CLASSIFICATION OF ILLICIT ACTIVITY OVER THE ETHEREUM NETWORK USING XGBOOST

STEVEN FARRUGIA

Using account transaction history to detect illicit behavior

MSc Computing Science
Computing Science
Faculty of Science and Engineering
University of Groningen

August 15th 2019

SUPERVISORS:
Dr. George Azzopardi
Dr. Joshua Ellul

The research work disclosed in this publication is partially funded by the
Endeavour Scholarship Scheme (Malta). Scholarships are part-financed
by the European Union - European Social Fund (ESF) -
Operational Programme II – Cohesion Policy 2014-2020
"Investing in human capital to create more opportunities and promote the well-being of society".

European Union – European Structural and Investment Funds
Operational Programme II – Cohesion Policy 2014-2020
"Investing in human capital to create more opportunities and promote the well-being of society"
Scholarships are part-financed by the European Union - European Social Funds (ESF)
Co-financing rate: 80% EU Funds;20% National Funds

Dedicated to my mother, Elaine Farrugia
1965 – 2014

ABSTRACT

The recent technological advent of cryptocurrencies and their respective benefits have also been shrouded with a number of illegal activities operating over the network. Illegal activities may comprise of money laundering, bribery, phishing, fraud and any other crime related schemes. In this work we focus on the Ethereum network, which has seen over 400 million transactions since its inception. Using 2179 accounts flagged by the Ethereum community for their illegal activity coupled with 2502 normal accounts, we seek to detect illicit accounts based on their transaction history using the XGBoost classifier. Using 10 fold cross-validation, XGBoost achieved an average accuracy of 0.963 ($\pm$0.006) with an average AUC of 0.994 ($\pm$0.0007). The top three features with the largest impact on the final model output were established to be 'Time diff between first and last (Mins)', 'Total Ether balance' and 'Min val received'. Based on the obtained results we conclude that the proposed approach is highly effective in detecting illicit accounts over the Ethereum blockchain. Our contribution is multi-faceted; firstly, we propose an effective method to detect illicit accounts over the Ethereum network; secondly, we provide insights about the most important features; and thirdly, we publish the data set that we compiled as a benchmark for future and related works.

ACKNOWLEDGMENTS

With great pleasure I would like to thank my advisors, Dr. George Azzopardi and Dr. Joshua Ellul, for their constant support and guidance throughout my MSc research. My experience working with them has been truly memorable - providing me with strong academic insights and knowledge which will be fundamental in my future career.

I would like to extend my gratitude to Corinne and Kyra who not only supported me but also helped me reach beyond what I thought was possible. A special thanks goes to all those who in some way contributed to the final project, friends, and colleagues.

Last but not least, I am sincerely honored to have been selected as the beneficiary of the Endeavour scholarship scheme.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

1 INTRODUCTION

In this section we present our interest in the detection of illicit behavior, specifically on the Ethereum network, and a structured approach to address the numerous anomalous transactions. Providing the overall scope of our work, we also discern the main three entities which may profit from such a developed tool.

1.1 PROBLEM DEFINITION

The Ethereum blockchain network was launched in 2015, after being proposed by Vitalik Buterin earlier in 2013, with the goal of solving and expanding functionality provided by the ubiquitous Bitcoin system through the introduction of Smart contracts [69]. This was all done while allowing for the decentralisation of computational logic and in tandem preserving an equal level of security and supporting the decentralisation of computational logic.

Fast forward till 2019, and Ethereum consolidated a market share of 14.5 billion USD (ranking 2^{nd} in the cryptocurrency domain) and the title of the largest blockchain with the capacity to deploy smart contracts. Having it's own cryptocurrency, known as Ether, the decentralised platform allows for three primary activities to be executed. These include i) the transfer of Ether, ii) creation of smart contracts and iii) execution of smart contract code [21].

Ethereum, like other cryptocurrency networks, has been subject to a number of entities/users carrying out illicit activity over the network. While pseudo-anonymity over the network is enforced and defined to be one of the fundamental features of such systems, this functionality has been exploited by users carrying out illicit behaviour over the network. Some of these illicit behaviors include smart-ponzi schemes, phishing, money laundering, fraud and crime related activity. Although cryptocurrency technology is constantly evolving and improving in order to provide a defect-less decentralised system for their intended uses, the overall reputation has been negatively impacted due to such aforementioned activities. This has inadvertently led to the development of methodologies and techniques to detect such operations.

Anomaly detection, a mechanism often implemented by banks to detect any anomalous fiat transactions, may also be implemented on blockchain networks - by analyzing the publicly available distributed ledgers produced by the respective cryptocurrencies.

1.2 CAN WE DETECT ILLICIT TRANSACTIONS - ANOMALY TRANSACTIONS?

The distributed ledger made publicly available by the Ethereum blockchain network would most certainly be categorized as big data - with just above a total of 400M transactions and 7.3M blocks as of March 2019 [30]. Having to sift through all these transactions, manually trying to identify any transactions believed to exhibit anomaly characteristics would be both impossible and never ending. The rate at which such blocks, specifically transactions and smart contracts, are being produced on the network would require the implementation of machine learning algorithms.

Such an approach would not only help identify historical transactions which exhibit such behavior, but also classify any newly created transactions with similar features. These features, determined both using the available transaction data as well as any features extracted, act as identifiers. Conceptually, the machine learning algorithms would help distinguish between transactions which display normal and anomalous behavior among user accounts by learning the corresponding features which relate to either anomalous or normal behavior.

We may also split the above primary question into a set of subquestions, defined by their order of implementation;

1. Which features from the transaction dataset are to be utilized as input variables for the algorithms used?
2. What additional features should be extracted to enhance the available data?
3. Which machine learning algorithm would yield the best results given the criteria available in an efficient manner?
4. Can we determine the effect the algorithm parameters have on the outcome of the model results as well as the optimal parameters? Could these aforementioned results be interpreted visually?
5. Can we indeed identify and justify any accounts which exhibit any anomalous behavior?

1.3 MOTIVATION

1.3.1 *Why analyse Ethereum?*

The Ethereum platform, a constantly evolving technology as stated by Vitalik Buterin, allows developers to develop applications easily while retaining the security and decentralization features enforced by the

blockchain [17]. Utilizing the proof of work consensus algorithm to deliver a public economic consensus coupled with the capabilities of a stateful Turing-complete virtual machine, allows for a generalized blockchain platform to be formed. It's potential has helped it become the leading smart contract platform - having the largest number of active developers (between January 2018 and February 2019) [71] and the most used platform [32]. Having a solid foundation and underlying architecture for development - through an abstract layer available over blockchain technology - together with the largest ensemble of developers in the domain, we determine that Ethereum exhibits a high potential for future growth and hence the selected platform for analysis.

1.3.2 *Why should we detect anomalous transactions?*

Banks, as well as government entities and corporations managing and overseeing the transfer of funds, have invested sizeable amounts of money in detecting any anomaly transactions occurring over their network. These transactions may form part of money laundering, bribery, fraud and crime related schemes.

Money laundering has been a great concern for banks dealing with fiat currencies. Till this day and age, even though a number of preventive measures have been implemented to detect any suspicious transaction behavior occurring over their network, numerous reports suggest that users are still able to circumvent these controls [54]. The problem is also not country specific but witnessed on a global scale, with some banks closing up after authorities identified the banks inability to carry out the necessary procedures and inform the relevant authorities [81].

Similar activity is also present over the Ethereum network, with entities and users utilizing the anonymity provided by Ethereum to hide their illicit intentions. The internal security mechanisms defined by blockchain technology (hash functions) are able to solve issues such as the double spending problem, but no tools are in place to detect or flag any suspicious behavior or transactions occurring over the network automatically. Smart contracts deployed over the network may also be hiding illegal schemes for which users are unaware of due to their misrepresented intentions of "high-yielding" investment opportunities [13]. An example of such deployed contracts include smart contract ponzi schemes - solely created with the intent of benefiting the contract creator and possible a few users who invested in the scheme initially. This may all be done while the identity of the aforementioned smart contract creator remains anonymous. Rubixi, a Smart contract ponzi scheme stated to provide a dynamic multiplying factor to ones Ethereum investment, guaranteed quick profits. Further analysis determined that only 22 out of 112 participants who

contributed towards the scheme made a profit, with only 2 making a 40% profit margin [13].

Other anomalous behavior over the network has also been discovered in February 2019, following an account which paid over 450,000$ spread out over 3 transactions in order to transfer a total of 0.1 ETH [34]. Such activity has raised concerns on the actual intentions of the executed transactions; were they genuine mistakes whereby the account owner mistook the transaction fee to be the Ether to be transferred or whether it is part of some larger money laundering scheme.

These two form part of a larger already existing suspicious behavior which is presently available over the Ethereum network. Having the correct tools to detect and flag such behavior would help improve the trust over the network; hindering and interrupting illegal activity from being carried out.

1.3.3 *Who are the stakeholders for such a tool?*

The mechanism to highlight any anomalous transactions may be utilized by government entities, developers, researchers, news agencies as well as private entities/corporations. The order of importance is mostly opinion based and may vary due to beliefs on how the Ethereum network (like all other networks) should be regulated.

1.3.3.1 *Government Entities*

There has been a large shift towards the legalisation aspect of such technologies, with Malta trying to establish itself as the jurisdiction of choice for Blockchain companies as well as a profound regulatory framework [10]. Having said that, while these regulations establish a sense of trust - creating standardized and well documented laws for companies - introducing such a tool will oversee all possible forms of illegality and hence it's effective use for Government entities.

1.3.3.2 *Developers, Researchers and News Agencies*

Developers, researchers and news agencies who have a keen interest in the Ethereum domain and would like to examine the networks behavior and activities. Their active participation and use of such a tool would help provide further insights pertaining to irregular behavior. The community as a whole may therefore contribute towards achieving a safer network by spreading awareness.

1.3.3.3 *Private entities/Corporations*

Corporations who develop smart contracts, carry out transactions on behalf of other entities along with private Ethereum blockchains may utilize such a tool to oversee the flow of funds. This would therefore allow for such entities to keep a close eye on activity across their own

developments - ensuring a safe network among users on the private network. The benefit would be two-fold; for consumers performing transactions over the network and entities providing any necessary services.

1.4 AIMS AND OBJECTIVES

This research aims to identify user account(s) through which a number of suspicious activity/transactions have been identified. We may therefore define the following two project aims;

1. Develop the necessary architecture for the final deliverable tool which may be subjected to future improvements/amendments.
2. Extract and build an adequate dataset to be used for the successful classification of Ethereum accounts using the supervised XGBoost classification model.

To satisfy and address the above aims, the following objectives have been defined - listed according to the order by which they must be completed;

1. Locate addresses for accounts which have been deemed to have participated in illicit activity
2. Obtain all possible transactions enacted by each account
3. Extract new features from the retrieved transactions per account
4. Generate the dataset necessary for classification with the newly extracted features per account - illicit and normal (randomly chosen accounts)
5. Generate results using XGBoost model
6. Carry out parameter optimization so as to further improve model results
7. Provide justification for the models produced results

1.5 SCOPE

The recent technological advent of Cryptocurrencies and their respective benefits have also been shrouded with a number of illegal activities operating over the network. Illegal activities may comprise of; money laundering, bribery, phishing, fraud and any other crime related schemes. While we do not have the resources or time available to potentially carry out our research over all cryptocurrencies networks, we have refined our scope down to the Ethereum network. In spite

of this fact, over 300 million transactions have been executed over the Ethereum network. Using 2179[1] accounts flagged by the Ethereum user community for their illicit activities coupled with 2502 normal accounts, we seek to detect illicit accounts based on their transaction history through the use of the XGBoost classification model. Flagged accounts, produced via the model results, may be looked into by other entities. Hence the proposed solution further instils users trust within the system - offering a safer environment by mitigating illegal activity over the network.

1.6 OVERVIEW OF THE PROPOSED SOLUTION

In order to achieve the desired outcome, the proposed solution can be broken down into 4 key steps as depicted in the flow chart Fig. 2.1. These include the i) acquisition of accounts pertaining to 'normal' and 'illicit' found at random and via the Ethereum community, respectively; ii) executing feature extraction on the respective accounts transactions; iii) applying a dimensionality reduction technique and classifying accounts using XGBoost and Random Forest; iv) utilize a number of visualizations to display the results in an easy to understand manner. Each process is fully dependent on the preceding steps and would therefore have a direct impact on the final results obtained by the machine learning (ML) algorithms.

Figure 1.1: Architectural overview of proposed solution.

A number of procedures have been omitted so as to provide a general understanding of the proposed solution. Realistically, each step contains numerous sub-processes in order to fulfil the final outcome. Further explanations on these sub-processes and justification for the decisions made have been provided in Section 2.3.

1 https://etherscamdb.info/scams

CHAPTER TWO

2 LITERATURE REVIEW

This chapter is composed of three sections; i) Background, ii) Literature Review and the iii) Research contribution. The sections are inherently linked together - first providing explanations on the main components of the project followed by a literature review on similar research in the domain and the projects proposed contribution in the field.

2.1 BACKGROUND

2.1.1 *Blockchain Technology and Decentralized Digital Currencies*

The desire and drive to necessitate decentralized funds/money has been explored in the past theoretically but had never come to fruition till the late 2000s [79]. In 2008, the emergence of Blockchain technology in conjunction with Bitcoin [41] presented us with a revolutionary technology.

At the present moment the digital economy is mostly reliant on trusted authorities. This inherently means that all transactions occurring online pass through the respective entities, requiring us end users to essentially trust that the transactions have been fulfilled in a secure and private manner. This is applicable in both the financial and non-financial domain.

Blockchain paves the way for the establishment of a distributed consensus system in the digital online domain [26]. In doing so, a public ledger of all transactions or digital activities enacted and distributed among engaging entities is available. In other words, the public ledger may be seen as a database containing all record history available for everyone to see, created with an append-only data structure and sustained through a peer-to-peer network [9]. Using the distributed consensus algorithm, transactions must be verified by the majority of entities within the system. Once the transaction has been appended to the ledger, no amendments may be carried out - also known as immutability - made possible through a combination of one-way cryptographic hash functions and the consensus algorithm [8]. This may be circumvented in the event that an individual or group of individuals has the majority control (over 50%) of the network. If this is not the case, the provided ledger is deemed to be both a historical and reconciliated rendition of the ground truth. This inadvertently instils end users with a sense of trust in the system, which is further enforced by public/private key technology without any third party human intervention. Double records, also known as the double spending problem,

are therefore intercepted and stopped from being recorded on the distributed ledger in an automated manner. Blockchain technology benefits are threefold; i) allows for algorithms to be deployed on top of the existing Blockchain technology to fulfil business processes or any other potential applicable uses in the form of smart legal contracts [25], ii) facilitate digital payments between entities in a trustworthy and secure manner and iii) stores all records and assets in the form of an immutable database [5].

2.1.2 *Ethereum*

The recent developments and advancements in Blockchain technology have given rise to many open source platforms, one of which is Ethereum. Developed and launched by Vitalik Buterin back in July 30^{th} 2015, Ethereum offers a substantial increase in both capability and purpose when compared to Bitcoin. While Bitcoin is limited to offering a single application of Blockchain technology - facilitating a peer-to-peer electronic payments system for Bitcoin payments and tracking of bitcoin ownership, Ethereum addresses several limitations mainly through offering full Turing-completeness through its programming language [28].

Ether is the digital currency or internal crypto-fuel used within the Ethereum network, for transaction fee payments. One Ether corresponds to 10^{18} Wei, for which Wei is the smallest amount of acknowledged Ether. This enhancement allows Ethereum to cater for all forms of computations, keep track of the state of the executing transaction and other advancements over the Blockchain architecture. Ethereum may therefore be seen as an abstract layer on top of the fundamental Blockchain technology, allowing end users to develop their own rules to define ownership rights, transaction formats and the outcome of state transition functions. One may achieve the aforementioned through smart contracts which require a certain set of parameters or conditions to be met prior to the execution of cryptographic rules.

The Ethereum state is composed of objects known as accounts, each of which is uniquely identifiable through a 20-byte address and their respective state transitions [18]. State transitions pertain to the direct transmission of digital value and information between accounts. Each account holds four fields; i) **Nonce** relating to the quantity of transactions issued by an account or new contracts established by the account, ii) **Ether Balance** defining the available owned Wei balance for an account, iii) **Contract Code Hash** pertaining to Ethereum Virtual Machine (EVM) hash code which is performed once an address receives a message call and iv) **Storage** corresponding to the 256-bit hash of the root node of the Merkle Patricia tree - the data structure used for storing all key-value pairs within the whole Ethereum domain.

Accounts fall under externally owned accounts (EOA) or contract accounts (CA), with the former being controlled via private keys and the latter through their respective contract code. EOAs may send messages by establishing and signing a transactions. On the other hand, since CAs contain code, receiving a message triggers the codes activation which effectively allows for reading and writing to internal storage, sending other messages or possibly creating new contracts.

Messages in Ethereum, although similar, differ from Bitcoin transactions in three different aspects; i) Ethereum messages may be formed through an EOA or CA whereas Bitcoin transactions are limited to external creation only, ii) Ethereum messages possess a specific field for data and iii) once a CA receives a message, a response may be provided through a defined function (if specified). Transactions in Ethereum are referred to as a signed data packet, containing the message to be transmitted from the origin - an EOA. Transactions contain 6 fields in total; i) Recipient, ii) Sender, iii) Quantity, iv) Data, v) Startgas and vi) Gasprice.

The Recipient and Sender fields define who is receiving or sending the transaction, respectively. Quantity relates to the value of Ether to be transferred; while data, unless otherwise specified, is by default defined as null. To avoid and ensure that no infinite loops are present within any defined code, resulting in a network meltdown, the two fields Startgas and Gasprice must be specified. Each constructed transactions is expected to construe a limit on how many computational steps are to be carried out along with any additional messages which may also be spawned upon execution. With Startgas defining the limit and gasprice the fee the sender is expected to pay the miner for each computation stage, this issue is catered for. In the event that a transaction has utilized all the gas delineated, any state changes are reverted excluding the payment of the fees.

Transaction fees double up as a regulatory mechanism, ensuring that the network is not abused, since issued transactions must be downloaded and verified via consensus [47]. Using the 'proof-of-work' concept, miners are rewarded for a probabilistic computational puzzle once completed through a series of hash function implementations. Requiring monumental amounts of computational power, a valid solution for the enlisted problem is relatively probabilistic, with a miners probability of finding a solution proportional to the miners computational power in the network. To increase the probability of rewards received while reducing the high variability in reward when mining alone, miners may also join mining pools having one designated pool operator responsible for distributing the sub-problems among members. The most prominent 60 miner pool are said to provide 90% of the total available power [33].

2.1.3 *Blockchain as Big data and potential applications*

Big data encapsulates a new technological paradigm pertaining to the high velocity, volume and variety (3 Vs [44]) with which data is being generated [45]. Without being limited to a specific domain, big data has the potential to revolutionize the operational procedures, such as key business performances, carried out in various industries. With such large quantities of data, one needs to deal with a number of issues such as data management, data quality and data security. Blockchain is able to resolve these issues by saving all data in a structured, encrypted and immutable format.

It's ability to maintain accurate data has led to it's applicability in Artificial Intelligence (AI) or Machine learning applications. Conventional analysis methods are limitted in use given the quantity of data to be analysed. Bitcoin has shown a large increase in size from it's inception, reaching an approximate size of 210 gigabytes as of the beginning of April 2019 [15]. This growth is also evident in other Blockchain protocols such as Bitcoin cash, Litecoin and Ethereum[1].

The data trust issues previously encountered by Internet of Thing (IoT) developments will be suppressed through the adoption of Blockchain services; leading emergence of a various IoT applications [67]. Examples include the authentication of interactions among connected vehicles and roadside infrastructures [29] as well as tracing and authenticating the source of goods [70].

2.1.4 *Feature set construction*

The accuracy of developed ML or AI models is heavily dependent on the dataset on which they are trained. A poor dataset containing irrelevant features would have a negative impact on the overall accuracy attained by the respective model. Finding a good data representation is ultimately domain specific and dependent on the available measurement values [37].

In several cases human expertise is often necessary to transform 'raw' data available into a set of useful features. One may also enhance the available features by extracting new features, through the combination/calculation of the acquired features, which may provide further insights on any patterns present within the data. This procedure is not only time consuming but also requires a significant level of domain knowledge in order to extract relevant features.

2.1.5 *Tree models - Decision trees, Random Forest and XGBoost*

Tree boosting has demonstrated its effectiveness in suitable machine learning applications ranging from smart spam classifiers to the de-

1 https://blockchair.com/ethereum/charts/Blockchain-size

tection of anomaly events in experiments from which new physics may emerge [19]. One can attribute the success of such applications to a combination of two key factors; powerful statistical models and learning systems which are able to scale and learn the relevant model of interest from vast datasets.

Tree models speculate that the relationship present between the predictors and the response may be defined locally through constant basis functions [57]. Dividing the input space into M disparate regions $R_1, ..., R_M$, each region is fitted with a constant function. The corresponding basis functions, also known as indicator functions $\phi_m(x) = I(x \in R_m)$, when combined form a tree model defined as

$$f(x) = \sum_{m=1}^{M} \theta_m I(x \in R_m) \tag{2.1}$$

for which θ_m corresponds to the constant fit for region R_m.

XGBoost, a decision-tree ensemble Machine Learning (ML) algorithm utilizes the gradient boosting architecture. The evolution of the XGBoost algorithm, for which we provide an explanation, may be seen in the figure below;

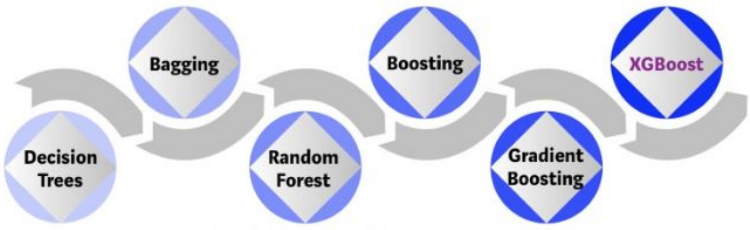

Figure 2.1: Evolution of Decision Trees [3]

Its foundation stems from the well-known structured *decision tree* methodology. A decision tree, taking into consideration the context to be addressed, designates a probability value to each valid and potential decision to form a so called 'decision-making device' [49]. In mathematical terms, $P(f|h)$ where f defines an element in the future vocabulary (set of available options) and h corresponds to the elements history (with regards to the current circumstance or background). Asking a certain series of questions relevant to the context $q_1, q_2, q_3, ..q_n$, we are able to determine the probability $P(f|h)$. Each question i is fully dependent on the results attained up to and including the $i-1$ position. The size of the decision tree to be formed is determined by the size of the training data available, formed by binary questions. Through the use of a classification tree, a question which constitutes k values is then decomposed into a series of binary questions on the respective k values. Following the order of these k questions in a stepwise manner, from the root node to the determined leaf node, we are then able to determine the correct label/definition of an unclassified object.

Bootstrap aggregating or *bagging*, similar to boosting, exploit the training dataset available so as to generate varying classifiers [66]. In simple terms, bagging generates diverse training sets through sampling with replacement from the available training examples - also known as bootstrapping.Therefore, provided a predetermined number of trials $t = 1, 2, .., T$, new training sets, the same size as the original available training set, are sampled. Due to sampling by replacement, these newly generated training sets at each t may contain multiple instances of the same example or none at all.

Using the training set samples, the learning systems allows for an ensemble of classifiers C^t to be generated. Aggregating the generated T classifiers, the final classifier C^* is developed. As a result, once a new instance x is provided for classification, each classifiers output $C^t(x) = k$ is recorded - resulting in the final class output $C^*(x)$ determined through a voting scheme (majority voting [43]). Situations which involve two or multiple classes obtaining the same number of votes may be solved arbitrarily.

The next essential layer, *random forests*, alters the approach by which the decision or regression trees are constructed. Introducing itself as a supplementary layer of randomness on top of bagging [46], each tree is assembled through a varied bootstrap sample of the dataset. Whereas in the traditional decision trees the variable on which a split occurs is determined through the largest contribution of information gain for all features, each tree is provided with only a bootstrap random subset of the available features on which the best split is carried out. This retrospectively counteracts issues such as overfitting - an issue commonly encountered and requiring attention in the machine learning domain.

One may compress the algorithm, both in classification and regression terms, into three key stages. Initially n_{tree} bootstrap samples are acquired from the available training dataset. For each respective sample an unpruned classification or regression tree is built, taking into account the following alteration; at each presented node instead of selecting the optimal split among all predictors, the predictors are randomly sampled m_{try} from which the optimal split is elected. Bagging is applicable whenever the size m_{try} is equal to the number of predictors p - an exclusive case. Last but not least, the final output provided new data is obtained by aggregating the predictions provided by each n_{tree}. In classification, this is a majority vote as opposed to averaging in regression.

Boosting was later introduced and amalgamated with random forest implementations. A variant of the committee method, boosting is comprised of training multiple 'base' classifiers in a sequential manner [14]. The error function corresponding to the training of a specific models fully depends on the performance of all the previous models in the 'chain'.

The intuition behind boosting stems from the weighting coefficient assigned to each available instance which is determined by the performance achieved by previous base classifiers on each respective instance. Specifically, instances which have been misclassified by one of the classifiers are assigned a larger weight. This will have adverse affects on the next available classifier in the sequence. A visual representation of the boosting mechanism is available in Fig. 2.2.

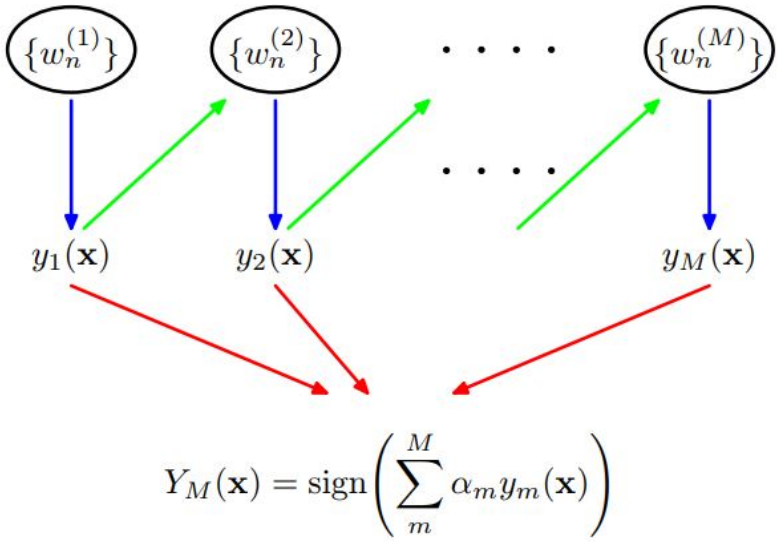

Figure 2.2: Boosting architecture [14]

Each base classifier $y_m(x)$ is trained on the presented weighted form of the training set (depicted via the blue directed arrows). The respective weights $w_n^{(m)}$ depend directly on the prior base classifier $y_{m-1}(x)$ (visually represented by means of the green arrows). Once all base classifiers are successfully trained, the final classifier $Y_M(x)$ is formed through a combination of all these trained base classifiers (seen graphically from the red arrows). Adaptive boosting, in short *AdaBoost*, is a widely used implementation of the boosting algorithm available on the majority of statistical computing languages.

The next improvement was the introduction of *Gradient Boosting*, which may also be considered as the implementation of gradient descent algorithms through the use of boosting [36]. Given the loss function, gradient boosting seeks to minimize the overall error through measuring the local gradient of the loss functions for the available set of parameters, iteratively tweaking them in the direction of the descending gradient. Once the calculated gradient is zero, the target global minima has been reached. Issues such as local minima [80] and plateaus [53] are a common threat, often giving the impression that the optimal solution has been reached.

The ideal loss function ultimately depends on the scenario but this freedom does not restrict the ability for gradient boosting to be performed as long as the loss function is differentiable. A variety of

parameters help reach the ultimate goal, as well as overcome challenges such as local minima and plateaus, one of which is the 'learning rate'. The learning rate has a direct effect on the step size. If set too small, the algorithm will require too many iterations in order to find the global minima exhausting available resources and time. If too large, one might risk surpassing the local minima resulting in a worse outcome than the previous step.

Short for Extreme Gradient Boosting, XGBoost offers a scalable and efficient implementation of the aforementioned gradient boosting framework [20]. In particular, XGBoost offers a number of advantages, some of which include: (i) The ability to address missing values present within the training set, (ii) scale to the demands of the dataset size (iii) offer an improved computationally effective alternative to that provided by gradient boosting algorithms, (iv) known to perform well in a number of varied competitions and studies [75] and (v) rank the features by order of their importance. During a machine learning competition made available by Kaggle in 2015, XGBoost was used throughout more than half of the winning solutions [82].

These benefits are mostly attributed to a number of system and algorithmic optimizations [3]. System improvements include;

1. **Parallelization:** Whereas traditional decision trees are built in a sequential manner, XGBoost parallelizes this task across multiple threads. This has a direct positive effect on the overall computation time.

2. **Tree Pruning:** Gradient Boosting Models (GBM) utilize a greedy stopping criterion when determining/computing the features to be split on (tree splitting). The negative loss criterion is also taken into consideration during the splitting stage. Contrary to this, XGBoost prunes backwards - using the 'max-depth' parameter first to limit the tree depth instead.

3. **Hardware Optimization:** The algorithm constructed in such a way that the hardware resources in an efficient manner. This is mainly attributed to the smart utilization of cache for storing gradient related data.

Algorithm enhancements are comprised of;

1. **Regularization:** To avoid unwanted model overfitting, more complex models are penalizes through L1 and L2 regularization.

2. **Sparsity Awareness:** By nature, XGBoost automicaly caters for different variations in sparsity patterns present within the dataset in an efficient manner.

3. **Weighted Quantile Sketch Algorithm:** The algorithm is employed in order to acquire the best splitting points with respect to the weighted dataset.

4. **Cross-validation:** The XGBoost implementation comes ready with cross-validation, enhancing the models' capability and reducing development time.

2.2 LITERATURE REVIEW

2.2.1 *Global economic crime*

Fraud has strengthened it's position as a considerable threat every year, affecting millions of people across the globe. A recent global economic crime survey recently conducted by PricewaterhouseCoopers in 2018 [65] established that close to half (49%) over the questioned 7,200 companies had been the victim of a fraud attack. Experts from HSN Consultants stated that online credit card fraud shall reach if not exceed $32 billion in the year 2020, with fraud in 2015 reaching $21.84 billion [2]. Visa, one of the worlds leaders in the domain of digital payments stated in 2016 that approximately 150 Million transactions occur every day over their network which translates to 1,667 transactions per second [78]. Mobile payments, facilitated through the development of mobile wallets, also contributed towards a total of $194.1 billion in 2017 [62]. Along with the main implication of financial loses, fraud has a negative effect on customer loyalty as well as converting physical to digital assets and vice versa. This is evident when 20% of clients affiliate with another bank after being a victim of fraud.

Manually reviewing transactions and business activity is still a widely adopted practice for the detection of fraud. Based on an annual Fraud Benchmark report carried out by CyberSource [58], 79% of North American businesses conduct manual reviews with 25% of these businesses on average manually reviewing executed orders. Nevertheless, following this manual review process, 89% of these orders were formally accepted by the respective businesses. This implies that in the majority of cases there is no need to conduct a review of the order - resulting in a waste of employee resources, money and time.

The process of automatic screening could drastically diminish the human resources necessary, limiting human intervention to cases which are highly suspicious [50]. To address such issues and hinder fraud related issues and other online crime, financial institutions have implemented various tools such as real-time credit authorization, address verification systems (AVS), card verification codes etc [61]. These technologies are heavily based on learned patterns and criteria, which possibly limits the ability to detect newly orchestrated attacks.

2.2.2 *Illicit activity over Blockchain networks*

Due to Blockchain protocols such as Bitcoin and Ethereum offering pseudonymity, cyber-criminals have picked up this technology as a

viable alternative to conduct essentially 'untraceable scams' [12]. A number of mechanisms have been introduced to reverse engineer ones identity [51, 68] on the Bitcoin network. This has been contradicted by effective techniques to retain ones anonymousness [6], which together implies that criminal behavior may be easily executed over the Bitcoin network while still being hard to identify.

Criminal activity over Ethereum comes in various forms, some of which include the classic ransomware approach, money laundering and Ponzi schemes [22]. Some of these activities utilize the underlying Ethereum smart contract technology in order to develop these so called Smart Ponzi Schemes. Since these smart contracts offer the automatic enforcement of executing computer programs without the inclusion of any trusted authorities [74], fraudsters are subjected to a number of attractive features;

1. **Anonymity:** The creator of the Ponzi scheme itself can remain anonymous since one is not required to reveal their identity when withdrawing or creating the contract itself.

2. **Unstoppable:** Once deployed, smart contracts cannot be terminated by a third party/central authority.

3. **False sense of trustworthiness:** Due to the fact that smart contract code is public, immutable and automatically enforced, investors may believe that their funds are safe and that they cannot be taken advantage of - and hence they are led to believe they have an equal opportunity of receiving interest on their investment.

Ponzi schemes have proliferated on the Ethereum network, disguising itself as 'high-yield' investment program in which investors are repaid through new users investments. This, coupled with peoples interest in participating in the Blockchain technology even though they lack the necessary knowledge to understand how it works, results in the successful fruition of such scams.

This may work well for the initial investors but the program implodes once no new users invest. Although it is difficult to provide a specific value on the whole economic impact attributed to such activities, some Ponzi schemes performing on Ethereum have accumulated over $43 million worth of Ether [63]. Similar activity is also present on other Blockchain protocols; with $7 million gathered by scams between 2^{nd} September 2013 and 9^{th} September 2014 [76] on the Bitcoin network. DappRader, providing insights and information on available distributed applications, in 2018 claimed that over a 24 hour period two of the three most popular Ponzi schemes (FOMO 3D and PoWHD3D) saw a trading volume of 20,000 ether (which at the time translates to $9 million) [27]. A well known Smart Ponzi scheme

'Rubixi' back in 2016 used a propaganda-like text to entice end users, stating;

```
"Hello! My name is Rubixi! I'm new & verified
pyramid smart contract on the Ethereum Blockchain.
When you send me 1 ether, I will multiply the
amount and send it back to your address when
the balance is sufficient. My multiplier factor is
dynamic (min. x1.2 max. x3), thus my payouts
are accelerated and guaranteed for months to
come"
```

Initially large payouts were promised but this soon not the case, since in less than a month time the contract received no further deposits following a bug being discovered in the contract code [13]. Only 23 of the 98 users who invested were paid, with the contract retaining a balance of 4 Ether.

Another smart contract in the form of a game, 'King of the Ether Throne', forces players to compete in order to acquire the respective 'King of the Ether' title [11]. In order to receive this title one must pay a claim price of 50% on top of the last price from which the current 'king' is provided a compensatory fee (99% of the claim price) along with a small fee (1%) to the contract owner [64]. Although relatively intriguing to new players, a development issue present in the smart contract solidity code (insufficient gas for miners to verify the transaction) caused unsuccessfully processed wallet contract payments made by the contract to be sent back to the contract itself. Therefore new kings were not compensated in away way.

Kaspersky (an anti-virus company) announced that phishing, in the second quarter of 2018, raked in $2.3 million for cybercriminals by exploiting ICOs [77]. These online criminals mimic new Ethereum-based startup companies websites - creating a deceptive and fake ICO website. In some cases, phishing sites appear before the project site has been deployed, manipulating potential investors into believing they are investing in the authentic ICO. An example of such behavior was witnessed during the Experty phishing attack, resulting in a total loss of $150,000 worth in Ethereum [35]. Potential investors were required to go through a registration procedure on their website in order to partake in the ICO. Scammers managed to steal the credentials of potential investors, later persuading them via email to transfer Ether through a fake invitation link.

Cybercriminals also utilize various social media platforms to expand their potential pool of users by created fake accounts to impersonate celebrities such as Elon Musk, Vitalik Buterin and Pavel Durov and so on [77]. As stated in Fig. 2.3 below, a fake account impersonating Vitalik Buterin (the creator of Ethereum) was set up on Instagram offering a giveaway once 0.5 Ether was sent to the provided address.

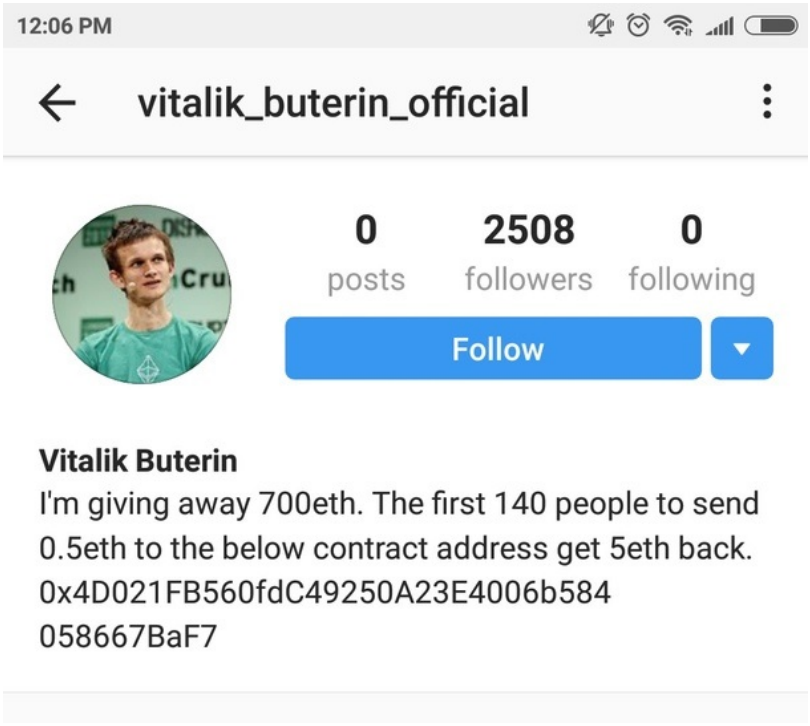

Figure 2.3: Fake Vitalik Buterin Instagram account [77]

2.2.3 *Approaches to combat illicit behaviour over Blockchain networks*

Illicit behavior carried out over Blockchain networks using cryptocurrencies heavily impacts the trust potential and current users have within the system. Stopping such anomalous behavior is vital for cryptocurrencies as this would help instil the idea of being a legitimate alternative to the current traditionally used fiat currency [52]. This can only be done if this behavior is detected in the first place.

In order to combat such behavior, a number of authorities and organisations look towards the implementation of systems utilizing machine learning techniques capable of detecting and alerting the necessary personnel. As explained in section 2.1.3, the volume of data generated along with the resources required to sift through such information naturally suggests the implementation of machine learning techniques to help identify any patterns related to anomalous behavior.

The problem is initiated by a set of three issues; i) Which are the most important features/dimensions (possibly combinations of) to distinguish normal from fraudulent behavior, ii) how should one create a 'profile' using the relevant features and iii) given the 'profile', when and how should you declare and prove the respective behavior [31].

Given that some of the largest cryptocurrencies have publicly available Blockchains, the data for such analysis may be easily gathered

following a well defined data acquisition and storage pipeline. Numerous approaches and implementations have been put forward in order to differentiate such activities over Blockchain networks, most of which includes supervised and unsupervised machine learning techniques.

In a study conducted in 2013 by Hirshman et al [39], the researchers were interested in three main criteria; i) clustering the available data to enhance data exploration, ii) detecting any money laundering or money mixing attempts as well as iii) tracing back to the origin of the mixing service transactions. Therefore, having acquired all the transactions up until May 2013 from the inception of Bitcoin, and filtering users with a minimum of 650 transactions, the authors were able to identify users which carried out transactions in an atypical manner - behavior typically associated with money laundering. Using two primary machine learning methods, K-Means and RolX [38], a subgraph of 6,058 hubs representing unique users was generated. Having extracted 15 features relating to the interaction and transaction values enacted by the stated accounts, 1000 iterations of various cluster sizes of K-means was performed from which $k = 5$ was found to be the optimal number of groups after calculating the error. The error corresponded to the sum of the distances from each point to its corresponding centroid. The most important identified features used to determine the clusters were;

1. The mean node degree of other users transactions were carried out with.

2. Variance of node degree of other users with which transactions were executed.

From the 5 main clusters identified, 2 of the least common had a large variance in the transaction amount. This implied that these account participated in transactions worth small values up to thousands of bitcoins. RolX (Role eXtraction), an unsupervised algorithm used to classify nodes of a graph into diverse classes, was used to expand upon these anomalous accounts identified by K-means. An example of one of the identified money mixing behavior may be seen in Fig. 2.4, highlighting how the bitcoins are recursively split and directed to two new addresses.

Another large contribution in the detection of illegal activity relates to identifying and detect Ponzi schemes deployed over the Ethereum network as smart contracts. Comparing a number of well known classification models known to perform, W. Chen et al classified smart contracts using combinations or isolated categories of features [23]. Seeking to identify the discriminative power of such combinations, three features were utilized; i) account features, ii) code features and a iii) combination of the two. 13 key account features were extracted along with 64 unique opcode features.

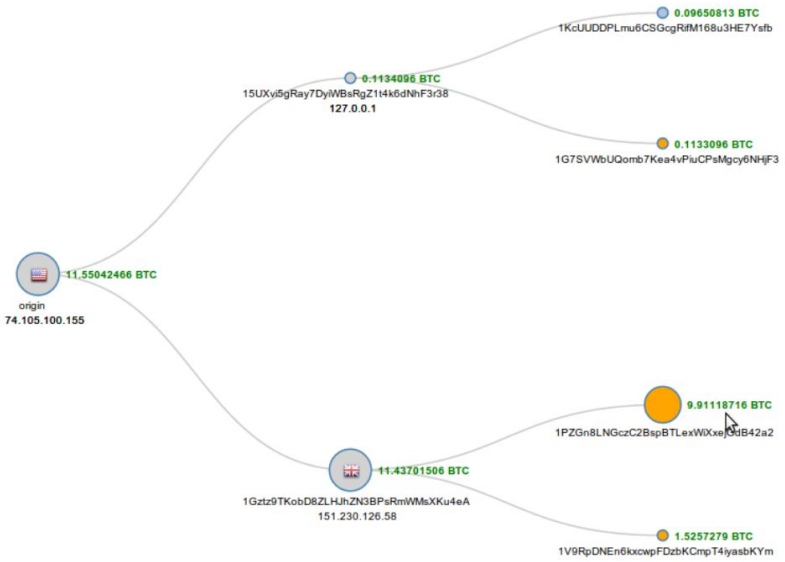

Figure 2.4: Money mixing example from one of identified accounts using Bitcoin [39]

The smart contract operation code reflects the activity and logic behind the smart contract from the EVM perspective [9]. Since in most cases the source code for most smart contracts is undisclosed, 394 carefully selected smart Ponzi schemes were identified from a subset of 3,780 open source contracts manually - allowing for the creation of a labelled dataset to be used for the classification models.

Following an evaluation of the said models with parameter tuning, Random Forest produced the optimal results in this two-class classification problem using a combination of the two features available.A word cloud graph depicting the variation in opcodes pertaining to the ponzi schemes and non-ponzi schemes was produced, and may be seen in Fig. 2.5. From analyzing the cloud graph, the authors note that the least used opcode in Ponzi scheme contracts, 20 instances in the analyzed 200 Ponzi scheme contracts, was CREATE. This in turn makes sense since the aforementioned opcode is utilized when creating a new account along with some associated code, and Ponzi schemes have almost no reason to create new accounts.

Although a number of developed smart contracts are deployed in an experimental stage, possibly affecting the overall performance of the model produced, the model classified 305 on 394 of the identified smart ponzi schemes with a probability over 90%. They estimate that over 500 smart ponzi schemes are available on the Ethereum network, taking into account the recall and precision achieved by their model.

A recent study conducted in August 2018 sought to detect fraudulent activity on the Ethereum network. O'Kane analyzed the Ethereum transactional data over a time frame [60], specifically between the 1^{st} and 10^{th} January 2018. Specifically, this period was chosen primarily

Figure 2.5: Opcode cloud graphs relating to ponzi scheme (left) and non-ponzi scheme (right) contracts [23]

for two reasons; the 4^{th} of January had the highest number of transactions over 24 hours as well as the price of Ethereum peaking during the month of January which would relate to a possible large variety in the transactions to be evaluated. The large number contributed towards an increase in exchange transactions where investors traded in their Ether for any other desired fiat currency.

Having utilized the JSON-RPC interface, the sender's and receiver's addresses were attained along with the gas price and gas used to carry out these transactions. 4000 scams available on Etherescamdb along with token addresses and exchange addresses were combined with the transactional data for analysis. Throughout the acquired period of interest 89 transactions were identified as scams; with either the sender or the receiver of a transaction corresponding to a scam address. Specifically, these fell under the phishing scam category with the most prevalent phishing site being that of a OmiseGo site asking for private keys. In less than 36 hours, between 5:50pm on the 9^{th} and 11:30pm on the 10^{th} of January, 74 addresses had been affected.

Using principal component analysis (PCA), K-means and random forest, 13 clusters produced a maximum Davies-Bouldin Index of 81.2%. Nine features, having a mean decrease in Gini less than 600, were retained. The most important features have been defined in Fig. 2.6. The 'toAcc' feature, defining the account ID which sent the Ether, was identified as the most important feature. Using these results to detect anomalies in their data, 5 outliers were detected based on their Euclidean distance from the cluster centers.

T. Chen et al designed an approach (overview in Fig. 2.7) to collect all transactional data, construct three graphs to represent the major activities carried out over the Ethereum network and conduct exploratory analysis to gain insight from these graphs [21]. The weighted-directed graphs characterized the three main activities performed on the Ethereum network; money flow graph (MFG), smart contract creation (SCC) and smart contract invocation (SCI). A total of 25,502,131

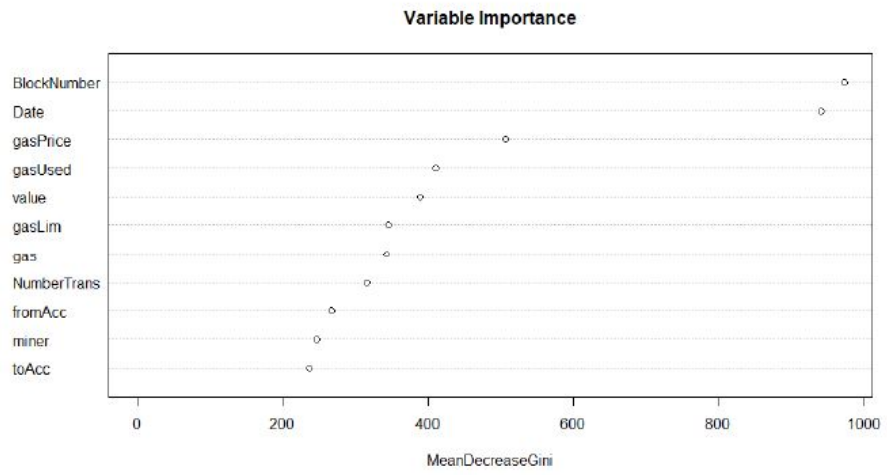

Figure 2.6: Variable importance based on Gini index [60]

external and 19,759,821 internal transactions were gathered from the launch of Ethereum till the 10^{th} June 2017; with external transactions executed by EOAs and also present on the Blockchain and internal transactions pertaining to transactions as a result of executing a smart contract but not available on the Blockchain. Such internal transactions were obtained by re-executing external transactions via their customized EVM.

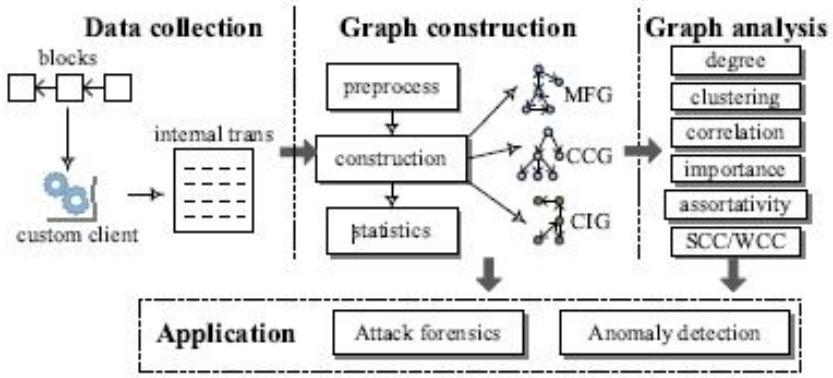

Figure 2.7: Overview of designed approach [21]

Following the parsing of external transactions and logs generated from the internal transactions, a total of 2,721,080 accounts were obtained; 2,121,146 EOAs and 599,934 smart contract accounts. Having analyzed the constructed graphs, three relatively insightful conclusions were derived at the time of writing;

1. Users prefer transmitting money on Ethereum as opposed to using smart contracts. This may be attributed to many users prior experience in other cryptocurrencies such as Bitcoin in conjunction with their shortcoming in smart contract use.

2. Smart contracts are still not commonly used throughout the Ethereum domain. This may be attributed to the lack of applications which may be converted and implemented using smart contracts, most of which are mainly financial applications.

3. The variability in Ethereum use among users suggests that most users may use Ethereum for testing purposes along with deploying toy contracts on it.

Along with a general analysis of such graphs, the results were also used in combination with a script capable of detecting abnormal contracts (defined as anomalies). This involved a correlation of the said three weighted graphs, return a boolean value to indicate if one of three threshold criteria have been surpassed or not. The authors classify a smart contract as abnormal if the account bears a large number of created contracts which are rarely used for money transfer or invocation. The authors highlight that the creation and deployment of smart contracts in a short period of time would have drastic effects on the propagation speed of blocks. This was the case in one of the accounts analysed who created 3898 contracts, 91% of which were created in a short period.

2.3 RESEARCH CONTRIBUTION

This work aims to provide a meticulous approach towards distinguishing abnormal or anomalous activity enacted by accounts over the Ethereum Blockchain network. In retrospect, the same implementation may be applied to other cryptocurrencies utilizing the Blockchain protocol. While the exact implementation may differ due to the available network APIs and protocols required to connect to the network, the same procedure may be applied.

After scouring the academic domain for similar implementations, the proposed work seems to form part of a unique niche. The majority of the evaluated research utilize unsupervised learning methods such as K-Means and Support Vector Machines (SVM) to detect any outliers or abnormalities available within user behavior. A limited amount of research surrounding this area of interest have been conducted using supervised learning approaches; with the ones conducted using variations of decision tree algorithms. Furthermore, surrounding the same topic of detecting abnormal behavior specifically with respect to the Ethereum network, research has been made in the detection of smart contract ponzi schemes. This effectively could work hand in hand with the proposed research by combating the development of illicit activity from multiple fronts.

The closest work identified was carried out back in October 2017 with a specific interest in the detection of suspicious behavior over the Bitcoin network [42]. Similar research was carried out, using graph extracted features in combination with Random Forest to classify an accounts activity. Our research improves on the Random Forest algorithm by utilizing the superior XGBoost algorithm while also extracting other relevant features defined in Table. A.1. At the time of writing this report, no similar work has been carried out on the Ethereum network - specifically analyzing the network at an 'account level'.

CHAPTER THREE

3

METHODOLOGY

In this section we provide an in-depth explanation and justification for the selected building blocks forming the final architecture; starting from the data acquisition stage up until the final generated results produced by the XGBoost model implementation.

The envisaged dataset required a combination of illicit accounts flagged by the Ethereum community and normal accounts which have not been identified for carrying out any illicit activities or transactions over the network. Therefore, we define the developed data pipeline used to acquire the desired dataset. The process, visually represented in figure 2.1, is initialised by two 'streams'; for normal and illicit accounts. Once the accounts had been obtained, an Etherscan application programming interface (API) was utilized to retrieve the transactions executed for each accounts from which we extract a number of features to enhance the dataset and pass to the classification models; Random Forest and XGBoost.

All development was carried out on a core i7-7700HQ 2.8GHz (up to 3.8GHz) with 16GB available RAM using Python v3.6.

3.1 ACQUIRING THE ETHEREUM ACCOUNTS DATASET

3.1.1 *Hosted vs Local Go Ethereum node*

In order to connect to the Ethereum network, two options are available. One may either connect to a fully connected node through a third party API (such as Infura[1]) or set up a local node using a Go Ethereum[2] (GETH) client. Both options provide the same functionality and end results. Since the only difference is the location of the node connected to the Ethereum network, the JSON-RPC API calls remain the same and therefore the only change needed when switching between the two options for data retrieval was the URL endpoint.

For initial testing purposes, an Infura account was created - providing the MAINNET[3] endpoint so as to set up a connection. Although the initial set up is relatively straight forward, the query time is relatively slow (approximately 100 times slower) when compared to a local node as seen from Fig. 3.1.

This was deemed fine for testing purposes but in order to create a larger database containing around 7,700 records this needed to

1 https://infura.io/
2 https://geth.ethereum.org
3 The original and main network for Ethereum transactions

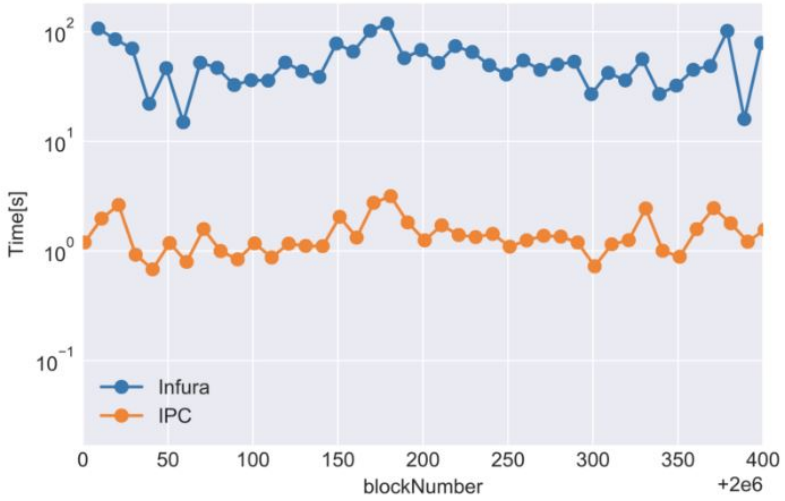

Figure 3.1: IPC local node vs. Infura Hosted node [73]

be improved. Being only part of the pipeline, execution time and data retrieval were kept in mind throughout the whole process for scalability purposes. Although this was/should not be the case, data was also being retrieved through a third party which may create a level of uncertainty on the legitimacy of the data acquired. These issues were circumvented when using the locally connected Ethereum node but at a cost of storage space (approximately 140GB). Naturally we were not interested in downloading the whole Ethereum network but a subset of the data where Ethereum was relatively active. This, based on a number of statistics provided by Etherscan[4], was deemed to be anything past June 2017.

A local GETH client, version 1.8.23-stable with Go Version 1.11.5[5], was installed and connected via terminal to the MAINNET using the set of parameters present in the script below;

```
geth --cache=8096
--syncmode "fast"
--shh --rpc
--rpcapi web3,eth,net,shh
--datadir "D:\Ethereum_Fast"
```

The cache parameter was increased for performance purposes to 8096Mb of ram. Ethereum offers three options to synchronise to the network; 'fast', 'full' and 'light' [40]. The 'full' node retrieves the block headers, block bodies and legitimizes every element. The 'light' synchronise surrenders old history, retrieving the current state. Essentially any element which needs to be verified must still be verified by a full node for any respective tree leaves. 'Fast' synchronise, similar to the

4 https://etherscan.io/chart/tx
5 https://geth.ethereum.org/downloads/

'full' parameter option, supports the acquisition of block headers and block bodies only processing the last few transactions. Hence, for the scope of the dataset requirements, the 'fast' option is sufficient since it obtains both the blocks and transactions, containing the respective account IDs executing such transactions using less computation and time overall.

The 'rpc' parameter enables the HTTP-RPC server for which we define the web3, eth, net and shh APIs for use. 'Datadir' defines the directory in which we would like the Ethereum blockchain to be stored or appended to locally (if part of the blockchain has already been stored in defined folder, the synchronisation process will continue off the last stored block). The list of all available command line options is available on the official go-ethereum github wikipage[6].

Once successfully initialised and connected to the network, the Ethereum blockchain is downloaded one block at a time starting from the genesis block (initial block).

3.1.2 *Random Selection of normal accounts*

The aforementioned procedure facilitated the ability to acquire transactions from an active period of the Ethereum network. Once the GETH client was running successfully, an Inter-process Communications (IPC) pipe is created automatically - which allows for another local connection to be set up with our GETH client.

The functional tool developed by Sokolowska [73] was utilized, capturing data from a passed endpoint (GETH node or Hosted node), parsing and appending to a database with 3 tables; transactions, blocks and quick-transactions. The original code[7] was altered to only retain transactions using the web3 API which provides the sender and receiver for each transaction.

Using the Web3 API and IPC pipe, the starting block number was specified along with a threshold number of blocks to store. Having identified block number 3,800,000, corresponding to a period of interest (mid-2017), approximately 5000 blocks were downloaded from which we are interested in selecting 3000 normal accounts at random. These blocks were then stored using the SQLite3 package in database (.db) format. The general process has been depicted in the flowchart Fig. 3.2. Having installed the DB Browser for SQLite version 3.11.1[8], the created .db file was imported and extracted as a comma separated values (.csv) file.

While this implementation may not be seen as the most efficient, the extraction of the .db file (approximately 100Mb) to .csv is executed

6 https://geth.ethereum.org/interface/Command-Line-Options
7 https://github.com/validitylabs/EthereumDB
8 https://www.sqlite.org/download.html

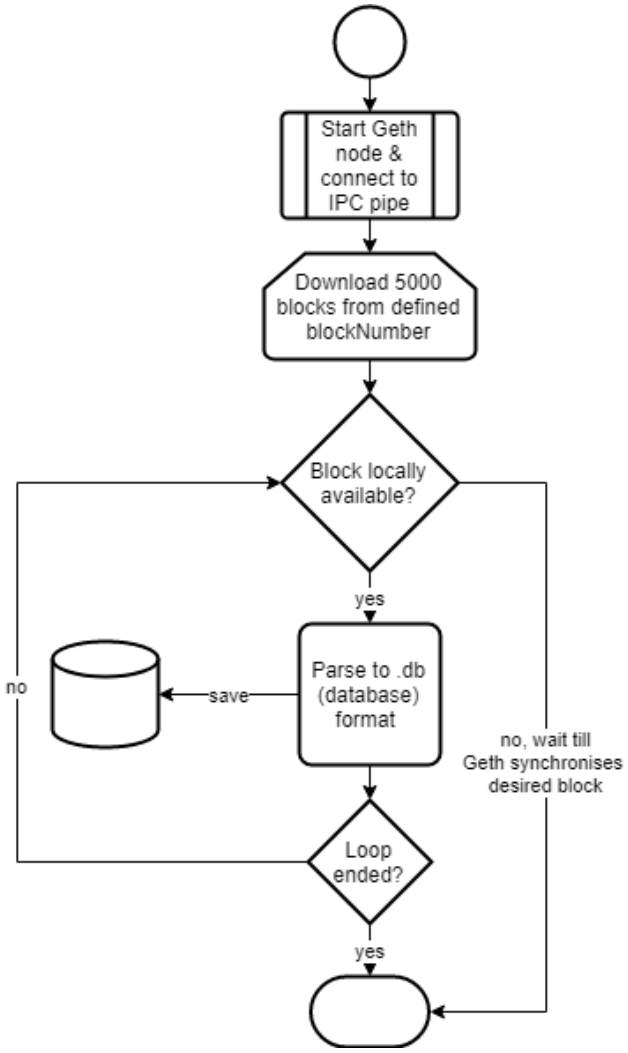

Figure 3.2: Flow diagram representing the acquisiton and parsing of Ethereum blocks to database format using the locally installed GETH client. The process is initialised using the command line argument specified in section 3.1.1.

in a matter of seconds and therefore serves its purpose. Furthermore, this procedure only needs to be carried out once.

Once the CSV file was generated, 3,000 random account IDs, either senders or recipients, were extracted from the available transactions. 2,502 accounts from the extracted accounts were unique and passed to the Etherscan transaction API prior to feature extraction. All accounts were cross-referenced with the accounts present in Fig. 3.1.3 to ensure that no documented illicit activity had been conducted by any of obtained accounts. These accounts were active during the period of mid-July 2017 - a period which saw an increase in Ethereum network usage.

3.1.3 *Illicit account IDs*

The MyCrypto team, together with the Ethereum community, work together to pinpoint any illicit activity being carried out by certain accounts - EOA or CA. Two separate repositories, both available online, were utilised [1, 55]. Updated on a daily basis, the majority of the addresses coincided with each other, resulting in a total of 2179 unique accounts.

Accounts were flagged for illicit behavior for a number of reasons, such as

1. Trying to imitate other contract addresses providing tokens
2. Scam lotteries
3. Fake ICOs
4. Imitating other users
5. Ponzi schemes
6. Phishing
7. Mirroring websites

All documented illicit accounts before the 17^{th} of April were collected after parsing the JSON formatted file obtained from Etherscamdb[9] via a GET Request and the Github repository downloaded JSON file[10], respectively.

3.2 ACCOUNT TRANSACTION API

Etherscan provides an API supporting the retrieval of the last 10,000 transactions carried out by an ERC20 account (Contract account following the ERC20 interface standard) or normal account. Paginated results, to further increase the threshold of retrievable transactions per account, is also feasible but as noted from the transactions obtained;

1. 10,000 transactions was sufficient enough to determine an accounts activity through the features extracted.
2. Execution time ranged depending on the accounts' activity. The more transactions an account performed, the longer the data retrieval time - ranging between approximately 1-25 seconds per account.

9 https://etherscamdb.info/api/scams/

10 https://github.com/MyEtherWallet/ethereum-lists/blob/master/src/addresses/addresses-darklist.json

3.2.1 *Normal Transactions*

The simplest form of transactions within the Ethereum network are between EOAs. These executed transactions involve the transfer of ether from the sender to the recipient. Other plausible transactions may involve contributing towards a Crowdsale of contract tokens in exchange for a quantity of the respective tokens [72]. In this respect, the value transferred is most of the time 0 and are referred to as functions for which the end user calls a 'claim' contract function to collect the due tokens.

Normal transactions contain an 'ISERROR' field which dictates whether a transaction was successful or not. An unsuccessful transaction may be due to many possible reasons, some of which include lack of necessary funds, gas limit exceeded or incorrect recipient account address. For the purposes of this research, we are only interested in the transactions which were successfully executed by ascertaining that the 'ISERROR' field is set to 0 and discarding the rest. The fields of interest, utilised for feature extraction have been defined in the table below;

Field of Interest	Description	Data Type
status	Determines whether account executed any transactions	Boolean (1 = Yes, 0 = No)
result	Wrapper containing all transactions. If 'status' field is '1' then transactions are available in 'result' field	JSON objects
to	Recipient account address for transaction	20-byte value
from	Sender account address for transaction	20-byte value
contractAddress	New address for contract created	20-byte value
value	Value in 'Wei' transferred	Long Integer
timestamp	Block timestamp containing transaction	UNIX time

Table 3.1: Normal transaction fields of interest

3.2.2 *ERC20 Transactions*

These transactions are solely related to the transfer of tokens which support/fall under the ERC20 interface standard. Therefore the transferable token must support these standards and only then are these tokens listed as transfer tokens. Contrary to the 'Normal Transactions', any listed transactions were successful and therefore no transaction filtering must be carried out. Transactions are obtained using the same JSON request format, for which we are interested in the following fields for feature extraction;

Field of Interest	Description	Data Type
status	Determines whether account executed any transactions	Boolean (1 = Yes, 0 = No)
result	Wrapper containing all transactions. If 'status' field is '1' then transactions are available in 'result' field	JSON objects
to	Recipient account address for transaction	20-byte value
from	Sender account address for transaction	20-byte value
contractAddress	Contract address	20-byte value
tokenName	Name of ERC20 transferred token	String/UTF8
tokenSymbol	Same as tokenName. Name of ERC20 transferred token	String/UTF8
value	Value in 'Wei' transferred	Long Integer
timestamp	Block timestamp containing transaction	UNIX time

Table 3.2: ERC20 token transaction fields of interest

3.3 FEATURE EXTRACTION

A total of 42 features were extracted per account using the transactions gathered in the previous section. A small subset of extracted features were similar to those acquired in similar work using graph analysis [21, 39]. Feature extraction was carried out in conjunction with the Etherscan transaction request API; generating the respective fields for each account and printing the row to a CSV file before moving to the next address available in the list. We have made the extracted dataset publicly available[11].

A complete list of the fields generated and available within the CSV file may be viewed in appendix Table. A.1.

3.4 DATA VISUALIZATION

Given that the task at hand contains 42 dimensions, a dimensionality reduction technique known as t-Distributed Stochastic Neighbor Embedding (t-SNE) was used for visualization purposes. The t-SNE is a non-linear method which supports the visualization of high dimensional data [48]. Effectively, this is done through mapping each data point to a position in a 2 or 3 dimensional space. In other words, a conversion is carried out from high-dimensional euclidean distances present among the datapoints to conditional probabilities representing similarities [7]. Therefore, given a high dimensional dataset $X = x_1, x_2, x_3, ..., x_n$ a conversion takes place such that the resultant dimensionality $Y = y_1, y_2, .., y_n$ may be visually represented through the form of a scatter plot.

The inherent aim is to preserve the structure present in the high-dimensional space when mapping to that of a low-dimensional space such that any potential insights are discovered. These insights would correspond to any relative similarities present in the accounts obtained. We implement and provide the resultant 2D and 3D t-SNE plots with respect to the account class label.

11 https://git.io/fjwFr

3.5 XGBOOST

Prior to model execution, the dataset (with the features extracted into a CSV file in the previous steps) needed to the be prepared. The subsets generated during the data preparation stage are utilized by the model for training and testing purposes, producing the resultant predictions and results. This was then followed by parameter tuning and optimisation in order to identify the optimal parameters delivering the best results given the resources available.

3.5.1 *Dataset preparation*

Two well known data preparation techniques were utilized to test the performance of the trained model; train-test split and k-fold cross-validation. Using the Flag as the target value, multiple train and test subset ratios were implemented with the models learning capability in mind. The stratify parameter was set to the output 'FLAG' variable to retain the ratio between the normal and illicit accounts.

Various configurations were also applied to the k-fold cross-validation, using 3-, 4-, 5- and 10- folds and the same stratify parameter set to the output 'FLAG' variable. The corresponding results have been provided in the results section. This was mainly done to identify if any overfitting or underfitting issues were present.

3.5.2 *Model Implementation*

XGBoost offers an easy to implement Python library, helping one achieve scalability, portability and accuracy. After installing the XGBoost package version 0.82, the library was imported into the Python project for use.

The model had to undergo the following procedure in order to produce the desired output;

1. Declare a XGBoost classifier object (and all the corresponding parameters to be considered for model execution)
2. Train the defined classifier object on the training set defined during the data preparation step above
3. Utilize the trained model to predict on the corresponding test set and output results

3.5.3 *Parameter optimization*

K-fold cross-validation is a well known model evaluation technique, allowing one to determine the optimal model parameters by splitting the available dataset into k-folds for training and testing purposes.

Various number of folds were assessed; 3-, 4-, 5- and 10- folds with the stratified sampling parameter set to the output 'FLAG' variable.

Three model parameters were evaluated; i) learning rate, ii) n_estimators and iii) max_depth. Many more XGBoost parameters may be explored but from the reviewed literature these were sufficient.

These configurations were assessed in tandem with grid-search cross-validation - an exhaustive parameter tuning method which identifies the optimal parameters for the respective models context. Due to grid-search cross-validation being computationally heavy, parameters were not assigned large ranges (also as a result of the limited computational and time resources available).

3.5.4 *Performance measures*

Two primary measures have been employed to determine the relative performance of the XGBoost model with distinctive parameters. Their respective outcomes provide no relative or direct information on the other measure; with the Area Under the ROC Curve (AUC) used to (help) identify the optimal model parameters whereas feature importance allow one to understand the relative features importance and model stability. We also provide an explanation of other measures used to further determine the models performance.

3.5.5 *ROC curve, AUC and Confusion Matrix*

To help identify the optimal parameters for the model in question, we utilize the Receiver Operating Characteristic (ROC) curve which presents us with the AUC measure in order to help determine the classifiers performance [16]. Classifiers during the testing produce a measurable quantity of instances which have been assigned a correct and incorrect class. These are usually documented in the form of a confusion matrix; listing the number of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). Although these measures provide an indication of the models performance, more meaningful measures may be formulated to highlight the performance of the model in specific scenarios. Sensitivity, specificity and accuracy are among these measures along with the cost of misclassification.

Accuracy is one of the simplest measures to determine a models performance. In simple terms, its the number of correct model predictions with respect to all the available examples being tested.

$$Accuracy = \frac{TP + TN}{CP + CN} \tag{3.1}$$

CN is the sum of $TN + FP$ while CP is $FN + TP$.

Sensitivity allows us to derive the proportion of real positives which have been correctly identified. It is also commonly known as recall

and True Positive Rate (TPR). The TP and TN in the given situation are the 'illicit' or 'normal' accounts [4], respectively.

$$Sensitivity = \frac{TP}{CP} \tag{3.2}$$

All three measures produce a value ranging between 0 and 1.

$$Specificity = \frac{TN}{CN} \tag{3.3}$$

Using the specificity to calculate the FPR which is equal to $1 - Specificity$, we are able to plot the ROC curve and obtain the AUC measure [56]. The ROC represents the measure of separability, expressing the models capability of differentiating between classes. XGBoost provides the probability that an account forms part of the illicit class. By varying the probability threshold with which a class output is determined, variant pairs of sensitivity and specificity are generated, which we use to build an ROC curve and compute the corresponding area under curve (AUC) [24]. The higher the AUC is the better the accuracy of our model is.

Given the risk attributed to the incorrect classification of accounts as well as its ability to encompass all criteria surrounding the models performance, we deem the AUC as the preferred measurement for identifying the optimal model of XGBoost with different configurations.

3.5.6 *Feature Importance*

XGBoost provides 3 main importance metrics on which the importance of all features trained on by the model are quantified and presentable. The resultant output is a matrix in the form of a table; with the first column listing the features and the other columns defining the importance value with regards to the importance metric provided. The aforementioned include;

1. **Weight** specifies the number of times a particular feature crops up in the trees defined by the model
2. **Gain** corresponds to the relative improvement in accuracy achieved by a feature to the branches it forms part of.
3. **Cover** or coverage relates to the number of observations with respect to the feature in question

Since we are also interested in the overall model rigidity/stability, we also look at how the model performs when provided with varying train/testing sizes. Therefore, using the same four train/test splits

defined in 3.5.1 we analyze how the features vary in importance with respect to the 3 importance metrics over an average of optimal folds. Although we expect some sort of variation to be expressed in the order of the features importance, the most stable metric would be that which retains a relatively similar order of features. More specifically, we consider the top 10 features to be the most valuable.

CHAPTER FOUR

4

RESULTS

The following section delivers the results obtained from the implemented methodology. Split into sectors, the results are available in the order by which the implementation is to be assessed; starting with some general dataset information, non-linear dimensionality reduction t-SNE method, XGBoost model evaluation and the most important XGBoost features.

4.1 GENERAL DATASET INFORMATION

Following feature extraction on the available accounts, we provide some general information on the complete generated dataset. Specifically we provide insights on features we deem to be of interest.

The bar chart present in Fig. 4.1 highlights the average time between the first and last transaction executed by an account with respect to the relative account FLAG. This in turn relates to the average duration of activity for the respective accounts.

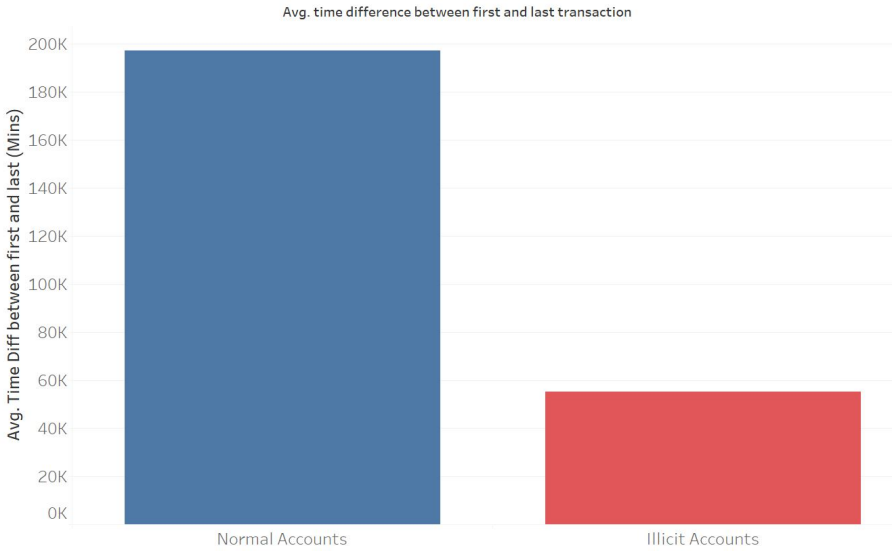

Figure 4.1: Average time between first and last transactions (in minutes) with respect to the account class ('normal' or 'illicit'). Normal and illicit accounts marked in blue and orange, respectively.

To highlight the large disparity present in an account activity, Fig. 4.2 and Fig. 4.3 capture the discrepancy in average Ether sent or received as well as the average number of transactions sent or received per account, respectively.

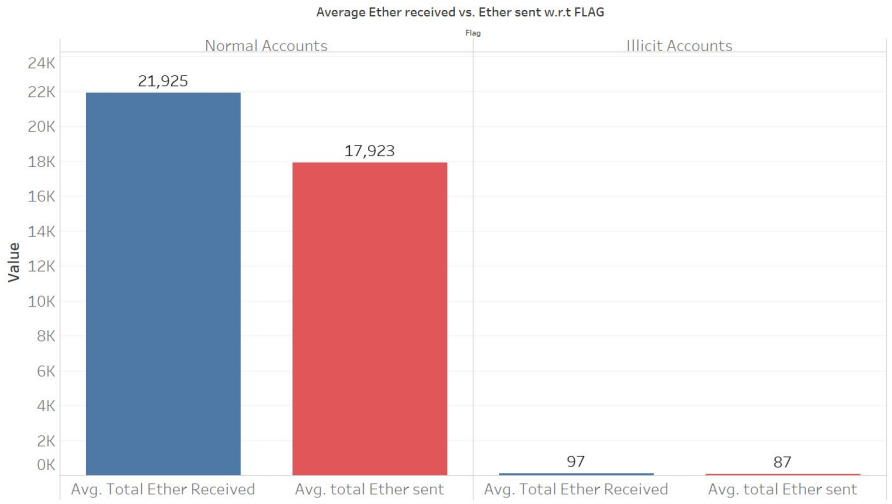

Figure 4.2: Average amount of Ether received and sent with respect to the account class.

We note that on average normal accounts are active for a longer period, send and receive a larger quantity of transactions along with Ether.

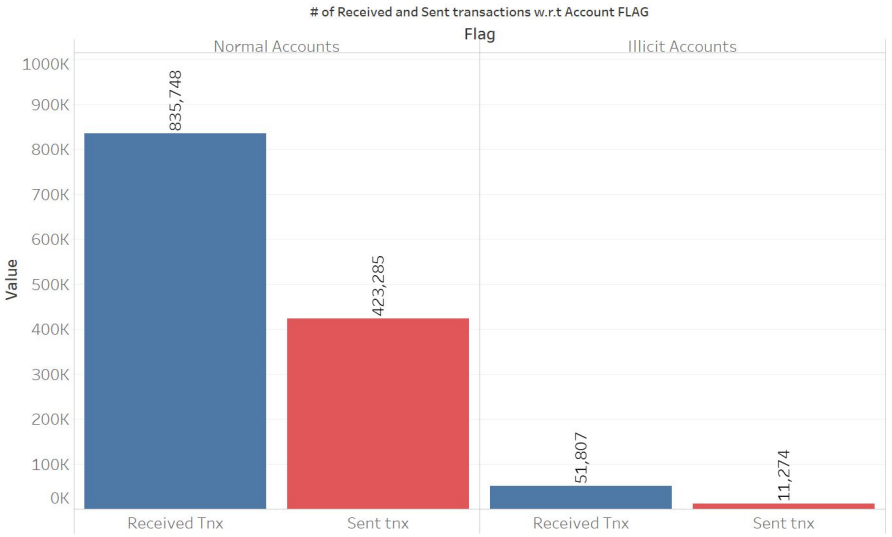

Figure 4.3: Number of sent and received account transactions with respect to the account flag.

Fig. 4.4 highlights the discrepancy in the median total Ether balance held relative to the account class.

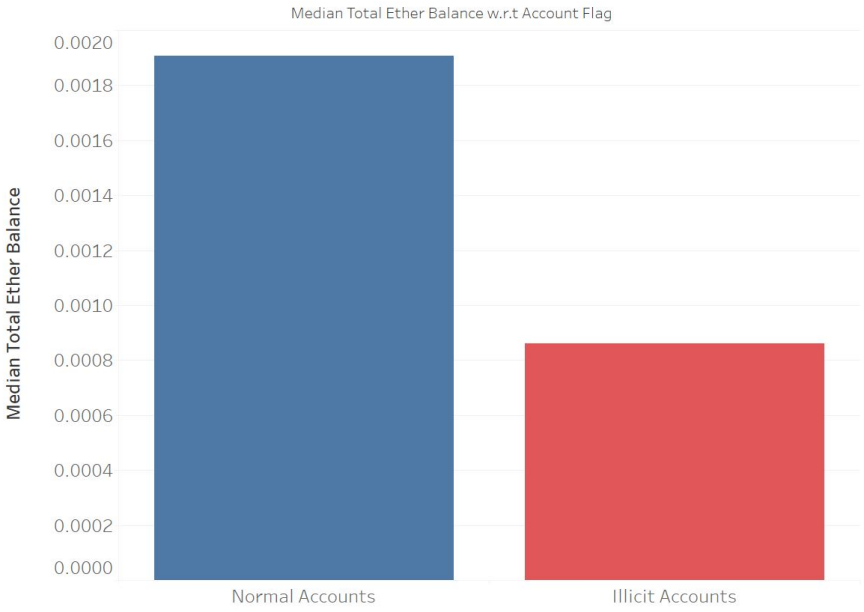

Figure 4.4: Median total Ether balance with respect to the account flag.

4.2 t-distributed stochastic neighbor embedding (t-sne)

To help visualize the available dataset, 2D and 3D t-SNE plots were generated as seen in Fig. 4.5 and Fig. 4.6, respectively.

From the 2D scatter plot in Fig. 4.5, we note that a few distinguishable clusters from both classes, noticeably the three red clusters available near the bottom right region of the plot. Nevertheless, there is still a substantial overlap of different class data points present throughout.

Providing a 30 degree tilt representation in a three-dimensional scatter plot, Fig. 4.6 provides an alternative representation of the 'illicit' account clusters which are evident in Fig. 4.5.

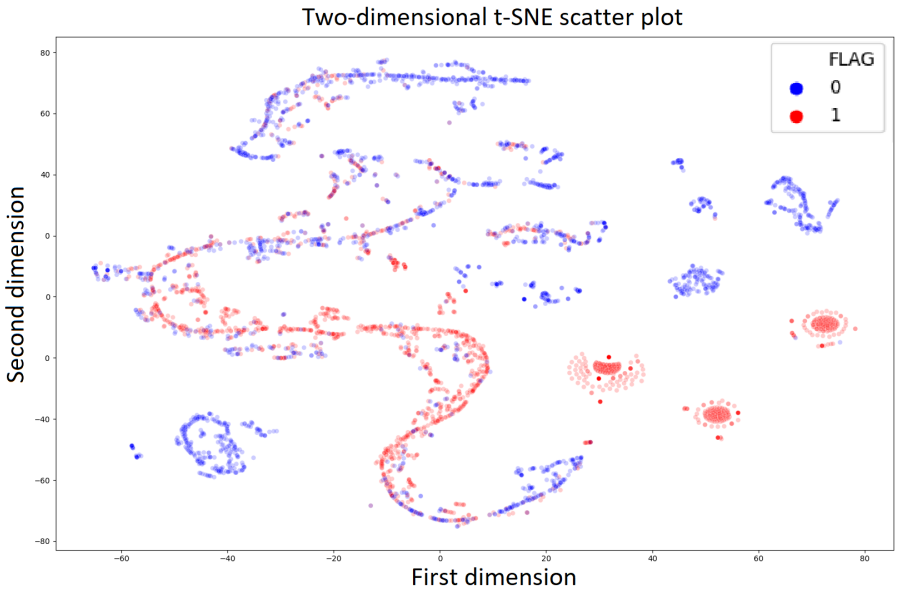

Figure 4.5: Two-dimensional scatter plot following t-SNE transformation with respect to account class. Red and blue data points represent the normal and illicit accounts, respectively.

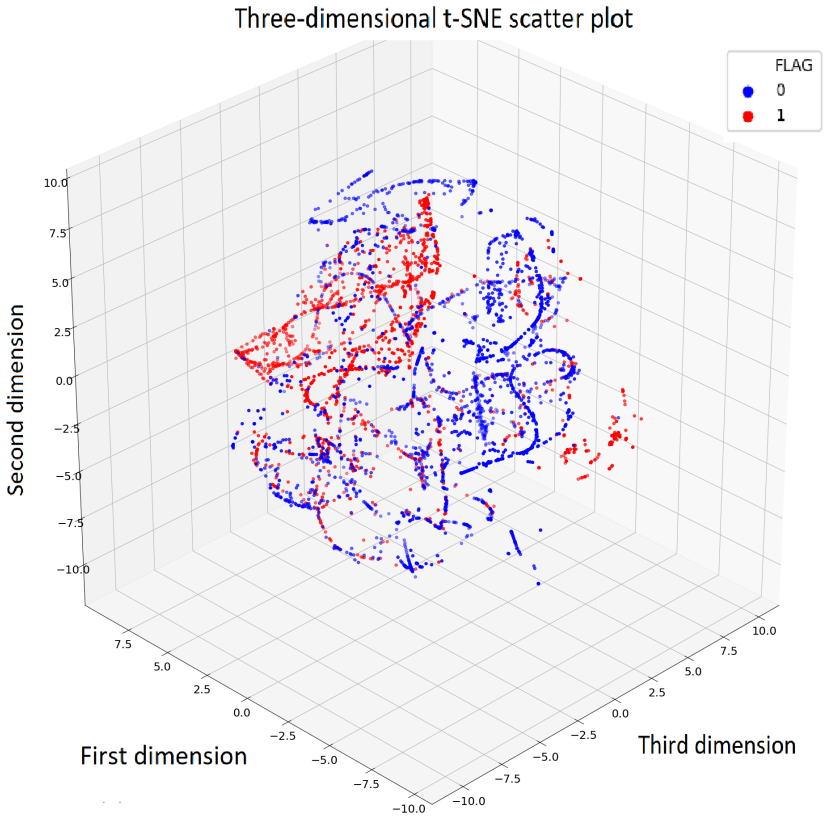

Figure 4.6: Three-dimensional scatter plot following t-SNE transformation with respect to account class. The blue and red data points represent the normal and illicit accounts, respectively.

Although these clusters may be identified through visual analysis, there is still a high level of impurity which indicates that the two classes are not linearly separable. The two figures highlight the necessity of machine learning techniques to help in distinguishing the two binary classes while providing a visual representation of the data at hand.

4.3 XGBOOST

Using grid search cross-validation we identify the optimal values for the three parameters of interest; i) learning rate, ii) number of trees (n_estimators) and iii) max tree depth (max_depth). The AUC measure was used to determine the optimal model performance with respect to the provided parameter values. From Fig. 4.7 we may note that the boosted trees performance is sub-optimal for combinations of low depth and n_estimators when assessed over 10-folds.

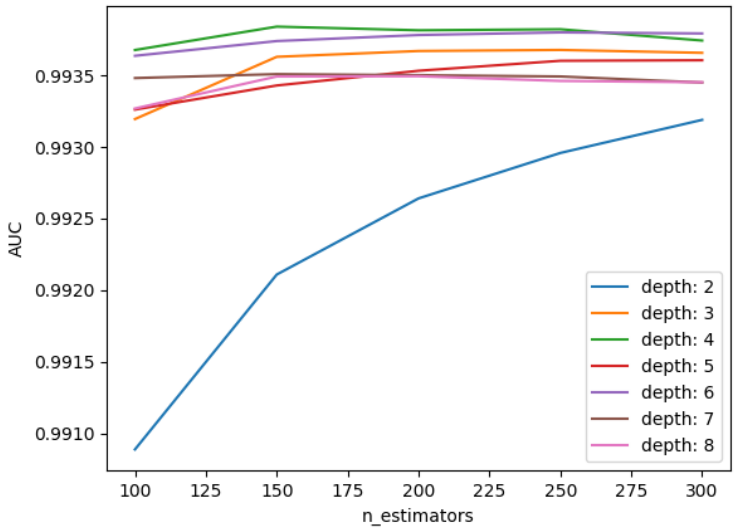

Figure 4.7: Grid-Search 10-fold cross-validation evaluation of n_estimators and max_depth parameters.

The optimal parameters, seen from the green line in the figure, were a max depth of 4 and n_estimators 150 which led to an AUC score of **0.994**. The corresponding accuracy for these optimal parameters was **0.963**.

Table 4.1 presents the cross-validation variation results, with 10-fold cross-validation achieving an optimum AUC of 0.994 (STD of 0.0007) with a tree depth of four, 150 trees and a learning rate of 0.2. The trained XGBoost models perform well in all k-folds, achieving relatively similar performance, with 10-fold cross-validation surpass-

ing the 5-fold results in the other performance values (accuracy and f1-score).

# of Folds	Optimal Depth	Optimal # estimators	Accuracy	F1-score	AUC	Execution time (sec)
3	4	300	0.960 (±0.002)	0.957 (±0.002)	**0.993** (±0.0005)	62.03
4	4	200	0.960 (±0.002)	0.957 (±0.002)	**0.993** (±0.0003)	91.13
5	3	250	0.962 (±0.004)	0.959 (±0.005)	**0.994** (±0.0008)	105.40
10	4	150	0.963 (±0.006)	0.960 (±0.006)	**0.994** (±0.0007)	230.60

Table 4.1: Results for 3-, 4-, 5- and 10-fold cross-validation. Each row reports the best parameters of XGBoost along with the respective average performance measurements. The values in the brackets are the standard deviations of the respective performance measurements across all folds. The last column specifies the total execution time measured on a regular laptop with a core i7-7700HQ 2.8GHz (turbo up to 3.8GHz) and 16GB available RAM using Python v3.6.

Seeking to understand further the models learning capabilities, we also provide the logarithmic loss as well as the classification error with respect to the number of iterations executed by XGBoost in Fig. 4.8 and Fig. 4.9, respectively.

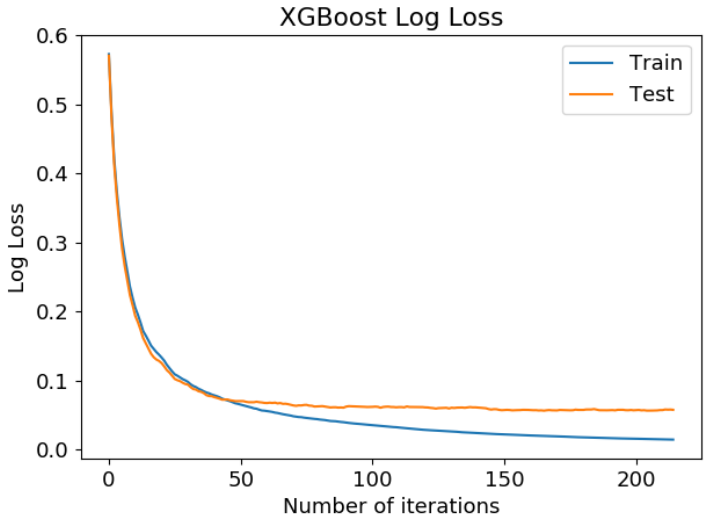

Figure 4.8: Average logarithmic loss over 10-fold cross-validation sets with respect to the number of XGBoost iterations.

The logarithmic loss function indicates the price paid for inaccuracy on the predictions made with the intent of finding a direct mapping between the vector space and the target function. From the two figures we may note that the learning algorithm is converging after approximately 100 iterations.

From the confusion matrix in Fig. 4.10 we may note that the model on average produced 6 False Positive (also Type-2 errors) and 7 False Negatives (also Type-1 errors). These represent the number of accounts

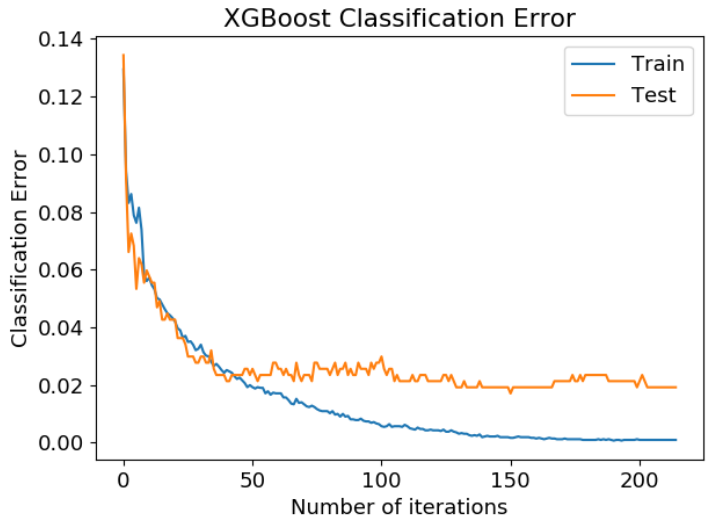

Figure 4.9: Average classification error over 10-fold cross-validation with respect to the number of XGBoost iterations.

the model classified as normal instead of illicit or illicit instead of normal, respectively. In retrospect, we are mostly interested in ensuring that illicit accounts are detected; therefore the 'False Negatives'.

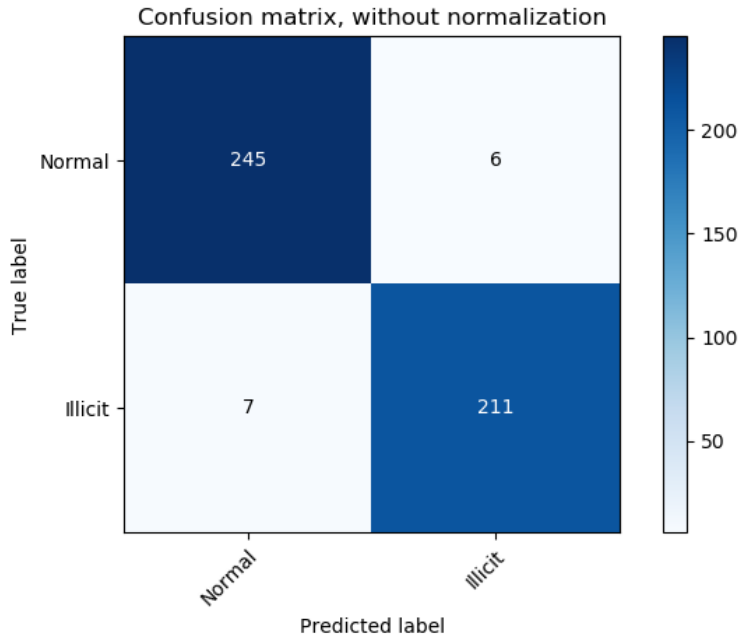

Figure 4.10: Average 10-fold cross-validation confusion matrix.

4.3.1 *Incorrectly classified account information*

We provide additional information pertaining to a few of the incorrectly classified accounts, namely three of each FP and FN category. These incorrectly classified accounts were derived from one of the 10-fold results for which we presented the results in Table 4.1. Using the Etherscan.io search functionality we may present readily available analytics and information on any existing Ethereum 20-byte account address.

False Positives (FP)	False Negatives (FN)
0x4ba0166ed4522e4e8452fad4850e82e6f905bb64	0x4ff4c15af13f6d4950448104d9b97387a03da6f9
0xafc8a3fb05567fcca7b88fcced6886434850e1bf	0x15f4a5d5bbc071fc20bd60b9b7d81aaa8a1142ab
0xa38d8109127028d3e1774413ef28db70c471cb54	0x0d4f74c538613ed6e6c8c1bc8896ecfd45f5ef23
0x5dc9378cd4be85b41699e9cdf25cc669c0696880	0x49311a81f5f5df6c109599ed5d7413d85b8bca18
0xcc6443b0f0a8cf107cb3350ebeae4b771158e652	0xbeebe66b6a511dd2accbff7f258554dbe97f1294
0x1c3f580daeaac2f540c998c8ae3e4b18440f7c45	0x2d791afb30b48a0ac39e6b416d2f46aa095b1704
	0xd4c9f4dbd26ae8c9b87a0d9b3368c0096241a162

Table 4.2: False Positives and False Negatives produced from XGBoost classification on 1 of the 10-folds.

4.3.1.1 *False Negatives - Type 2*

- **Address:** 4c15af13F6D4950448104D9B97387A03dA6f9
  Account was active for a period of 3 hrs in total, carrying out 5 transactions; 3 incoming and 2 outgoing on the 25th of September 2018. All transactions were sent to or received from unique addresses. Account shows signs of an externally owned account (EOA).

- **Address:** 0x15f4a5d5bbc071fc20bd60b9b7d81aaa8a1142ab
  Account was said to be a phishing scam for a startup called OneLedger. The account was active for 53 days between February 8^{th}, 2018 and April 3^{rd} starting off with an Ether balance of 0.8. 7 Transactions were executed in total; with 2 incoming and 5 outgoing transfers. All transactions were carried out with unique account addresses - therefore interacting with new accounts.

- **Address:** 0x0d4f74c538613ed6e6c8c1bc8896ecfd45f5ef23
  Account was said to have partaken in a Phishing scam concerning the Origin Protocol (allowing for a decentralized marketplace for tokens [59]). A total of 209 transactions were carried out by this illicitly available smart contract, out of which only 6 incoming transaction addresses were unique. Account activity was for less than 12 hrs on the 14^{th} April, 2018.

4.3.1.2 *False Positives - Type 1*

- **Address:** 0x4BA0166eD4522E4E8452Fad4850e82e6f905bb64
  The account executed a total of 54 transactions, with ERC-20 Token transfer events account for 14 of them. Having had the highest ETH balance of 2.0253 which translated to $589.02 on the respective day, the accounts activity spans approximately two years - initiated in June 2017. The majority of the transactions were to unique addresses (from/to which no prior transactions were carried out). As of 31/05/2019, the account has a balance of 0.2158 Ether.

- **Address:** 0xAFc8a3Fb05567fCca7B88FCCEd6886434850E1Bf
  In all a total of 12 transactions were executed; with 1 transaction incoming and 11 outgoing. All transactions were made to 1-time addresses up until all the initial balance of 3.9964476 Ether purchased on June 1^{st}, 2017 was distributed among the 11 unique accounts.

- **Address:** 0xa38d8109127028d3E1774413eF28Db70C471CB54
  A contract account which participated in 144 transactions from the 8^{th} June 2017 till the 27^{th} of June 2017. Titled as PaymentForwarder, the contract has seen payments above 1000 Ether (transaction hash: '0x3888e8a643a0d0b5e3240872d87aabacf67c8edb6e14142e50e04360d0141893') in a single transaction. The majority of all executed transactions were from/to new accounts.

4.4 XGBOOST FEATURE RANKING

The XGBoost package also offers 3 feature importance measures for which we provide the resultant top 10 features. As explained in section 3.5.3, three measures are available for which we provide their respective results in Fig. 4.11, Fig. 4.12 and Fig. 4.13.

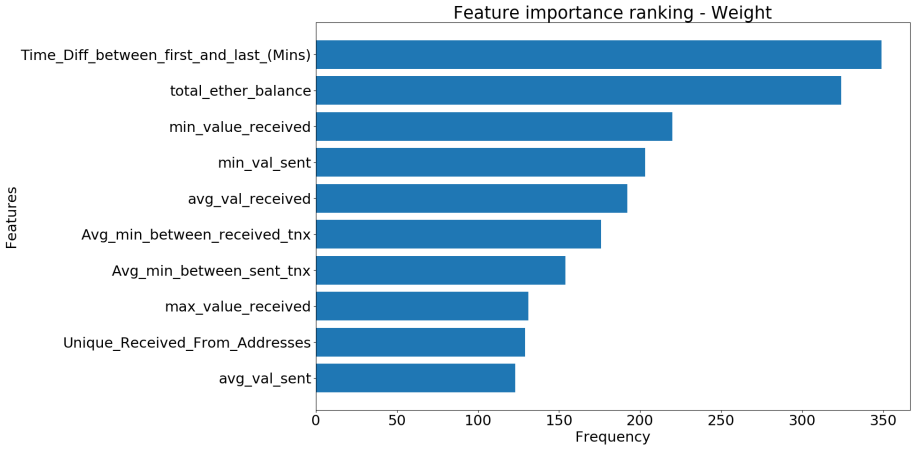

Figure 4.11: The average ranking of the top 10 features determined by XGBoost with respect to 'Weight'. The frequency defines to number of times the variable was split on.

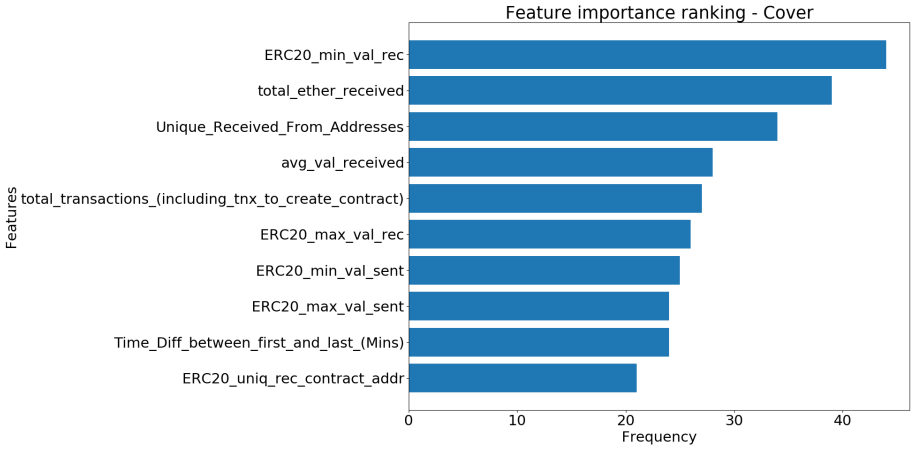

Figure 4.12: The average ranking of the top 10 features determined by XGBoost with respect to 'Cover'. The frequency defines to number of times the variable was split on.

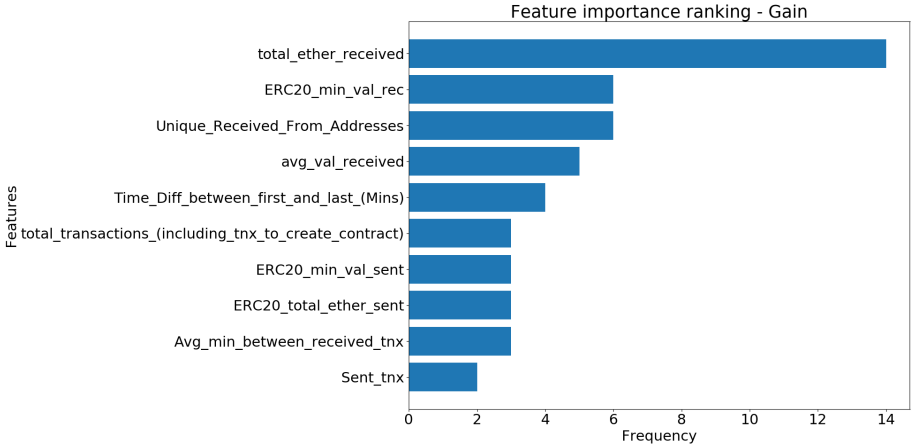

Figure 4.13: The average ranking of the top 10 features determined by XGBoost with respect to 'Gain'. The frequency defines to number of times the variable was split on.

A number of train-test split configurations were also implemented in order to ascertain the rigidity and stability of the features with respect to the models being built. From these numerous tests, we conclude that the 'Weight' parameter delivers similar feature importance order, especially for the top 10 features. The features retained their respective order, moving 1 or 2 places up or down respectively but remaining in the top 10 features. Whenever features were not displayed in the top 10, due to them being 9^{th} or 10^{th} usually, they were present in the top 15 features.

The complete XGBoost feature ranking for all the 42 features has been made available in the Appendix, Table. A.1.

4.5 RANDOM FOREST IMPLEMENTATION

We assess the results obtained by a basic 'Random Forest' model. Made available in Table 4.3, the discrepancy in performance attained by the 'Random Forest' model with respect to that generated by XGBoost is evident. An average accuracy and AUC of **93.902** and **0.9836** respectively over 10-fold cross-validation with stratification on the account class.

	Precision	**Recall**	**f1-score**	**support**
Normal	0.936	0.954	0.938	251
Illicit	0.946	0.924	0.932	218
Micro Average	0.938	0.938	0.938	469
Macro Average	0.94	0.938	0.938	469
Weight Average	0.94	0.938	0.938	469

Table 4.3: The results obtained from 10-fold cross-validation using Random Forest.

Due to the evident variation when compared to Table 4.1, further parameter optimization was not carried out. We therefore place our focus on the XGBoost model implemented.

CHAPTER FIVE

5 DISCUSSION

5.1 DISCUSSION

Similar research in the domain address the problem from alternative perspectives; with the majority of works looking to detect Ponzi schemes masked as smart contracts, anomaly transactions associated with illegal activity as well as money laundering schemes. While these approaches positively contribute towards mitigating illicit activity over blockchain networks, they do not provide an approach capable of detecting such behavior at an 'account level' with a high level of accuracy.

Amalgamating various ideas and approaches used in similar work, mainly feature extraction and feature importance, we present a novel approach capable of detecting illicit activity on the Ethereum network at an account level using the state of the art XGBoost classification model. Using a balanced dataset, with 'normal' and 'illicit' accounts the model proved to be highly effective, with an average AUC of 0.994 and standard deviation of 0.0007 over 10-fold cross-validation. We also note that the model manifests generalization capabilities while also being resilient towards overfitting; determined by i) the low standard deviations in AUC and accuracy, as well as ii) the results obtained from smaller number of k-folds for cross-validation. This was all achieved with a total execution time of less than 4 minutes (core i7-7700HQ 2.8GHz with turbo boost up to 3.8GHz and 16GB available RAM). Although efficiency is not a primary concern for such applications, the resources required in order to generate the aforementioned results are indicative of a highly scalable system.

Assessing the rank of the feature importance, we note that the time between the first and last transaction (representing the total duration of account usage) has the largest impact on the account class. On further inspection we realise that the average time for normal and illicit accounts is 136.9 and 38.4 days, respectively. This might suggest that illicit accounts are established for the sole purpose of engaging in their respective illegal activities. The average minimum Ether sent was also 18% higher for normal accounts. A large disparity was also present in the average available total balance, with illicit accounts retaining a median of 0.000861 Ether compared to 0.001908 Ether held by normal accounts. Additionally, we also observe that the average value of Ether sent to a contract as well as the number of unique address to which tokens were sent via ERC20 standard contracts were the least important features.

5.2 LIMITATIONS

The EtherscanAPI provides a maximum of 10,000 transactions per account. While this was sufficient enough to train the model and produce the level of performance attained, it would be interesting to see if acquiring all the transactions per account would have any effect on the results achieved. While this is desirable, this will come at a cost of time (more transactions would increase time to retrieve and compute the features generated).

The number of available documented scams is rather limited in size; only increasing intermittently when the Ethereum community identifies or encounters a potential scam. A larger dataset may increase the variety of scams acquired; possibly helping in the identification of inaccurately classified account instances.

5.3 FUTURE WORK

There are various possibilities for future work. One direction may be to investigate the adaptability of our proposed approach for the detection of illicit activity on other blockchain networks. Another approach would be to explore whether the described methodology may be altered to detect illicit activity at a 'transactional level'.

Additionally, one may also look into other potential features which may further contribute towards improved classification results. For instance, internal transactions which do not appear on the Ethereum blockchain but are brought about through the execution of a contract (known as internal message calls) may also be analysed and included as additional features for the XGBoost classification model. These additional message calls may provide more discriminative features which would further aid the classification model in identifying the target class (normal or illicit account).

Another possibility would be to further enhance the features extracted through the implementation of graph analysis.

Part I

APPENDIX

APPENDIX

A

Table A.1: Complete list of the 42 extracted features.

	Complete set of Features extracted			
	Extracted Feature	**Rank**	**Description**	**Data Type**
1	**Avg_min_between_sent_tnx**	7	Average time between sent transactions for account in minutes	Integer
2	**Avg_min_between_received_tnx**	6	Average time between received transactions for account in minutes	Integer
3	**Time_Diff_between_first_and_last(Mins)**	1	Time difference between the first and last transaction	Integer
4	**Sent_tnx**	17	Total number of sent normal transactions	Integer
5	**Received_tnx**	18	Total number of received normal transactions	Integer
6	**Number_of_Created_Contracts**	28	Total Number of created contract transactions	Integer
7	**Unique_Received_From_Addresses**	8	Total Unique addresses from which account received transactions	Integer
8	**Unique_Sent_To_Addresses**	20	Total Unique addresses from which account sent transactions	Integer
9	**Min_Value_Received**	3	Minimum value in Ether ever received	Double
10	**Max_Value_Received**	9	Maximum value in Ether ever received	Double
11	**Avg_Value_Received**	5	Average value in Ether ever received	Double
12	**Min_Val_Sent**	4	Minimum value of Ether ever sent	Double
13	**Max_Val_Sent**	13	Maximum value of Ether ever sent	Double
14	**Avg_Val_Sent**	10	Average value of Ether ever sent	Double
15	**Min_Value_Sent_To_Contract**	39	Minimum value of Ether sent to a contract	Double
16	**Max_Value_Sent_To_Contract**	40	Maximum value of Ether sent to a contract	Double
17	**Avg_Value_Sent_To_Contract**	41	Average value of Ether sent to contracts	Double
18	**Total_Transactions(Including_Tnx_to_Create_Contract)**	12	Total number of transactions	Integer
19	**Total_Ether_Sent**	14	Total Ether sent for account address	Double
20	**Total_Ether_Received**	11	Total Ether received for account address	Double
21	**Total_Ether_Sent_Contracts**	38	Total Ether sent to Contract addresses	Double
22	**Total_Ether_Balance**	2	Total Ether Balance following enacted transactions	Double
23	**Total_ERC20_Tnxs**	16	Total number of ERC20 token transfer transactions	Integer
24	**ERC20_Total_Ether_Received**	19	Total ERC20 token received transactions in Ether	Double
25	**ERC20_Total_Ether_Sent**	24	Total ERC20 token sent transactions in Ether	Double
26	**ERC20_Total_Ether_Sent_Contract**	37	Total ERC20 token transfer to other contracts in Ether	Double
27	**ERC20_Uniq_Sent_Addr**	26	Number of ERC20 token transactions sent to Unique account addresses	Integer
28	**ERC20_Uniq_Rec_Addr**	25	Number of ERC20 token transactions received from Unique addresses	Interger
29	**ERC20_Uniq_Rec_Contract_Addr**	23	Number of ERC20 token transactions received from Unique contract addresses	Integer
30	**ERC20_Avg_Time_Between_Sent_Tnx**	31	Average time between ERC20 token sent transactions in minutes	Integer
31	**ERC20_Avg_Time_Between_Rec_Tnx**	33	Average time between ERC20 token received transactions in minutes	Integer
32	**ERC20_Avg_Time_Between_Contract_Tnx**	32	Average time ERC20 token between sent token transactions	Integer
33	**ERC20_Min_Val_Rec**	15	Minimum value in Ether received from ERC20 token transactions for account	Double
34	**ERC20_Max_Val_Rec**	21	Maximum value in Ether received from ERC20 token transactions for account	Double
35	**ERC20_Avg_Val_Rec**	22	Average value in Ether received from ERC20 token transactions for account	Double
36	**ERC20_Min_Val_Sent**	27	Minimum value in Ether sent from ERC20 token transactions for account	Double
37	**ERC20_Max_Val_Sent**	29	Maximum value in Ether sent from ERC20 token transactions for account	Double
38	**ERC20_Avg_Val_Sent**	30	Average value in Ether sent from ERC20 token transactions for account	Double
39	**ERC20_Uniq_Sent_Token_Name**	34	Number of Unique ERC20 tokens transferred	Integer
40	**ERC20_Uniq_Rec_Token_Name**	36	Number of Unique ERC20 tokens received	Integer
41	**ERC20_Most_Sent_Token_Type**	42	Most sent token for account via ERC20 transaction	String
42	**ERC20_Most_Rec_Token_Type**	35	Most received token for account via ERC20 transaction	String

BIBLIOGRAPHY

[1] URL: https://etherscamdb.info/api/scams/.

[2] In: *The Nilson Report* 1096 (2016). URL: https://nilsonreport.com/upload/content_promo/The_Nilson_Report_10-17-2016.pdf.

[3] Vishal Morde 10155881593927876. *XGBoost Algorithm: Long May She Reign!* 2019. URL: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d.

[4] Douglas G Altman and J Martin Bland. "Diagnostic tests. 1: Sensitivity and specificity." In: *BMJ: British Medical Journal* 308.6943 (1994), p. 1552.

[5] Saifedean Ammous. "Blockchain Technology: What is it good for?" In: *Available at SSRN 2832751* (2016).

[6] Elli Androulaki, Ghassan O Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. "Evaluating user privacy in bitcoin." In: *International Conference on Financial Cryptography and Data Security*. Springer. 2013, pp. 34–51.

[7] Walter Edwin Arnoldi. "The principle of minimized iterations in the solution of the matrix eigenvalue problem." In: *Quarterly of applied mathematics* 9.1 (1951), pp. 17–29.

[8] Tomaso Aste, Paolo Tasca, and Tiziana Di Matteo. "Blockchain technologies: The foreseeable impact on society and industry." In: *Computer* 50.9 (2017), pp. 18–28.

[9] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. "A survey of attacks on Ethereum smart contracts." In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 1007.

[10] Osato Avan-Nomayo. *Malta Gov't Committed to 'Blockchain Island' Vision Despite Criticism*. 2019. URL: https://bitcoinist.com/malta-govt-committed-to-blockchain-island-vision-despite-criticism/.

[11] Tanya Bahrynovska. *History of Ethereum Security Vulnerabilities, Hacks and Their Fixes*. 2019. URL: https://applicature.com/blog/blockchain-technology/history-of-ethereum-security-vulnerabilities-hacks-and-their-fixes.

[12] Massimo Bartoletti, Barbara Pes, and Sergio Serusi. "Data mining for detecting Bitcoin Ponzi schemes." In: *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE. 2018, pp. 75–84.

[13] Massimo Bartoletti, Salvatore Carta, Tiziana Cimoli, and Roberto Saia. "Dissecting Ponzi schemes on Ethereum: identification, analysis, and impact." In: *arXiv preprint arXiv:1703.03779* (2017).

[14] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[15] *Bitcoin blockchain size 2010-2019 | Statistic*. URL: https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/.

[16] Andrew P Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms." In: *Pattern recognition* 30.7 (1997), pp. 1145–1159.

[17] Vitalik Buterin. "Ethereum: Platform Review." In: *Opportunities and Challenges for Private and Consortium Blockchains* (2016).

[18] Vitalik Buterin et al. "A next-generation smart contract and decentralized application platform." In: ().

[19] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system." In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM. 2016, pp. 785–794.

[20] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, and Yuan Tang. "Xgboost: extreme gradient boosting." In: *R package version 0.4-2* (2015), pp. 1–4.

[21] Ting Chen, Yuxiao Zhu, Zihao Li, Jiachi Chen, Xiaoqi Li, Xiapu Luo, Xiaodong Lin, and Xiaosong Zhange. "Understanding ethereum via graph analysis." In: *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE. 2018, pp. 1484–1492.

[22] Weili Chen, Zibin Zheng, Jiahui Cui, Edith Ngai, Peilin Zheng, and Yuren Zhou. "Detecting Ponzi schemes on Ethereum: Towards healthier blockchain technology." In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2018, pp. 1409–1418.

[23] Weili Chen, Zibin Zheng, Edith C-H Ngai, Peilin Zheng, and Yuren Zhou. "Exploiting Blockchain Data to Detect Smart Ponzi Schemes on Ethereum." In: *IEEE Access* 7 (2019), pp. 37575–37586.

[24] Hyun-Soo Choi, Siwon Kim, Jung Eun Oh, Jee Eun Yoon, Jung Ah Park, Chang-Ho Yun, and Sungroh Yoon. "XGBoost-Based Instantaneous Drowsiness Detection Framework Using Multitaper Spectral Information of Electroencephalography." In: *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM. 2018, pp. 111–121.

[25] Christopher D Clack, Vikram A Bakshi, and Lee Braine. "Smart contract templates: foundations, design landscape and research directions." In: *arXiv preprint arXiv:1608.00771* (2016).

[26] Michael Crosby, Pradan Pattanayak, Sanjeev Verma, Vignesh Kalyanaraman, et al. "Blockchain technology: Beyond bitcoin." In: *Applied Innovation* 2.6-10 (2016), p. 71.

[27] Brady Dale, Brady Dale, and David Floyd. *Ponzi Games Are Breaking Out on the Ethereum Blockchain*. 2018. URL: https://www.coindesk.com/scam-or-be-scammed-ponzi-games-are-breaking-out-on-ethereum.

[28] Chris Dannen. *Introducing Ethereum and Solidity*. Springer, 2017.

[29] Ali Dorri, Marco Steger, Salil S Kanhere, and Raja Jurdak. "Blockchain: A distributed solution to automotive security and privacy." In: *IEEE Communications Magazine* 55.12 (2017), pp. 119–125.

[30] Etherscan.io. *Ethereum Blockchain Explorer*. URL: https://etherscan.io/.

[31] Tom Fawcett and Foster J Provost. "Combining Data Mining and Machine Learning for Effective User Profiling." In: *KDD*. 1996, pp. 8–13.

[32] Gianni Fenu, Lodovica Marchesi, Michele Marchesi, and Roberto Tonelli. "The ICO phenomenon and its relationships with ethereum smart contract environment." In: *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE. 2018, pp. 26–32.

[33] Antonio Fernández Anta, Paul Rimba, Andrés Abeliuk, Manuel Cebrian, Ioannis Stavrakakis, An Binh Tran, and Oluwasegun Ojo. "Miner Dynamics on the Ethereum Blockchain." In: (2019).

[34] Priyeshu Garg. *Mistake or Money Laundering Scheme? Someone Pays $450,000 in Fees to Send 0.1 ETH*. 2019. URL: https://cryptoslate.com/mistake-or-money-laundering-scheme-someone-pays-450000-in-fees-to-send-0-1-eth/.

[35] Yashu Gola. *Ethereum Most Popular Crypto Among Phishers who made $2.3 Million in Q2 2018: Kaspersky*. 2018. URL: https://www.ccn.com/ethereum-most-popular-crypto-among-phishers-who-made-2-3-million-in-q2-2018-kaspersky.

[36] *Gradient Boosting Machines*. URL: http://uc-r.github.io/gbm_regression.

[37] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh. *Feature extraction: foundations and applications*. Vol. 207. Springer, 2008.

[38] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. "Rolx: structural role extraction & mining in large graphs." In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2012, pp. 1231–1239.

[39] Jason Hirshman, Yifei Huang, and Stephen Macke. "Unsupervised approaches to detecting anomalous behavior in the bitcoin transaction network." In: *3rd ed. Technical report, Stanford University* (2013).

[40] Richard HorrocksRichard Horrocks. *What is Geth's "light" sync, and why is it so fast?* URL: https://ethereum.stackexchange.com/questions/11297/what-is-geths-light-sync-and-why-is-it-so-fast.

[41] Seyoung Huh, Sangrae Cho, and Soohyung Kim. "Managing IoT devices using blockchain platform." In: *2017 19th international conference on advanced communication technology (ICACT)*. IEEE. 2017, pp. 464–467.

[42] Frank Jobse. "Detecting suspicious behavior in the Bitcoin network." PhD thesis. Tilburg University, 2017.

[43] Ron Kohavi and Clayton Kunz. "Option decision trees with majority votes." In: *ICML*. Vol. 97. 1997, pp. 161–169.

[44] Doug Laney. "3D data management: Controlling data volume, velocity and variety." In: *META group research note* 6.70 (2001), p. 1.

[45] In Lee. "Big data: Dimensions, evolution, impacts, and challenges." In: *Business Horizons* 60.3 (2017), pp. 293–303.

[46] Andy Liaw, Matthew Wiener, et al. "Classification and regression by randomForest." In: *R news* 2.3 (2002), pp. 18–22.

[47] Loi Luu, Yaron Velner, Jason Teutsch, and Prateek Saxena. "Smartpool: Practical decentralized pooled mining." In: *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 2017, pp. 1409–1426.

[48] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[49] David M Magerman. "Statistical decision-tree models for parsing." In: *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 1995, pp. 276–283.

[50] Shamil Magomedov, Sergei Pavelyev, Irina Ivanova, Alexey Dobrotvorsky, Marina Khrestina, and Timur Yusubaliev. "Anomaly Detection with Machine Learning and Graph Databases in Fraud Management." In: *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS* 9.11 (2018), pp. 33–38.

[51] 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*. ACM. 2013, pp. 127–140.

[52] Weizhi Meng, Elmar Wolfgang Tischhauser, Qingju Wang, Yu Wang, and Jinguang Han. "When intrusion detection meets blockchain technology: a review." In: *Ieee Access* 6 (2018), pp. 10179–10188.

[53] Tom Michael Mitchell. *The discipline of machine learning*. Vol. 9. Carnegie Mellon University, School of Computer Science, Machine Learning . . ., 2006.

[54] *More banks are caught up in money-laundering scandals*. 2019. URL: https://www.economist.com/finance-and-economics/2019/03/09/more-banks-are-caught-up-in-money-laundering-scandals.

[55] MyEtherWallet. *MyEtherWallet/ethereum-lists*. URL: https://github.com/MyEtherWallet/ethereum-lists/blob/master/src/addresses/addresses-darklist.json.

[56] Sarang Narkhede. *Understanding AUC - ROC Curve*. 2018. URL: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

[57] Didrik Nielsen. "Tree Boosting With XGBoost-Why Does XGBoost Win" Every" Machine Learning Competition?" MA thesis. NTNU, 2016.

[58] *Online Fraud Benchmark Report: Persistence is critical*. 2017. URL: https://www.cybersource.com/content/dam/cybersource/2017_Fraud_Benchmark_Report.pdf.

[59] *Origin Protocol*. URL: https://www.originprotocol.com/en.

[60] Eibhlín O'Kane. "Detecting Patterns in the Ethereum Transactional Data using Unsupervised Learning." In: (2018).

[61] Suvasini Panigrahi, Amlan Kundu, Shamik Sural, and Arun K Majumdar. "Credit card fraud detection: A fusion approach using Dempster–Shafer theory and Bayesian learning." In: *Information Fusion* 10.4 (2009), pp. 354–363.

[62] Al Pascual, Kyle Marchini, and Sarah Miller. *2016 Identity Fraud: Fraud Hits an Inflection Point*. 2016. URL: https://www.javelinstrategy.com/coverage-area/2016-identity-fraud-fraud-hits-inflection-point.

[63] Andy Peterson. *Self-Proclaimed 'Exit Scam' and Gambling DApps Rake in $43M in Ether*. 2018. URL: https://www.ccn.com/self-proclaimed-exit-scam-and-gambling-dapps-rake-in-43m-in-ether.

[64] *Post-Mortem Investigation (Feb 2016)*. 2016. URL: https://www.kingoftheether.com/postmortem.html.

[65] *Pulling fraud out of the shadows*. 2018. URL: https://www.pwc.com/gx/en/forensics/global-economic-crime-and-fraud-survey-2018.pdf.

[66] J Ross Quinlan et al. "Bagging, boosting, and C4. 5." In:

[67] Kefa Rabah. "Convergence of AI, IoT, big data and blockchain: a review." In: *The Lake Institute Journal* 1.1 (2018), pp. 1–18.

[68] Fergal Reid and Martin Harrigan. "An analysis of anonymity in the bitcoin system." In: *Security and privacy in social networks*. Springer, 2013, pp. 197–223.

[69] Marit Rudlang. "Comparative Analysis of Bitcoin and Ethereum." MA thesis. NTNU, 2017.

[70] Sara Saberi, Mahtab Kouhizadeh, Joseph Sarkis, and Lejia Shen. "Blockchain technology and its relationships to sustainable supply chain management." In: *International Journal of Production Research* 57.7 (2019), pp. 2117–2135.

[71] Maria Shen. *Dev Report*. 2019. URL: https://medium.com/@ElectricCapital/dev-report-476df4ff1fd2.

[72] Bruno Skvorc. *Ethereum: Internal Transactions & Token Transfers Explained*. 2018. URL: https://www.sitepoint.com/ethereum-internal-transactions-token-transfers/.

[73] Aleksandra Sokolowska. *How to interact with the Ethereum blockchain and create a database with Python and SQL*. 2018. URL: https://medium.com/validitylabs/how-to-interact-with-the-ethereum-blockchain-and-create-a-database-with-python-and-sql-3dcbd579b3c0.

[74] Nick Szabo. "Formalizing and securing relationships on public networks." In: *First Monday* 2.9 (1997).

[75] L Torlay, Marcela Perrone-Bertolotti, E Thomas, and Monica Baciu. "Machine learning–XGBoost analysis of language networks to classify patients with epilepsy." In: *Brain informatics* 4.3 (2017), p. 159.

[76] Marie Vasek and Tyler Moore. "There's no free lunch, even using Bitcoin: Tracking the popularity and profits of virtual currency scams." In: *International conference on financial cryptography and data security*. Springer. 2015, pp. 44–61.

[77] Maria Vergelis, Nadezhda Demidova, and Tatyana Shcherbakova. *Spam and phishing in Q2 2018*. 2018. URL: https://securelist.com/spam-and-phishing-in-q2-2018/87368/.

[78] Jan Vermulen. *Bitcoin and Ethereum vs Visa and PayPal – Transactions per second*. 2017. URL: https://mybroadband.co.za/news/banking/206742-bitcoin-and-ethereum-vs-visa-and-paypal-transactions-per-second.html.

[79] Dejan Vujičić, Dijana Jagodić, and Siniša Ranđić. "Blockchain technology, bitcoin, and Ethereum: A brief overview." In: *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*. IEEE. 2018, pp. 1–6.

[80] XuGang Wang, Zheng Tang, Hiroki Tamura, Masahiro Ishii, and WD Sun. "An improved backpropagation algorithm to avoid the local minima problem." In: *Neurocomputing* 56 (2004), pp. 455–460.

[81] Martin Young. *Short The Bankers: Another Major Bank Ordered Closed for Money Laundering*. 2019. URL: https://www.newsbtc.com/2019/02/20/short-the-bankers-another-major-bank-ordered-closed-for-money-laundering/.

[82] Dahai Zhang, Liyang Qian, Baijin Mao, Can Huang, Bin Huang, and Yulin Si. "A data-driven design for fault detection of wind turbines using random forests and xgboost." In: *IEEE Access* 6 (2018), pp. 21020–21031.

DECLARATION

Put your declaration here.

Groningen, Netherlands, August 15th 2019

Steven Farrugia