in the Bitcoin network. By representing Bitcoin transactions and addresses as nodes and relationships in a graph database, we can explore and query the data in a way that emphasizes the connections between different elements. This can be particularly useful for analyzing complex relationships in the Bitcoin network.

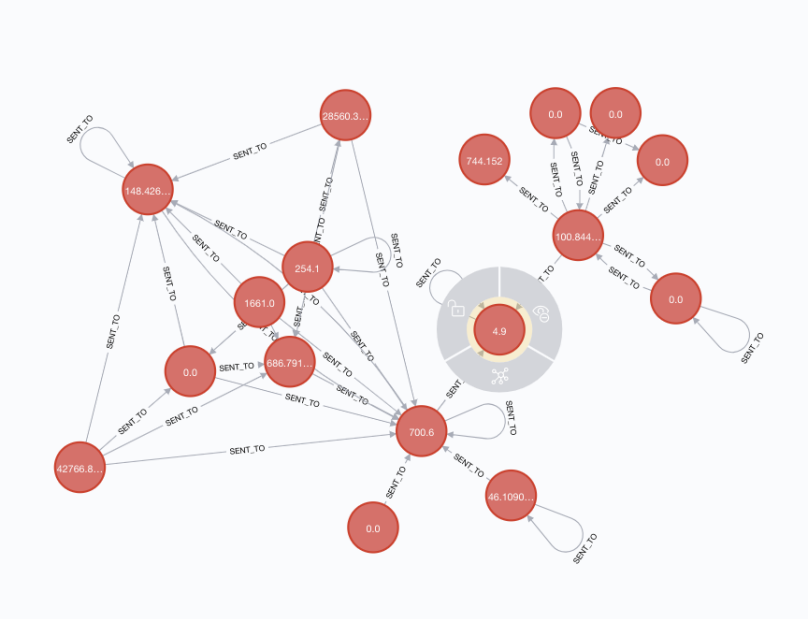

Figure 4.2: Neo4j output for Bitcoin transactions

Output file	Description
blocks.csv	block-hash ; height ; version ; blocksize ; hashPrev
transactions.csv	txid ; hashBlock ; version ; lockTime
tx-in.csv	txid ; hashPrevOut ; indexPrevOut ; scriptSig ; sequence
tx-out.csv	txid ; indexOut ; height ; value ; scriptPubKey ; address

Table 4.3: Rusty block parser outputs

In the next step, we have to apply a pre-clustering method to calculate the entity ID for each Bitcoin address with using the common input ownership heuristic. The converting step converts the block parser outputs to an SFrame for memory management. This step uses the TuriCreate library for converting to the SFrame.

In the web scraping step, we need to label Bitcoin addresses in different classes, to test machine-learning classification algorithms, so we have to find tagged Bitcoin addresses through different websites such as Wallet Explorer, Bitcoinwhoswho, and Bitcoinabuse. This step has two main implementation factors. First, we need to preprocess the available data on these websites. For this purpose, we should use Regular Expression (Regex). It is a powerful tool used for pattern matching within strings. Regular expressions provide concise and flexible means to search, match, and manipulate text. Second, most of the websites have rate limiting. Rate limiting is a technique used by web services to control the amount of incoming requests from a single user or client in order to prevent abuse, ensure fair usage, and maintain system stability. It is often expressed as a certain number of requests allowed per minute, hour, or another defined time period. To solve this problem we have to send requests in a specific pattern.

In the next step, we should obtain the maximum block numbers based on the available memory limitation. Since there is a large amount of data we can not consider many block numbers. By using memory efficiency methods such as SFrame and Dask, the maximum address number that we can use is about 3M Bitcoin addresses. In the next step, we calculate the features for each Bitcoin address using Python and the TuriCreate built-in functions. The features that we calculate, focused on Bitcoin graph behavior. The next step focuses on creating the dataset with the provided data from the previous steps. In the next step, we try to find the most important features that have the most effect on the results. For this purpose, we explore the coefficients of the features, as they represent the strength and direction of the relationship between each input feature and the output variable.

The last 3 steps rely on machine learning methods that we applied to our dataset.

4.4 Project main challenges

Our project has three main challenges, that will be discussed here. First, Bitcoin transactions behave like a graph but showing these graph behaviors and finding the links that exist between transactions are not simple. There are about 1 billion Bitcoin transactions, and a lot of Bitcoin addresses in the Bitcoin blockchain. These Bitcoin addresses may be used in different blocks; for example, one Bitcoin address may appear in block 200 and after several times the Bitcoin address reappears in block 9000; therefore, finding relations between such Bitcoin addresses that are reused in different blocks is not simple. So, finding relationships and graph behavior explicitly in the Bitcoin blockchain is not easy. To solve this problem, we use the Neo4j database to show Bitcoin transaction links explicitly. By using the Neo4j database we can find Bitcoin graph features more accurately and simply.

Second, there is no publicly available dataset suitable for testing machine-learning classification algorithms for Bitcoin addresses, also the labeled data is scarce. To solve this challenge, we use web scrapping to find labeled data through trusted websites.

Third, our project is a mixture of big data and machine learning. Working with this huge amount of data needs to have unlimited memory or use memory efficiency methods to manage memory usage. There are different memory efficiency methods such as using big data structures, deleting unused variables, and setting libraries' cache files to save in storage instead of memory. Deleting unusable variables to release memory is a good approach for memory management but it is not simple. Since there are some hidden interconnections between different variables if we delete one variable the allocated memory may not be released if this variable has interconnections with other variables. So, to solve this problem we use some memory management tools such as memory visualization and memory profiler that show variables interconnections and memory usage of each line of code. These are the main challenges in our project that we solved with different methods.

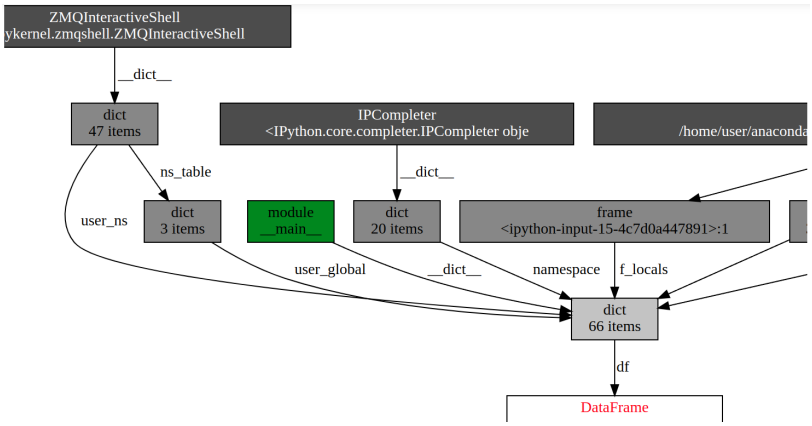

Figure 4.3: Memory visualization interconnections

4.5 Conclusion and Future work

In this study, we presented classifying Bitcoin addresses into binary and multi-class categories and linking Bitcoin addresses to their real-world owners with our own dataset that contains a very large amount of recent Bitcoin addresses. We used Bitcoin graph features and deep learning models and all of these help to achieve good results in classification in comparison with previous studies. However, previous studies relied on heuristics that primarily focused on common input ownership and change addresses for clustering. Previous studies provided numerical results in terms of the number of clusters or the number of addresses within a class. Previous studies do not provide accurate results in the classification and clustering of Bitcoin addresses, also they did not provide information about linking Bitcoin addresses to real owners.

For future research, there are some fields to further improve the classification and analysis of Bitcoin addresses. Adding more off-chain information, into the extracted features can enhance the accuracy and precision of address classification. Future research can find more real owners of Bitcoin addresses to increase the accuracy of linking Bitcoin addresses to their real-world owners. The proposed dataset can be more enriched for anti-money laundry applications by adding illicit Bitcoin addresses to the dataset, this is an interesting field for future research

Appendix A

Our experiment results

These Bitcoin addresses are Found in the DirtyHash [54] scam blacklist and dark web websites that are labeled as Fraud.

Bitcoin address	Description	Predicted
3LnzwDcMdRFbVLG6B71e68ydQ4JYWaKUrE	Dark Web UnlockDevices Marketplace [57, 58]	Marketplaces
37s7r9QiE6pJM5aipFHgpHigCswamXZ4jo	BancoPanama Marketplace [59, 58]	Marketplaces
1FFQrpn8oMUdRmBm9RrcjprngxrYbLaYNw	Dark web AlphaBayMarket Marketplace [60]	Marketplaces
112FWGSL2q7rVTgabQuJbo3WwKid8dMEtj	Dark web Joker's Stash Marketplace [61]	Marketplaces
1QATskw4LGVjhfB5UPZwiyVLKP9zdPcKir	Dark web Scam Advisor Marketplace [62]	Marketplaces
1DZNCsgd9BW7zfwj63A4yYKAGMCog6P7Eh	Dark web SilkRoad Marketplace	marketplaces

34xp4vRoCGJym3xR7yCVPFHoCNxv4Twseo	Binance Exchange [63, 64, 65]	Exchange
3LCGsSmfr24demGvriN4e3ft8wEcDuHFqh	CoinCheck Exchange [66]	Exchange
3MgEAFWu1HKSnZ5ZsC8qf61ZW18xrP5pgd	OK Exchange [67]	Exchange
bc1qchctnvmdva5z9vrpxkkxck64v7nmzdtyxsrq64	Bitmex Exchange [68]	Exchange
1N52wHoVR79PMDishab2XmRHsbekCdGquK	Bittrex Exchange [69, 70]	Exchange
16rF2zwSJ9goQ9fZfYoti5LsUqqegb5RnA	OKX Exchange [71, 72]	Exchange
1J1F3U7gHrCjsEsRimDJ3oYBiV24wA8FuV	F2Pool Pool [73, 74]	Pool
3HhF6vJaghhZGCP7StsLw1gwNJyUtRUSAC	Crypto Marketplace [75]	Marketplace
39884E3j6KZj82FK4vcCrkUvWYL5MQaS3v	Binace Exchange [76]	Exchange
12TaAbLWBNKB1NLYH92CPnC1DizQoNK6FN	Gambling [77]	Gambling
162bzZT2hJfv5Gm3ZmWfWfHJjCtMD6rHhw	Gate.io Exchange [76]	Exchange
3BMEXqGpG4FxBA1KWhRFufXfSTRgzfDBhJ	Bitmex Exchange [76]	Exchange
1LoPb7P5ix4rknWzFH4NHEoFQyru6ett3z	Hacking Darkweb Marketplace [78]	Marketplace
1LU4sSfLAukDcbJim9dvNKELHxk4YMSnW1	F2Pool Pool [76]	Pool
3M219KR5vEneNb47ewrPfWyb5jQ2DjxRP6	Binance Exchange [79, 65, 63]	Exchange
1A7znRYE24Z6K8MCAKXLmEvuS5ixzvUrjH	Kraken Exchange [80]	Exchange
bc1qm34lsc65zpw79lxes69zkqmk6ee3ewf0j77s3h	Binance Exchange [81, 82, 63]	Exchange
3FupZp77ySr7jwoLYEJ9mwzJpvoNBXsBnE	OK Exchange [76]	Exchange
1G1pCNLKZCZde4dgznZDE5wiikQeyDGeuh	Brawker Marketplace [83]	Marketplace

1CbR8da9YPZqXJJKm9ze1GYf67eKAUfXwP	Sheep Marketplace [84]	Marketplace
174psvzt77NgEC373xSZWm9gYXqz4sTJjn	Sheep Marketplace [84]	Marketplace
1B3csgD92bu53CFE44F4dG3MYf6fHM1PVL	BitBooks Marketplace [85]	Marketplace
1CK6KHY6MHgYvmRQ4PAafKYDrg1ejbH1cE	SlushPool Pool [76]	Pool
1MHiYYKmXJPLhbEq2oihAHaHHmwveSm8bU	Deepbit Pool [86]	Pool
3H5JTt42K7RmZtromfTSefcMEFMMe18pMD	Poloniex Exchange [76]	Exchange
bc1q080rkmk3kj86pxvf5nkxecdrw6nrx3zzy9xl7q	KuCoin Exchange [76]	Exchange
1GrwDkr33gT6LuumniYjKEGjTLhsL5kmqC	Bybit Exchange [76]	Exchange
1E8xjHavR1NwVv6tWTC6pX7Z93MWmMmPTx	Minng Pool [87]	Pool
3E2adcep2NRRpriLnWn1AvW3AHKqBx2mMr	Kraken Exchange [76]	Exchange
13Ygcd5spiSnexGkhL29y6J8EokMKX4jW9	Gambling [77]	Gambling
19ihxgbVYHv5P9uAWhPJYcPEqJfqkrGy9n	Mining Pool [87]	Pool
17A16QmavnUfCW11DAApiJxp7ARnxN5pGX	Poloniex Exchange [80]	Exchange
1Kr6QSydW9bFQG1mXiPNNu6WpJGmUa9i1g	Bitfinex Exchange [76]	Exchange
3LYJfcfHPXYJreMsASk2jkn69LWEYKzexb	Binance Exchange [63]	Exchange
1DarkNeMzR1nzUeKkMYs3hWJhSvRzgZ4Fk	Agora darkweb Marketplace [88]	Marketplace
1Eg8WSxvfeZaKmqhNZrECTUb58x3y1AWYN	Sheep Marketplace [89]	Marketplace

3LQUu4v9z6KNch71j7kbj8GPeAGUo1FW6a	Binance Exchange [76]	Exchange
1GympjxbuH3jWUiCHNV4bZccf8zh8aS3fw	Gambling [77]	Gambling
38UmuUqPCrFmQo4khkomQwZ4VbY2nZMJ67	OK Exchange [76]	Exchange
3HroDXv8hmzKRtaSfBffRgedKpru8fgy6M	Gate.io Exchange [76]	Exchange
3DVJfEsDTPkGDvqPCLC41X85L1B1DQWDyh	OKEx Exchange [76]	Exchange
18cHaZhMX8ps4kUy8HbnXMKfYnp82ovUZJ	Mining Pool [87]	Pool
3JZq4atUahhuA9rLhXLMhhTo133J9rF97j	Bifinex Exchange [76]	Exchange
336xGpGweq1wtY4kRTuA4w6d7yDkBU9czU	Coincheck Exchange [80]	Exchange
14eQD1QQb8QFVG8YFwGz7skyzsvBLWLwJS	Kraken Exchange [80]	Exchange
1DYJYmg5u5RpCNSujCQd36ATRavYqbXUry	Weipool Pool [90]	Pool
16rCmCmbuWDhPjWTrpQGaU3EPdZF7MTdUk	Bittrex Exchange [80, 91]	Exchange
16rCmCmbuWDhPjWTrpQGaU3EPdZF7MTdUk	Bitstamp Exchange [80]	Exchange
1AnwDVbwsLBVwRfqN2x9Eo4YEJSPXo2cwG	Kraken Exchange [80]	Exchange
14dxwuQwkQiLbZjJFfciZ26xSGdRU5mKEp	P2Pool Pool [92]	Pool
15qx9ug952GWGTNn7Uiv6vode4RcGrRemh	blockcypher Marketplace [93]	Marketplace
1Ewkm6ejW9HXZRQ7dMofVchhbcuofXTJTg	Digital Art Marketplace [94]	Marketplace
3FHNBLobJnbCTFTVakh5TXmEneyf5PT61B	Binace Exchange [76]	Exchange
3HcEUguUZ4vyyMAPWDPUDjLqz882jXwMfV	Kraken Exchange [76]	Exchange
36hmYCWsH4us2XUEokRVndJxAzSfDkLXXw	Others [2]	Others
18YAvNScUiVqcSnWy8QYjw3FABymw2PiKz	Others [2]	Others
18DRbhvVMsnhQfVPYqoRmyyB3NAKycWTG9	Ball Pool [92]	Pool
1J1zegkNSbwX4smvTdoHSanUfwvXFeuV23	P2Pool Pool [92]	Pool

38fJPq4dYGPoJizEUGCL9yWkqg73cJmC2n	KuCoin Exchange [76]	Exchange
14kmvhQrWrNEHbrSKBySj4qHGjemDtS3SF	Gate.io Exchange [76]	Exchange
1JosHWaA2GywdZo9pmGLNJ5XSt8j7nzNiF	MPoolMonitor Pool [95]	Pool
3QQQftBYbBoKADrovaeveX4Gs6H6MMu6u3	OKEx Exchange [76]	Exchange
3Hi5VHVgmYZYfAPc9aNvQoNXyEv5rYvJQN	Bitstamp Exchange [76]	Exchange
34HpHYiyQwg69gFmCq2BGHjF1DZnZnBeBP	Binace Exchange [76]	Exchange
1ELRFi1VuxMxmzwTyLxRzVNEoxihxsRUBF	E-Pool Pool [96]	Pool
1L2g5dL3burP7vA2agbfeKw6RZKGnRurMf	MegaCoin Pool [97]	Pool
bc1q2qqqt87kh33s0er58akh7v9cwjgd83z5smh9rp	Bybit Exchange [76]	Exchange
3D2oetdNuZUqQHPJmcMDDHYoqkyNVsFk9r	Bitfinex Exchange [80]	Exchange
13ULoNrYTavFPovDkR1DL3iAZksShbQP7g	Bitcoin Chaser Gambling [98]	Gambling
3FrSzikNqBgikWgTHixywhXcx57q6H6rHC	Binance Exchange [80]	Exchange
3Cbq7aT1tY8kMxWLbitaG7yT6bPbKChq64	Huobi Exchange [80]	Exchange
1FuckbFLZpmWLuyHyFJw1RGkWm3yRM1L5D	Ball Pool [92]	Pool
3Eh8hVjYCEVBdxsnyD2P2zrVLGZi84gzi7	Kraken Exchange [76]	Exchange
3GUBZQVvqNsoe6xg5XVvv5X7iJvMUP2Gtg	Others [2]	Others
1MJtsk4AUWU4fSr2sSMzbg8WTUoUdw8byM	Others [2]	Others
1C7VMBwL1DEnxGvEE9zPWkUgK4QiT6QrS8	bustabit.com Gambling [99]	Gambling
3JmxvMqm35aLDUHXDbESy6rQz4M8MBQD32	OKEx Exchange [76]	Exchange
1Bat7qkbWDjN3SqXYyYCmBqKNbTsnuZsVY	Ball Pool [92]	Pool

395vnFScKQ1ay695C6v7gf89UzoFpx3WuJ	Binance Exchange [76]	Exchange
1DXhNVViVYu4xPGUu4pYH2cTr72UPMqKAt	Bitcoin faucet Gambling [100]	Gambling
3ANziRvoBdNGkmGopgyhvzPuBvcL8sRL7S	OKEx Exchange [76]	Exchange
3EGdfMJbhPCnxN44SKNZ94AVt9wwULd67S	Kraken Exchange [76]	Exchange
1FoWyxwPXuj4C6abqwhjDWdz6D4PZgYRjA	Binance Exchange [76]	Exchange
3LhfhgcYNiCceJrjDuMbYWqxPkKkgDQ4ay	OKEx Exchange [76]	Exchange
1Kd8zawHp2Wy5y2JE36mfc5VdfKkgpMqDg	Others [2]	Others
1MooseXJNFugR6r26aGk2AM8v7Crrk3iVE	BetMoose Gambling [101]	Gambling
3J1oFuTTWhHGJF2vLorfvReLkirX1JtWJj	Kraken Exchange [76]	Exchange
11111111111111111114oLvT2	Binance Exchange [76]	Exchange
3G7e21FgygBmWDRMykauLANpuBK8iKqXpJ	OKx Exchange [76]	Exchange
1StatsQytc7UEZ9sHJ9BGX2csmkj8XZr2	ELIGIUS Pool [102]	Pool
33W3jbvNdykbyWtv1bvwLKuJrXW2cUvjzk	OKEx Exchange [76]	Exchange
bc1ql7r624hyquhlk5z42gyercm63ujyu62a3fg64k	OKEx Exchange [76]	Exchange
1JQULE6yHr9UaitLr4wahTwJN7DaMX7W1Z	OK Exchange [76]	Exchange
1LnoZawVFFQihU8d8ntxLMpYheZUfyeVAK	OK Exchange [76]	Exchange
1DcT5Wij5tfb3oVViF8mA8p4WrG98ahZPT	OK Exchange [76]	Exchange
3NAE4hKEvxkjjyRFSWTPbczTE3NgXm8SWh	Others [2]	Others
1CY7fykRLWXeSbKB885Kr4KjQxmDdvW923	OK Exchange [76]	Exchange
36anhsRT1mMZjd617yTb3omgTGEPhSTURc	Others [2]	Others
36NkTqCAApfRJBKicQaqrdKs29g6hyE4LS	OK Exchange [76]	Exchange
3MHcS3Cy6HrDSoXVTc2ttQbx7ZK47i95cb	OKEx Exchange [76]	Exchange

13oPp2XG8PJJUbKUuhQ7ptMC3hnVcc1QzA	SuzukiDICE Gambling [103]	Gambling
12ZDggFG3EyyPuz7PwEq3o58gWWCgB5pf3	YABTCL.com Gambling [104]	Gambling
bc1quq29mutxkgxmjfdr7ayj3zd9ad0ld5mrhh89l2	OKEx Exchange [76]	Exchange
1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY	F2Pool Pool [105, 106]	Pool

Table A.1: Fraudulent Bitcoin address classification results

Bibliography

[1] wallet explorer. *Wallet Explorer .com: smart Bitcoin block explorer*. https://www.wallet explorer .com/. Accessed: Aug. 08, 2023. Sep. 10, 2023.

[2] bitcoin whos who. *BitcoinWhosWho*. https://www.bitcoinwhoswho.com/. Accessed: Aug. 08, 2023. Year Published/ Last Updated.

[3] Satoshi Nakamoto and A Bitcoin. "A peer-to-peer electronic cash system". In: *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf* 4.2 (2008), p. 15.

[4] developer bitcoin. *developer*. https://developer. bitcoin.org/devguide/ block$_chain.htmlblock-chain$. Accessed: Aug. 08, 2023. Year Published/ Last Updated.

[5] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.

[6] Elham A Shammar, Ammar T Zahary, and Asma A Al-Shargabi. "A survey of IoT and blockchain integration: Security perspective". In: *IEEE Access* 9 (2021), pp. 156114–156150.

[7] Google. *Bitcoin$_BigQuery$*. https://cloud.google.com/blog/topics/public-datasets/bitcoin-in-bigquery-blockchain-analytics-on-public-data. Accessed: Aug. 08, 2023. Year Published/ Last Updated.

[8] OXT. *Bitcoin$_BigQuery$*. https://oxt.me/. Accessed: Aug. 08, 2023. Year Published/ Last Updated.

[9] Turi Create. *Turi Create*. https://apple.github.io/ turicreate/ docs/ api/ turicreate. toolkits. graph$_analytics.html$. Accessed: Aug. 09, 2023. Year Published/ Last Updated.

[10] K-Core. *What is the K-Core of a Graph*. https://www.youtube.com/ watch? v=rHVrgbc $_3JA$. Accessed: Aug. 09, 2023. Year Published/ Last Updated.

[11] deep$_learning$. *The Data Scientist*. https://thedatascientist.com/what-deep-learning-is-and-isnt/. Accessed: Aug. 09, 2023. Year Published/ Last Updated.

[12] Kaggle. *Kaggle*. https://www.kaggle.com/datasets/ellipticco/elliptic-data-set. Accessed: Aug. 08, 2023. July 31, 2019.

[13] Catarina Oliveira, João Torres, Maria Inês Silva, David Aparício, João Tiago Ascensão, and Pedro Bizarro. "GuiltyWalker: Distance to illicit nodes in the Bitcoin network". In: *arXiv preprint arXiv:2102.05373* (2021).

[14] Joana Lorenz, Maria Inês Silva, David Aparício, João Tiago Ascensão, and Pedro Bizarro. "Machine learning methods to detect money laundering in the bitcoin blockchain in the presence of label scarcity". In: *Proceedings of the first ACM international conference on AI in finance*. 2020, pp. 1–8.

[15] Soroor Motie and Bijan Raahemi. "Financial fraud detection using graph neural networks: A systematic review". In: *Expert Systems with Applications* (2023), p. 122156.

[16] Guoxiang Tong and Jieyu Shen. "Financial transaction fraud detector based on imbalance learning and graph neural network". In: *Applied Soft Computing* 149 (2023), p. 110984.

[17] Khalid Alkhatib and Sayel Abualigah. "Anti-Laundering Approach for Bitcoin Transactions". In: *2023 14th International Conference on Information and Communication Systems (ICICS)*. IEEE. 2023, pp. 1–6.

[18] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I Weidele, Claudio Bellei, Tom Robinson, and Charles E Leiserson. "Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics". In: *arXiv preprint arXiv:1908.02591* (2019).

[19] Youssef Elmougy and Ling Liu. "Demystifying Fraudulent Transactions and Illicit Nodes in the Bitcoin Network for Financial Forensics". In: *arXiv preprint arXiv:2306.06108* (2023).

[20] Ismail Alarab, Simant Prakoonwit, and Mohamed Ikbal Nacer. “Comparative analysis using supervised learning methods for anti-money laundering in bitcoin”. In: *Proceedings of the 2020 5th international conference on machine learning technologies*. 2020, pp. 11–17.

[21] Ismail Alarab, Simant Prakoonwit, and Mohamed Ikbal Nacer. “Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain”. In: *Proceedings of the 2020 5th international conference on machine learning technologies*. 2020, pp. 23–27.

[22] Jack Nicholls, Aditya Kuppa, and Nhien-An Le-Khac. “FraudLens: Graph Structural Learning for Bitcoin Illicit Activity Identification”. In: *Proceedings of the 39th Annual Computer Security Applications Conference*. 2023, pp. 324–336.

[23] Leandro L Cunha, Miguel A Brito, Domingos F Oliveira, and Ana P Martins. “Active Learning in the Detection of Anomalies in Cryptocurrency Transactions”. In: *Machine Learning and Knowledge Extraction* 5.4 (2023), pp. 1717–1745.

[24] Samantha Jeyakumar, Eugene Yugarajah, Andrew Charles, Punit Rathore, Mari muthu Pala nis wami, Vallipuram Muthukkumarasamy, and Zhé Hóu. “Feature Engineering for Anomaly Detection and Classification of Blockchain Transactions”. In: *Authorea Preprints* (2023).

[25] Patel Nikunjkumar Sureshbhai, Pronaya Bhattacharya, and Sudeep Tanwar. “KaRuNa: A blockchain-based sentiment analysis framework for fraud cryptocurrency schemes”. In: *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE. 2020, pp. 1–6.

[26] Xuanwen Huang, Yang Yang, Yang Wang, Chunping Wang, Zhisheng Zhang, Jiarong Xu, Lei Chen, and Michalis Vazirgiannis. “Dgraph: A large-scale financial dataset for graph anomaly detection”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 22765–22777.

[27] Elena V Feldman, Alexey N Ruchay, Veronica K Matveeva, and Valeria D Samsonova. “Bitcoin abnormal transaction detection based on machine learning”. In: *Recent Trends in Analysis of Images, Social Networks and Texts: 9th International Conference, AIST 2020, Skolkovo, Moscow, Russia, October 15–16, 2020 Revised Supplementary Proceedings 9*. Springer. 2021, pp. 205–215.

[28] Wai Weng Lo, Gayan K Kulatilleke, Mohanad Sarhan, Siamak Layeghy, and Marius Portmann. “Inspection-L: self-supervised GNN node embeddings for money laundering detection in bitcoin”. In: *Applied Intelligence* (2023), pp. 1–12.

[29] Wai Weng Loa, Mohanad Sarhana, Siamak Layeghya, and Marius Portmanna. “Inspection-L: A Self-Supervised GNN-Based Money Laundering Detection System for Bitcoin”. In: (2022).

[30] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. “Evolvegcn: Evolving graph convolutional networks for dynamic graphs”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 04. 2020, pp. 5363–5370.

[31] Joana Susan Lorenz. “Machine learning methods to detect money laundering in the Bitcoin blockchain in the presence of label scarcity”. PhD thesis. Universidade NOVA de Lisboa (Portugal), 2021.

[32] Dylan Vassallo, Vincent Vella, and Joshua Ellul. “Application of gradient boosting algorithms for anti-money laundering in cryptocurrencies”. In: *SN Computer Science* 2 (2021), pp. 1–15.

[33] Author/ Organization. *DirtyHash*. https://medium.com/elliptic/the elliptic data set opening up machine learning on the blockchain e0a343d99a14. Accessed: Aug. 06, 2023. 2019.

[34] Xi He, Ketai He, Shenwen Lin, Jinglin Yang, and Hongliang Mao. “Bitcoin address clustering method based on multiple heuristic conditions”. In: *IET Blockchain* 2.2 (2022), pp. 44–56.

[35] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. “A fistful of bitcoins: characterizing payments among men with no names”. In: *Proceedings of the 2013 conference on Internet measurement conference*. 2013, pp. 127–140.

[36] Daniel Goldsmith, Kim Grauer, and Yonah Shmalo. “Analyzing hack subnetworks in the bitcoin transaction graph”. In: *Applied Network Science* 5.1 (2020), pp. 1–20.

[37] Pranav Nerurkar. “Illegal activity detection on bitcoin transaction using deep learning”. In: *Soft Computing* 27.9 (2023), pp. 5503–5520.

[38] Giuseppe Di Battista, Valentino Di Donato, Maurizio Patrignani, Maurizio Pizzonia, Vincenzo Roselli, and Roberto Tamassia. “Bitconeview: visualization of flows in the bitcoin transaction graph”. In: *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*. IEEE. 2015, pp. 1–8.

[39] Baokun Zheng, Liehuang Zhu, Meng Shen, Xiaojiang Du, Jing Yang, Feng Gao, Yandong Li, Chuan Zhang, Sheng Liu, and Shu Yin. “Malicious bitcoin transaction tracing using incidence relation clustering”. In: *Mobile Networks and Management: 9th International Conference, MONAMI 2017, Melbourne, Australia, December 13-15, 2017, Proceedings 9*. Springer. 2018, pp. 313–323.

[40] Dorit Ron and Adi Shamir. “Quantitative analysis of the full bitcoin transaction graph”. In: *Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers 17*. Springer. 2013, pp. 6–24.

[41] Michael Fleder, Michael S Kester, and Sudeep Pillai. “Bitcoin transaction graph analysis”. In: *arXiv preprint arXiv:1502.01657* (2015).

[42] Tin Tironsakkul, Manuel Maarek, Andrea Eross, and Mike Just. “Tracking mixed bitcoins”. In: *International Workshop on Data Privacy Management*. Springer. 2020, pp. 447–457.

[43] Yan Wu, Anthony Luo, and Dianxiang Xu. “Identifying suspicious addresses in Bitcoin thefts”. In: *Digital Investigation* 31 (2019), p. 200895.

[44] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. “Bitiodine: Extracting intelligence from the bitcoin network”. In: *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*. Springer. 2014, pp. 457–468.

[45] Ardeshir Shojaeinasab, Amir Pasha Motamed, and Behnam Bahrak. “Mixing detection on bitcoin transactions using statistical patterns”. In: *IET Blockchain* (2022).

[46] Archie Norman. *thesis-bitcoin-clustering*. https://github.com/archienorman11/thesis-bitcoin-clustering. Accessed: Aug. 09, 2023. 2017.

[47] Elli Androulaki, Ghassan O Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. "Evaluating user privacy in bitcoin". In: *Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers 17*. Springer. 2013, pp. 34–51.

[48] Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. "Bitcoin mining pools: A cooperative game theoretic analysis". In: *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*. 2015, pp. 919–927.

[49] Baokun Zheng, Liehuang Zhu, Meng Shen, Xiaojiang Du, and Mohsen Guizani. "Identifying the vulnerabilities of bitcoin anonymous mechanism based on address clustering". In: *Science China Information Sciences* 63 (2020), pp. 1–15.

[50] Blockchain $_{chart}$. *Blockchain.com*$_{C}harts$. https://www.blockchain.com/ explorer/ charts/ n-transactions -total. Accessed: Aug. 08, 2023. Year Published/ Last Updated.

[51] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. "Fast unfolding of communities in large networks". In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[52] Bernhard Haslhofer. *bitcoingraph*. https://github.com/behas/bitcoingraph. Accessed: Aug. 09, 2023. 2016.

[53] Bitcoin Abuse. *Bitcoin Abuse Database*. https://www.bitcoinabuse.com/. Accessed: Aug. 09, 2023. Year Published/ Last Updated.

[54] Dirty Hash. *Dirty Hash*. https://dirtyhash.com/. Accessed: Aug. 08, 2023. Year Published/ Last Updated.

[55] Twitter. *Binance*. https://twitter.com/binance/ status/ 9616 66467 32535 8081?lang=en. Accessed: Aug. 06, 2023. February 08, 2018.

[56] Michael Egger. *rusty-blockparser*. https://github.com/gcarq/rusty-blockparser.git. Accessed: Aug. 09, 2023. 2023.

[57] Dark Web. *DarkWebUnlockDevices*. http://unlockdehrka3cbn.onion/home. Accessed: Aug. 06, 2023. 2019.

[58] benjaminstrick. *benjaminstrick*. https://benjaminstrick.com/tracing - an-offshore - bank - and -a - dark - web -service - using - the - blockchain - an -osint - investigation/. Accessed: Aug. 06, 2023. 2019.

[59] Banco Panama. *BancoPanama*. http://bancopanuemswrrz.onion/index.html. Accessed: Aug. 06, 2023. 2019.

[60] Naoki Hiramoto and Yoichi Tsuchiya. "Measuring dark web marketplaces via Bitcoin transactions: From birth to independence". In: *Forensic Science International: Digital Investigation* 35 (2020), p. 301086.

[61] Joker. *darkwebjoker*. 2xgjz4warswhkhjx.onion/cards.html. Accessed: Aug. 06, 2023. February 08, 2018.

[62] Scamadvisor. *darkweb Scamadvisor*. l5jcgrava4h 2joxfcnyas 7qvkqjdze ywn sqnt rmwqpf q7u4 rz2iw jzyd.onion. Accessed: Aug. 06, 2023. February 08, 2018.

[63] Binaceforom. *Binaceforom*. https://www.binance.com/en/blog/community/our commitment to transparency 2895840147147652626. Accessed: Aug. 06, 2023. February 08, 2018.

[64] Binace2block. *Binace2block*. https://www.blockchain.com/explorer/addresses/btc/34xp4vR oCGJym3xR7 yCVPFHoCNxv4Twseo. Accessed: Aug. 06, 2023. February 08, 2018.

[65] Binace 2 bitinfo. *Binace 2 bitinfo*. https://bitinfocharts.com/bitcoin/address/34xp4vRoCGJ ym 3xR7 yC VPF HoCNxv4Twseo. Accessed: Aug. 06, 2023. February 08, 2018.

[66] bitinfochartstop. *bitinfochartstop*. https://bitinfocharts.com/ bitcoin/ address/3LCGsS mfr 24de mGvri N4e3ft8wEc DuHFqh. Accessed: Aug. 06, 2023. February 08, 2018.

[67] bitinfochartstopOK. *bitinfochartstopOK*. https://bitinfocharts.com/bitcoin/address/3MgEAF Wu1 HK SnZ5ZsC8qf61 ZW18xrP 5pgd. Accessed: Aug. 06, 2023. February 08, 2018.

[68] bitinfochartstopBitmex. *bitinfochartstopBitmex*. https://bitinfocharts.com/bitcoin/address/ bc1 qc h c tnvmdva5z9vrpx kkxck64v7nmzd tyxsrq64. Accessed: Aug. 06, 2023. February 08, 2018.

[69] bitinfochartstopBitmex. *bitinfochartstopBitmex*. https://bitinfocharts.com/bitcoin/address/ 1N 52 w H oVR 79PM Dish ab2XmR Hsbek CdG quK. Accessed: Aug. 06, 2023. February 08, 2018.

[70] Bittrexblockchain. *Bittrexblockchain*. https://www.blockchain.com/ explorer/ addresses/ btc / 1N5 2w HoVR79 PMDishab2XmRHs bekC dGquK. Accessed: Aug. 06, 2023. February 08, 2018.

[71] bitinfoOKX1. *bitinfoOKX1*. https://bitinfocharts.com/bitcoin/ address/ 16rF2 zw SJ 9 goQ 9fZfYoti 5LsUqqeg b5RnA. Accessed: Aug. 06, 2023. February 08, 2018.

[72] OKXbockchain. *OKXbockchain*. https://www.blockchain.com/explorer/addresses/ btc/ 16 rF2zw SJ9g oQ9fZf Yoti5LsUqq egb5RnA. Accessed: Aug. 06, 2023. February 08, 2018.

[73] F2Poolblock. *F2Poolblock*. https://www.blockchain.com/explorer/addresses/btc/1J1F 3U7 gHr Cj s E s Ri mDJ3oY BiV24 wA8FuV. Accessed: Aug. 06, 2023. February 08, 2018.

[74] F2Poolbitinfo. *F2Poolbitinfo*. https://bitinfocharts.com/bitcoin/address/1J1F3 U7gHrCjsEsR imDJ3oYBiV2 4wA8FuV. Accessed: Aug. 06, 2023. February 08, 2018.

[75] cryptoMarket. *cryptoMarket*. https://cryptoexchange.com/marketplace. Accessed: Aug. 06, 2023. February 08, 2018.

[76] bitinfo. *bitinfo*. https://bitinfocharts.com/top-100-richest-bitcoin-addresses.html. Accessed: Aug. 06, 2023. February 08, 2018.

[77] bitcointrading. *bitcointrading*. http://www.bitcointrading.com/forum/. Accessed: Aug. 06, 2023. February 08, 2018.

[78] Darkwebhack. *Darkwebhack*. http://yjmmvrbklyrljwnbjwfjqr7 ipdh4ooppt66jmo 4rvvalnpzc-qmwexvid.onion/. Accessed: Aug. 06, 2023. February 08, 2018.

[79] Bince3block. *Bince3block*. https://www.blockchain.com/explorer/addresses/btc/3M219KR 5vEneNb47e wrPfWyb5jQ2DjxRP6. Accessed: Aug. 06, 2023. February 08, 2018.

[80] gitbit. *gitbit*. https://gist.github.com/f13end/bf88acb162bed0b3dcf5e35f1fdb3c17. Accessed: Aug. 06, 2023. February 08, 2018.

[81] Binance4block. *Binance4block*. https://www.blockchain.com/explorer/ addresses/ btc/ bc 1qm 34l s c6 5 zpw79lxes69zkqmk6e e3ewf0j77s3h. Accessed: Aug. 06, 2023. February 08, 2018.

[82] Binance4bit. *Binance4bit*. https://bitinfocharts.com/bitcoin/ address/ bc1 qm34l sc 65zpw7 9lxes6 9z kqmk6ee3ewf0 j77s3h. Accessed: Aug. 06, 2023. February 08, 2018.

[83] brawker. *brawker*. http://www.coindesk.com/bitcoin-brawker-shut-down/. Accessed: Aug. 06, 2023. February 08, 2018.

[84] sheep. *sheep*. https:// www.reddit.com/r/ Sheep Marketplace/ comments/ 1rvlft/i just chased him through$_{ab}itcointumblerand/$. Accessed: Aug. 06, 2023. February 08, 2018.

[85] BitBooks. *BitBooks*. BitBooks.co. Accessed: Aug. 06, 2023. February 08, 2018.

[86] deepbit. *deepbit*. http://deepbit.net/. Accessed: Aug. 06, 2023. February 08, 2018.

[87] Mining. *Mining*. http://hhtt.1209k.com/. Accessed: Aug. 06, 2023. February 08, 2018.

[88] agora. *agora*. https://darknetmarkets.org/guides/agora-marketplace-guides/. Accessed: Aug. 06, 2023. February 08, 2018.

[89] wired. *wired*. http://www.wired.com/2013/12/fbi$_{w}allet/$. Accessed: Aug. 06, 2023. February 08, 2018.

[90] weipool. *weipool*. http://weipool.org. Accessed: Aug. 06, 2023. February 08, 2018.

[91] Bittrexblock. *Bittrexblock*. https://www.blockchain.com/btc/ address/ 16r CmCm buW Dh PjWTrpQGaU 3EPdZF7MTdUk. Accessed: Aug. 06, 2023. February 08, 2018.

[92] bitcointalk. *bitcointalk*. https://bitcointalk.org. Accessed: Aug. 06, 2023. February 08, 2018.

[93] blockcypher. *blockcypher*. http://www.blockcypher.com. Accessed: Aug. 06, 2023. February 08, 2018.

[94] slur. *slur*. http://slur.io. Accessed: Aug. 06, 2023. February 08, 2018.

[95] MPoolMonitor. *MPoolMonitor*. https://bitcointalk.org. Accessed: Aug. 06, 2023. February 08, 2018.

[96] e-pool. *e-pool*. http://e-pool.net. Accessed: Aug. 06, 2023. February 08, 2018.

[97] MegaCoin. *MegaCoin*. http://mega.minepool.net. Accessed: Aug. 06, 2023. February 08, 2018.

[98] bitcoinchaser. *bitcoinchaser*. http://bitcoinchaser.com. Accessed: Aug. 06, 2023. February 08, 2018.

[99] bustabit. *bustabit*. https://bitcointalk.org. Accessed: Aug. 06, 2023. February 08, 2018.

[100] bitcoinzebra. *bitcoinzebra*. http://gamblingwithbitcoins.com/recommends/bitcoinzebra/. Accessed: Aug. 06, 2023. February 08, 2018.

[101] betmoose. *betmoose*. http://gamblingwithbitcoins.com/recommends/betmoose/. Accessed: Aug. 06, 2023. February 08, 2018.

[102] ELIGIUS. *ELIGIUS*. http://eligius.st. Accessed: Aug. 06, 2023. February 08, 2018.

[103] suzukidice. *suzukidice*. http://suzukidice.com/. Accessed: Aug. 06, 2023. February 08, 2018.

[104] yabtcl. *yabtcl*. http://gamblingwithbitcoins.com/recommends/yabtcl/. Accessed: Aug. 06, 2023. February 08, 2018.

[105] F2Pool2. *F2Pool2*. https://www.blockchain.com/. Accessed: Aug. 06, 2023. February 08, 2018.

[106] f2pool4. *bitcoinchaser*. https://www.f2pool.com. Accessed: Aug. 06, 2023. February 08, 2018.