Identifying likely reputable Ethereum Blockchain based Projects

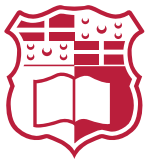
Cyrus Malik

Supervisor: Prof Joshua Ellul

Co-Supervisor: Dr Josef Bajada

May 2023

Submitted in partial fulfilment of the requirements for the degree of Computing Science.



L-Università ta' Malta
Faculty of Information & Communication Technology

Abstract

The identification of likely-reputable projects based on the Ethereum Blockchain is a crucial task in the ever-evolving landscape of blockchain technology. As the number of Ethereum-based projects increases, differentiating between trustworthy ventures and potentially fraudulent ones becomes increasingly challenging. While previous research has focused on detecting illicit behavior, it has not addressed the specific aspect of investigating whether a project's potential to be a reputable one could be determined in an automated fashion. Consequently, this research aims to investigate whether machine learning algorithms and data analysis techniques can be used to assist investors, developers, regulators, and the Ethereum community in evaluating project reputations.

The research endeavors to achieve classification of a dataset consisting of 2,179 accounts identified by the Ethereum community for their association with illicit activities, alongside 3,977 accounts that are considered to have a high likelihood of being reputable, obtained from CoinGecko. Through evaluation process utilizing 10-fold cross-validation, the LightGBM algorithm achieves an impressive average accuracy of 0.9984 ($\pm$0.0003) and an average AUC of 0.9991 ($\pm$0.0003). Notably, the most influential features in the final model are "Time difference between received transactions (minutes)," and "Average time difference between ERC20 received transactions." Based on the obtained results, it was shown that the proposed approach effectively identifies likely reputable projects on the Ethereum Blockchain.

The outcomes of this research will contribute to establishing a more informed and secure investment environment within the Ethereum ecosystem. Stakeholders will benefit from making well-informed decisions, mitigating risks associated with unreliable projects, and fostering the growth of reputable ventures. Additionally, this research sets the groundwork for future advancements in project evaluation methodologies, further enhancing the development and maturity of the Ethereum Blockchain ecosystem.

Acknowledgements

With great pleasure, I would like to express my deepest appreciation to the following people for their unwavering support during the work of my final year project. First and foremost, my supervisors Prof. Joshua Ellul and Dr. Josef Bajada for their constant, helpful and instant feedback given. Finally, I am deeply indebted to my family, and girlfriend Ms Jessica Ghori, for their constant support and patience while listening to me talking non-stop about this project.

Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence.

AUC Area Under ROC Curve.

CA Contract accounts.

EFB Exclusive Feature Bundling.

EOA Externally Owned Accounts.

EVM Ethereum Virtual Machine.

FN False Negatives.

FP False Positives.

GOSS Gradient-based One-Side Sampling.

ICO Initial Coin Offerings.

IoT Internet of Things.

ROC Receiver Operating Characteristic.

TN True Negatives.

TP True Positives.

TPR True Positive Rate.

1 Introduction

The Ethereum blockchain network, introduced in 2015, aimed to enhance Bitcoin's capabilities by introducing Smart contracts [1]. It achieved decentralization of computational logic while maintaining security and endorsing decentralized computational processes. By 2019, Ethereum became the second-largest cryptocurrency with a market share of 14.5 billion USD [1]. Ethereum facilitates the transfer of Ether (Ethereum's currency), creation, and execution of smart contracts

Similar to other platforms, Ethereum has encountered instances where certain entities or users have engaged in illicit activities on the network. These individuals take advantage of the pseudo-anonymity provided by the network to carry out such activities. This raises the question whether it is possible to identify a given project as being likely-reputable in one way or another [2].

1.1 Problem Definition

While most research focus and look at detection of illicit activity [3], ponzi-scheme detection [4] or anomaly detection [5]; these methodologies do not determine whether a given project is reputable and only identify illicit behaviour. Through this research, we aim to devise a systematic approach that leverages transaction analysis to evaluate the credibility, transparency, and overall trustworthiness of Ethereum projects. By examining the transaction history of projects on the Ethereum blockchain, we seek to differentiate between projects with strong reputations and those that may be associated with illicit activities.

1.2 Can we identify likely-reputable projects?

The publicly accessible distributed ledger provided by the Ethereum blockchain network would undeniably fall under the category of big data - encompassing over 1.923 billion[2] transactions and 17.29 million blocks as of May 2023. The manual scrutiny of these transactions in an attempt to detect any abnormal patterns would be an impractical and perpetual task. In light of the ongoing generation of blocks, which encompass transactions and smart contracts, the adoption of machine learning algorithms becomes imperative to cope with the volume.

Machine learning algorithms would not only facilitate the detection of past transactions associated with reputable projects but also categorize newly generated

[1] https://coinmarketcap.com/currencies/ethereum/

[2] https://blockchair.com/ethereum/charts/total-transaction-count

transactions exhibiting comparable characteristics. These identifying features are derived from both the existing transaction data and any extracted features. Conceptually, the implementation of machine learning algorithms would assist in differentiating transactions that exhibit ”likely-reputable” behavior from those displaying unusual patterns within user accounts. This differentiation is achieved by the algorithms learning and recognizing specific features associated with either anomalous or typical likely-reputable behavior.

We can further divide the aforementioned main question into a series of research questions based on the sequence in which they will be executed.

(RQ1) What input variables should be selected from the transaction dataset for the algorithms employed?

(RQ2) Is it possible to ascertain the effectiveness of algorithm parameters on the model's results and determine the optimal parameters? Can these results be interpreted visually?

(RQ3) Can we reliably identify and provide justifications for any accounts that display anomalous behavior?

(RQ4) Which supplementary features can be extracted to enrich the existing data?

(RQ5) Which machine learning algorithm can effectively produce the most favorable outcomes based on the given criteria?

1.3 Motivation

1.3.1 Why analyse Ethereum?

According to Vitalik Buterin, the Ethereum platform is a continuously evolving technology that enables developers to easily create applications while maintaining the security and decentralized nature of the blockchain [6]. It has become the predominant platform for smart contracts, drawing the highest number of engaged developers [3] and enjoying widespread adoption in the industry. With a robust foundation and architecture, facilitated by an abstract layer built on blockchain, along with a substantial community of developers, Ethereum showcases significant potential for future growth. Therefore, it has been chosen as the platform for analysis.

[3] https://dappradar.com/blog/dappradar-2019-dapp-industry-review

1.3.2 Stakeholders for such a tool

The individuals and groups interested in a tool for identifying likely-reputable projects on Ethereum would encompass:

Investors and Traders

Participants in the Ethereum ecosystem involved in development and entrepreneurship would find value in a tool that aids in evaluating the reputation and credibility of other projects. It would help them identify trustworthy collaborations and partnerships, fostering a more transparent and reliable Ethereum environment.

Researchers and Analysts

Professionals in the blockchain and cryptocurrency field conducting research and analysis would appreciate a tool that assists in assessing project reputations. It would enhance their research capabilities, enabling them to generate valuable insights and contribute to the collective understanding of reputable Ethereum projects.

Regulatory Authorities

Entities responsible for overseeing the cryptocurrency and blockchain industry would have an interest in a tool that helps identify reputable projects. It would support their efforts to safeguard investors, combat fraud, and ensure compliance with applicable regulations.

These stakeholders have diverse interests and motivations within the Ethereum ecosystem and would all have a vested interest in a tool that assists in identifying likely-reputable projects.

1.4 Aims and Objectives

This research aims to identify likely-reputable projects through a series of transactions that have been conducted by a list of reputable projects from a trust-able source. We may therefore define the following three project aims.

1. Develop a comprehensive methodology for identifying likely-reputable Ethereum Blockchain based projects using transactional data and machine learning techniques. This involves designing a framework that effectively analyzes and classifies project reputability based on transactional patterns with a high accuracy.

In order to address the aforementioned aims, the subsequent objectives have been established and arranged in the order they need to be accomplished;

1. Build 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 LightGBM classification model).
3. Evaluate the performance of the model in detecting likely-reputable and illicit addresses.

The following are the steps in order of execution that need to be taken to fulfil each objective set;

1. Identify addresses associated with reputable projects from a trusted source.
2. Retrieve all relevant transactions conducted by each identified project.
3. Extract additional features from the obtained transactions for each project and extract illicit projects from [3].
4. Create a dataset comprising the newly extracted features per account, categorizing them as likely-reputable or illicit.
5. Utilize an appropriate model to generate results.
6. Perform parameter optimization to enhance the results of the model.
7. Present a rationale for the results obtained from the produced models.

1.5 Contributions

The approach proposed and investigated in this FYP is being prepared for submission to be published in an academic venue under the guidance of the supervisors.

2 Background

This chapter serves as a foundation for understanding the key concepts and context surrounding the topic. It provides an overview of Blockchain technology, Ethereum, tree models and finally the machine learning models used.

2.1 Decentralized Digital Currencies and Blockchain Technology

Although the need for decentralized funds/money has been theoretically examined in the past, it didn't actually materialize until the late 2000s. [7]. In 2008, the advent of Blockchain technology in conjunction with Bitcoin gave rise to a revolutionary technology.

Currently, digital economy primarily relies on trusted authorities. This naturally implies that all online transactions go through the relevant entities, requiring us end uses to place our trust that these transactions have been fulfilled in a secure and private manner. This holds for both financial and non-financial domains.

The introduction of Blockchain established a distributed consensus system in the digital online domain [8]. Due to Blockchain, a public ledger of all transactions or digital activities enacted and distributed among engaging entities is available. This can be better understood as a publicly available database containing all record history available for everyone to see, created with an append-only (thus ensuring immutability) data structure and sustained through a peer-to-peer network [9].By utilizing the distributed consensus algorithm, transactions must be verified by the majority of the entities within the system. Once a transaction has been appended to the ledger, no changes can be carried out - thus providing immutability - made possible through a combination of one-way cryptographic hash functions and the consensus algorithm [10]. However, this can be avoided in the case that an individual or group of individuals has the majority control (over 50%) of the network. Otherwise, the provided ledger is regarded to be both a historical and reconciliated rendition of the ground truth. This allows end users to place their trust in the system, which is further enforced by public/ private key technology without any third party human intervention. Thus, double records, also popularly known as the 'double spending problem', are therefore blocked and stopped from being recorded on the distributed ledger in an automated manner. This leads us to concluding that Blockchain technology benefits are threefold; i) enables deployment of algorithms on top of the existing Blockchain technology to complete business processes or any further suitable uses in the form of smart legal contracts [11], ii) Ease digital payments between entities in a secure and trustworthy

manner and iii) store all records and assets in the form of an immutable database [12].

2.2 Ethereum

Blockchain technology gave rise to many open source platforms, one of which is Ethereum. Developed and launched by Vitalik Buterin back in July 30th 2015, when compared to Bitcoin, Ethereum offers better capability and purpose. While Bitcoin is limited to offering a single application of Blockchain technology - enabling a peer-to-peer electronic payments system for Bitcoin payments and bitcoin ownership, Ethereum tackles several limitations through providing full Turing-completeness through its programming language [13].

Ether is the digital currency or internal crypto-fuel utilized within the Ethereum network. It is worth noting that one Ether is equivalent to 10^{18} Wei, with Wei being the smallest recognized unit of Ether. This enhancement enables Ethereum to accommodate various types of computations and maintain the record of transaction execution states, among other advancements in Blockchain architecture. Ethereum can be viewed as a conceptual layer built upon the underlying Blockchain technology, empowering end users to establish their own regulations regarding ownership rights, transaction formats, and the outcomes of state transition functions through the use of smart contracts. Smart contracts necessitate specific parameters or conditions to be fulfilled before cryptographic rules can be executed.

The Ethereum state is comprised of objects known as accounts, each of which is uniquely identifiable through a 20-byte address and their respective state transitions [14]. Each account holds the following 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 [15] - the data structure used for storing all key-value pairs within the whole Ethereum domain.

Accounts on Ethereum belong to two main categories which are externally owned accounts (EOA) or contract accounts contract acccounts (CA), with the formed being controlled via private keys and the latter through their respective contract code. EOAs can choose to send messages by establishing and signing transactions. Whereas, since CAs contain code, receiving a message triggers the code activation which effectively allows for reading and writing to internal storage, sending other messages or possibly creating new contracts.

Although similar to Bitcoin transactions, messages in Ethereum differ in three

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

In order to ensure that the network is not abused, transaction fees double up as a regulatory mechanism, since issued transactions must be downloaded and verified via consensus [16]. By using the 'proof-of-work' concept, miners are rewarded for a probabilistic computational puzzle once completed through a series of hash function implementations. Requiring vast amounts of computational power, a viable 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 choose to join mining pools having one designated pool operator responsible for distributing the sub-problems among members. According to reports, the top 60 miner pool supplies 90% of the total available power [17].

2.3 Blockchain as Big data and potential applications

Big data is a new technological paradigm that encompasses the high velocity, volume, and diversity of data [18] with which data is being generated [19]. Since big data is not 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 extraordinary 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 mitigate these issues by storing all data in a structured, encrypted and immutable format.

Its ability to maintain accurate data has led to its applicability in Artificial Intelligence (AI) or Machine Learning applications. Conventional analysis methods are limited in use given the quantity of data to be analysed. Bitcoin for example has shown a massive increase in size since its inception, reaching an approximate size of 404 gigabytes as of the beginning of July 2022 [1]. This growth can also be seen in other Blockchain protocols such as Bitcoin cash, Litecoin and Ethereum[2].

Through the adoption of Blockchain services, previously encountered data trust issues by Internet of Things (IoT) developments will be suppressed. Moreover, the

[1] https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/
[2] https://blockchair.com/ethereum/charts/blockchain-size

leading emergence of various IoT applications [20]. Examples include the authentication of interactions among connected vehicles and roadside infrastructures [21] as well as tracing and authenticating the source of goods [22].

2.4 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 in turn negatively impact the overall accuracy of the respective model. Finding a good data representation is ultimately domain specific and dependent on the available measurement values [23].

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, 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.

2.5 Binary Classification

A fundamental and widely employed task in the field of machine learning is the categorization of data instances into one of two distinct classes, typically referred to as positive and negative. The objective is to accurately assign new data points to their respective classes based on the available features. Several algorithms, such as logistic regression, support vector machines, decision trees, random forests, and neural networks, can be utilized to tackle binary classification problems.

The performance of binary classification models is typically evaluated using various metrics, including accuracy, precision, recall, F1 score, and the receiver operating characteristic (ROC) curve. These metrics provide insights into the model's ability to correctly classify instances from both positive and negative classes, and allow for a comprehensive assessment of its performance.

2.5.1 ROC curve, AUC and Confusion Matrix

During testing, classifiers produce a measurable quantity of instances that have been labeled as a correct and incorrect class. These are recorded 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 model's performance, more meaningful measures may be formulated to highlight the

performance of the model in specific scenarios, namely, sensitivity, specificity, and accuracy.

Accuracy is a fundamental metric used to assess the performance of a model. It quantifies the proportion of correct predictions made by the model relative to the total number of examples evaluated.

$$Accuracy = \frac{TP + TN}{TN + FP + FN + TP} \tag{2.1}$$

Another measure used to assess a model's performance is sensitivity, also referred to as **recall** or True Positive Rate (TPR), which determines the proportion of correctly identified real positives.

$$Sensitivity = \frac{TP}{FN + TP} \tag{2.2}$$

Moreover, Precision is also an indicator of a machine learning model's performance and can be described as the quality of a positive prediction made by the model.

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

The last measure Specificity can be defined as the model's ability to predict a true negative of each category available.

$$Specificity = \frac{TN}{TN + FP} \tag{2.4}$$

One can combine Precision and Recall, to formulate another measure namely the F1-Score. It is calculated as the harmonic mean of the precision and recall scores and is defined in 2.5

$$F1Score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{2.5}$$

All measures yield a numerical value within the range of 0 to 1.

By leveraging the concept of specificity, which is calculated as $1 - Specificity$, one can effectively generate the Receiver Operating Characteristic (ROC) curve and measure the Area Under ROC Curve (AUC) [24]. The ROC curve serves as an indicator of a model's ability to differentiate between different classes, demonstrating its capacity for separability. A higher AUC value indicates a corresponding higher level of accuracy in a model.

2.6 Tree Models - Decision Trees, Random Forests and LightGBM

Tree boosting has shown to be highly effective in suitable machine learning applications ranging from smart spam classifiers [25] to the detection of anomaly events in experiments [26] from which new physics may emerge. These applications have only been successful due to their utilization of powerful statistical models and learning systems which are able to scale and learn the relevant model of interest from vast datasets.

Tree models pose a conjecture that the relationship present between the predictors and the response may be defined locally through the constant basis functions [27]. 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

$$f(x) = \sum_{m=1}^{M} \theta_m I(x \in R_m) \tag{2.6}$$

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

LightGBM, short for Light Gradient-boosting machine is a highly efficient gradient boosting decision tree algorithm utilizes the gradient boosting architecture. The evolution of the LightGBM algorithm, for which we provide an expanation, may be seen in the figure below;

Figure 2.1 Evolution of Decision Trees.

The fundamental basis of LightGBM originates from the well-known structured decision tree methodology. In the context of decision-making, a decision tree assigns probabilities to potential choices, creating a device for making informed decisions [28]. Mathematically, this is represented as $P(f|h)$, where f represents a future element or available options, and h corresponds to the historical elements related to the current situation. By asking a series of relevant questions $(q_1, q_2, q_3, ..., q_n)$, the probability $P(f|h)$ can be determined. Each question relies on the previous results obtained, and the size of the decision tree depends on the available training data, consisting of binary

(yes/no) questions. Using a classification tree, a question with multiple values is decomposed into a series of binary questions for each value. By sequentially following these questions from the root node to a leaf node, the correct label or definition of an unclassified object can be determined.

Similarly to boosting, bootstrap aggregating or ”bagging” utilizes the training dataset to generate multiple classifiers [29]. Bagging involves sampling with replacement from the training points to create diverse training sets. This process is repeated for a predetermined number of trials, resulting in new training sets of the same size as the original set. These new sets may contain repeated instances or exclude some examples due to sampling with replacement.

By using these training set samples, an ensemble of classifiers denoted as C^t is generated, and the final classifier C is obtained by aggregating the T classifiers. When classifying a new instance x, the output of each classifier $C^t(x) = k$ is recorded, and the final class output $C^(x)$ is determined through a voting scheme, such as majority voting for multi-classification tasks.

The subsequent layer, random forests, modifies the construction approach of decision or regression trees [30]. Random forests introduce an additional layer of randomness to bagging by assembling each tree using a varied bootstrap sample and considering only a random subset of features for the best split at each node. This helps address issues like overfitting commonly encountered in machine learning.

The algorithm can be further simplified into three stages for both classification and regression. Initially, bootstrap samples are acquired from the training dataset. Subsequently, an unpruned tree is constructed for each sample. However, instead of selecting the optimal split from all predictors, a random subset of predictors is sampled, and the best split is determined from this subset. Bagging takes place when the size of the subset matches the number of predictors. Finally, the final output is obtained by aggregating the predictions from each tree, using majority voting for classification or averaging for regression.

Boosting is a sequential training technique that is integrated into random forest approaches. It involves training a series of base classifiers in a sequential manner, where the error function is influenced by the performance of all the preceding models in the sequence [31]. Instances that were misclassified by previous classifiers are assigned higher weights, which affect subsequent classifiers. Figure 2.2 illustrates the overall architecture of the boosting mechanism.

As shown, each base classifier is trained on a weighted form of the training set, with weights determined by the prior base classifier. After training, the weak classifiers are combined to form the final classifier known as the Strong classifier. AdaBoost, short for Adaptive Boosting, is a widely used implementation of the boosting algorithm in statistical computing languages [33].

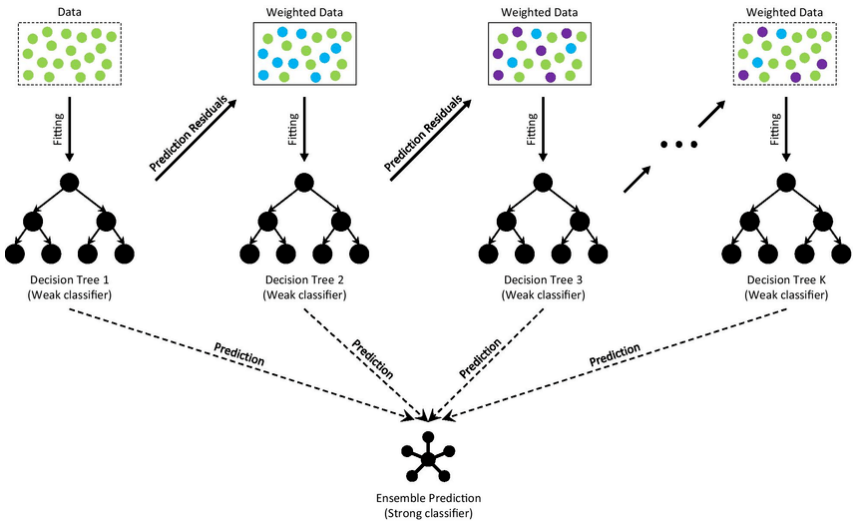


Figure 2.2 Boosting Architecture [32].

An enhancement to the boosting algorithm is *Gradient Boosting*, which applies gradient descent algorithms with boosting techniques to minimize total error iteratively by optimizing parameters in the descending direction [3]. Challenges such as local minima and plateaus are encountered during optimization, where the optimal solution may appear to be reached [34, 35]. The choice of the loss function depends on the scenario, as long as it is differentiable. The learning rate is an important parameter that affects the step size. A low learning rate requires more iterations to find the global minima, while a high learning rate increases the risk of surpassing local minima [36].

LightGBM is a highly efficient and scalable implementation of the gradient boosting framework [37]. It offers several advantages, such as handling missing values, scalability with large datasets, and improved computational efficiency compared to other gradient boosting algorithms. LightGBM is particularly known for its speed, surpassing popular alternatives like XGBoost [38, 39]. This speed can be attributed to two key factors: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB).

GOSS is a sampling technique that selectively retains instances with large gradients, which contribute significantly to the computation of information gain. By dropping instances with small gradients, more accurate estimation of information gain is achieved, enhancing the overall performance of LightGBM [37].

EFB addresses the challenge of feature sparsity, which is common in machine learning applications with large feature spaces. By mapping the optimal feature selection problem to a graph coloring problem, LightGBM efficiently reduces the number of relevant features without significant loss. It employs a greedy algorithm with a constant approximation ratio to solve the problem effectively [37, 40, 41].

[3] `http://uc-r.github.io/gbm_regression`

3 Literature Review

This chapter provides a comprehensive review of the existing literature on the topic of identifying likely-reputable Ethereum blockchain-based projects. It aims to explore the current state of research and highlight the key findings, methodologies, and insights that have been developed in this area.

3.1 Predicting successful Ethereum-based ICOs

In the domain of identifying likely-reputable projects on the Ethereum blockchain, the authors in [42] explore the use of machine learning techniques to predict the success of Initial Coin Offerings (ICO) based on the Ethereum blockchain. ICO refer to a fundraising method used by cryptocurrency and blockchain-based projects. Although the focus of this study is on ICOs, it is still related to the general theme of identifying reputable Ethereum blockchain-based projects using machine learning.

The authors recognize the increasing popularity of ICOs as a fundraising method within the Ethereum ecosystem. However, the high rate of failure and the prevalence of fraudulent ICOs pose significant challenges for investors and regulators. To address this, the study proposes a machine learning-based approach to predict the success of Ethereum-based ICOs.

The research employs various machine learning algorithms, including decision trees, random forests, support vector machines, and artificial neural networks. Features such as social media sentiment, project characteristics, and market variables are considered as inputs to the models. By training and evaluating these models on a dataset of historical ICOs, the study aims to identify patterns and indicators that can predict the success or failure of future ICOs.

While the specific focus is on ICOs, the study's methodology of utilizing machine learning techniques to assess the potential of Ethereum-based projects aligns with the broader objective of identifying likely-reputable projects. By leveraging transactional data and other relevant features, machine learning models can help evaluate the credibility and potential success of Ethereum-based projects, contributing to the identification of reputable ventures in the ecosystem.

3.2 Ponzi Schemes Detection

"Ethereum Smart Contracts Analysis: Detecting Ponzi Schemes" by Zaremba and Al Qudah (2020) [43] focuses on analyzing Ethereum smart contracts to detect Ponzi schemes. While the specific emphasis is on Ponzi schemes, which are a form of

fraudulent activity, the detection methodology discussed in the paper is related to assessing the likelihood of project reputability. The authors propose a machine learning-based approach to analyze Ethereum smart contracts and identify patterns indicative of Ponzi schemes. They extract various features from smart contract transaction data, including transaction frequency, contract balance changes, and the involvement of multiple contracts. These features are then used to train a machine learning model to classify smart contracts as either Ponzi or non-Ponzi.

Although the focus is on detecting Ponzi schemes, the methodology employed in this study can be adapted to identify reputable projects as well. By analyzing transactional data and applying machine learning techniques, patterns and characteristics that indicate reputable projects can be identified. This approach can help differentiate trustworthy ventures from potentially fraudulent ones, contributing to the overall assessment of project reputability on the Ethereum blockchain.

3.3 Detecting Ethereum smart contract vulnerabilities

Another approach titled, “A machine learning-based approach for detecting Ethereum smart contract vulnerabilities,” [44] explores the use of machine learning techniques to identify vulnerabilities in Ethereum smart contracts. The researchers in this study propose a machine learning-based approach to identify potential security flaws in Ethereum smart contracts. They leverage transactional data and various features extracted from smart contracts to train a machine learning model. The model is trained on a dataset of known vulnerable and non-vulnerable smart contracts.

By analyzing the transactional history and features of smart contracts, the model learns patterns and characteristics associated with vulnerable contracts. It can then classify new contracts as either potentially vulnerable or non-vulnerable based on their features. The study demonstrates the effectiveness of the machine learning approach in detecting vulnerabilities and provides insights into potential security risks in Ethereum smart contracts.

Although this particular study focuses on vulnerabilities rather than reputability, it highlights the application of machine learning techniques for analyzing Ethereum blockchain-based projects. The approach of utilizing transactional data and extracting relevant features can be extended to assess the reputability of projects by incorporating additional factors specific to reputability assessment.

4 Methodology

This chapter focuses on the approach and techniques used to identify likely-reputable Ethereum blockchain-based projects using machine learning. The objective of this chapter is to provide a detailed explanation of the steps and methodologies employed to evaluate the reputability of projects within the Ethereum ecosystem

All development was performed on an AMD Ryzen 7 5700U 1.8GHz (up to 4.3GHz) with 8GB available RAM using Python v3.9.12.

4.1 Overview of the proposed solution

To attain the intended outcome, the proposed solution can be divided into four primary stages, illustrated in Fig. 4.1 of the flowchart. These stages encompass: i) Obtaining both ”likely-reputable” and ”illicit” projects. ii) Conducting feature extraction on the transactions associated with each respective account. iii) Employing dimensionality reduction techniques and employing LightGBM and Logistic Regression algorithms to classify the projects. iv) Utilizing various visualizations to present the results in a clear and comprehensible manner. Each step is reliant on the preceding stages and significantly influences the final outcomes produced by the machine learning (ML) algorithms.

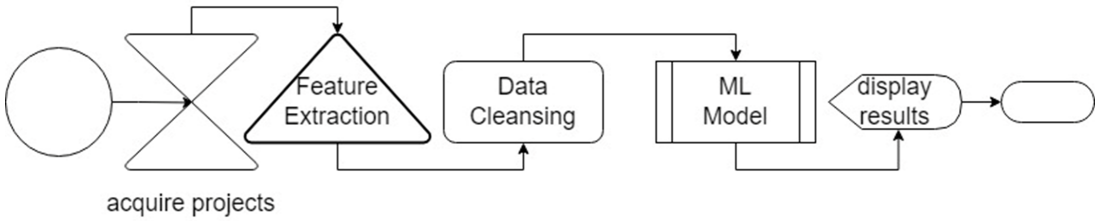


Figure 4.1 Architectural overview of proposed solution

4.2 Data Acquisition

An important factor in evaluating our proposed methodology involved the datasets used. The pseudonymous nature of cryptocurrencies [14, 45, 46] presents challenges in determining the ground truth, particularly in identifying illicit and reputable accounts, resulting in limited data availability. Additionally, acquiring data from comparable domains like traditional financial accounts proved to be difficult due to the sensitivity of the information involved. Extensive research was conducted to identify suitable datasets that align with our research problem. Since Farrugia et al’s dataset has

been highly used as a reference point in related literature for example in [47, 48], we selected the same dataset utilized - *Ethereum Illicit Accounts* [3].

4.2.1 Ethereum Illicit Accounts dataset

The *Ethereum Illicit Accounts dataset* consisted of accounts associated with illicit activities on the Ethereum platform. These accounts were utilized as the non-reputable or illicit set in the analysis. The dataset consisted of a total of 4,681 instances, with 2,179 deemed as illicit and 2,502 as normal accounts. For the purpose of this research, we discard the “normal” accounts (random accounts selected from the Ethereum network which are neither reputable nor illict) as we are only interested in differentiating between illicit and likely-reputable accounts. By using supervised learning, there is no method to verify whether a normal account which is identified as likely-reputable is actually reputable or not. Furthermore, The dataset included a comprehensive set of 42 features derived from the transaction history of specific Ethereum accounts. These features encompassed various aspects such as the Total Ether Balance and Min Value Received. For a detailed description of these features, please refer to Appendix B.

4.2.2 Acquiring the Ethereum Likely-reputable Accounts dataset

The *Likely-reputable Ethereum Accounts dataset* consists of known Ethereum projects which were collected from recommended reputable sources [49] such as CoinGecko. A total of 4,060 smart contracts were collected, these are smart contracts that we identify to be likely-reputable. Once collected, we built the same features proposed by Farrugia et al (to the best ability of available data). In order to build features(such as number of received transactions, sent transactions etc.) for these acquired smart contracts, we utilized the Etherscan API, which provides access to the most recent 10,000 transactions conducted by either a normal account or an ERC20 account (following the ERC20 interface standard). The results can be paginated to further increase the threshold of retrievable transactions per account but as noted from the transactions obtained:

1. 10,000 transactions were sufficient enough to determine an account’s 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 or 3.5 hours per 500 accounts.

Normal Transactions

The most basic type of transactions on the Ethereum network occurs between Externally Owned Accounts (user accounts on the Ethereum blockchain that is controlled by an external entity). These transactions involve the transfer of ether from the sender to the recipient. There are also other types of transactions, such as participating in a Crowdsale to acquire contract tokens in exchange for a specific quantity of tokens [50]. In these cases, the value transferred is often 0, and they are commonly referred to as functions where the end user invokes a 'claim' contract function to receive the corresponding tokens.

More on Normal Transactions and the fields of interest utilised for feature extractions can be found in Appendix B.1.

ERC20 Transactions

These transactions are specifically associated with the transfer of tokens that adhere to the ERC20 interface standard. The ERC20 refers to a set of rules and guidelines that define a common interface for tokens on the Ethereum blockchain. A normal transaction can be successful or unsuccessful, however all listed ERC20 transactions on Etherscan are considered successful thus eliminating the need for transaction filtering. More on ERC20 Transactions and the fields of interst utilised for feature extraction can be found in Appendix B.2.

Once these transactions were acquired, the final dataset consisted of 3,978 accounts due to missing features for some of the smart contracts collected from CoinGecko.

4.3 Feature Extraction

Once the raw data was collected for the likely-reputable set of projects, feature extraction was performed to generate the same features used by Farrugia et al.

The *Ethereum Illicit Accounts dataset* by [3, 51] consists of 42 features per account. Most of these features were recreated for the list of projects from CoinGecko that we identify to be likely-reputable. A subset of the extracted features showed similarities to those obtained in related research utilizing graph analysis [2]. The process of feature extraction was conducted in conjunction with the Etherscan transaction request API, and by performing a big query on the *Ethereum Blockchain* dataset provided by Google [1]. Most of the features were generated by using the Etherscan API however, for features such as total Ether sent and total Ether received,

[1] `https://www.kaggle.com/datasets/bigquery/ethereum-blockchain`

the BigQuery (a web service provided by Google Cloud Platform that offers a fully managed and serverless data warehouse for analyzing large datasets). dataset was used as it encompasses over a larger time frame.

Moreover, since the dataset provided by [3] considers transactions for illicit and normal accounts till June 2019, the same date range was used when collecting data for the likely-reputable Ethereum accounts data-set's set of features to avoid bias in data collection.

Finally, for some features, the Etherscan API returned no value for more than 76% of smart contract addresses from CoinGecko; these features were not included in the final dataset and are shown in Appendix B.

The final dataset contains 6,156 instances of projects where 2,179 of these are illicit and 3,978 likely-reputable projects from CoinGecko. Moreover, the dataset also consists of the 38 features from the 42 features collected in [3]. A complete list of the fields generated and available within the CSV file may be viewed in appendix A.

4.4 Data Visualization

To visualize the final dataset, a technique called t-Distributed Stochastic Neighbor Embedding (t-SNE) was employed, which reduces the dimensionality of the dataset containing 38 dimensions. The t-SNE is a non-linear method which allows visualization of high dimensional data [52]. This is conducted by mapping each data point to a position in a 2 or 3 dimensional space. Specifically, a conversion is carried out from high-dimensional euclidean distances present among the data points to conditional probabilities representing similarities [53]. Thus, given a high dimensional dataset $X = x_1, x_2, x_3, .., , x_n$ a conversion takes place such that the resultant dimensionality $Y = y_1, y_2, y_3, .., , y_n$ may be visually represented as a scatter plot.

The main objective is to maintain the structural integrity within the high-dimensional space while transforming it into a lower-dimensional space, thus ensuring the discovery of any possible valuable insights. 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.

4.5 LightGBM

Prior to model execution, the dataset (with the features extracted explained above) needed to be prepared. The subsets generated during the data preparation stage are utilised by the model for training and testing purposes, producing the resultant

predictions and results. This was followed by parameter tuning and optimisation in order to identify the optimal parameters delivering the best results given the resources available.

4.5.1 Dataset preparation

Before feeding the data into the model, a small statistical analysis was conducted on the dataset in order to identify relationships between the features and remove any redundancies present within the dataset. To identify correlation between the features themselves, the Pearson correlation coefficient was utilized and the features demonstrated in Table B.3 provided in Appendix B shows a strong correlation of more than 95%. Consequently, these features were removed. Through this experiment and results, (**RQ1)** has been answered.

In order to evaluate the performance of the trained model, two well-known data preparation methods were used; 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 model's learning capability in mind. The stratify parameter was set to the output 'FLAG' variable to retain the ratio between illicit and likely-reputable 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.

4.5.2 Model Implementation

LightGBM offers an easy to implement Python library, helping one achieve scalability, portability and accuracy. In order to use LightGBM in Python project, the library was simply imported into the project for use.

The model underwent the following series of steps to generate the intended outcome;

1. Declare a LightGBM 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. Apply the trained model to make predictions on the associated test set and observe the outcomes.

4.5.3 Parameter Optimisation

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

Three key model parameters were assessed, including the learning rate, n_estimators, and max_depth. While there are additional LightGBM parameters that could be investigated, the reviewed literature indicates that these parameters are adequate for the current study..

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).

4.5.4 Performance Measures

Two primary measures have been utilized to determine the relative performance of the LightGBM model with distinctive parameters. Their respective outcomes provide no relative or direct information on the other measure; with the AUC used to identify the optimal model parameters whereas feature importance furthers understanding of the relative features importance and model stability. To further determine the model's performance; accuracy, recall, f1-score and precision were used. Explanation on these is provided in Section 2.5.1

4.5.5 Feature importance

LightGBM offers two primary metrics that quantify and present the importance of all features trained by the model. The method *feature_importance()* returns a matrix with one importance value per feature, and supports two types of importance, based on the value of importance_type. The aforementioned include;

1. **Gain** which specifies the cumulative gain of all splits using this feature
2. **Split** which specifies the number of splits this feature was used in.

To assess the model's stability and reliability, we analyze its performance under different train-testing size scenarios. We examine feature importance based on two metrics across optimal folds and expect some variation in the order of importance. However, a stable metric will maintain a consistent feature ranking. We focus on the top 10 features as the most valuable in this analysis.

5 Evaluation

This chapter provides an evaluation conducted to determine the proposed approach's appropriateness. The results are presented in order by which the implementation is to be assessed; starting with some general dataset information, nonlinear dimensionality reduction t-SNE method, LightGBM model evaluation and the most importand LightGBM features identified.

5.1 General Dataset Information

Once feature extraction was conducted 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. 5.1 highlights the average time difference between received transactions executed by an account with respect to the relative account FLAG i.e., whether an account was flagged as illicit or likely-reputable. This in turn relates to the average duration of activity for the respective accounts.

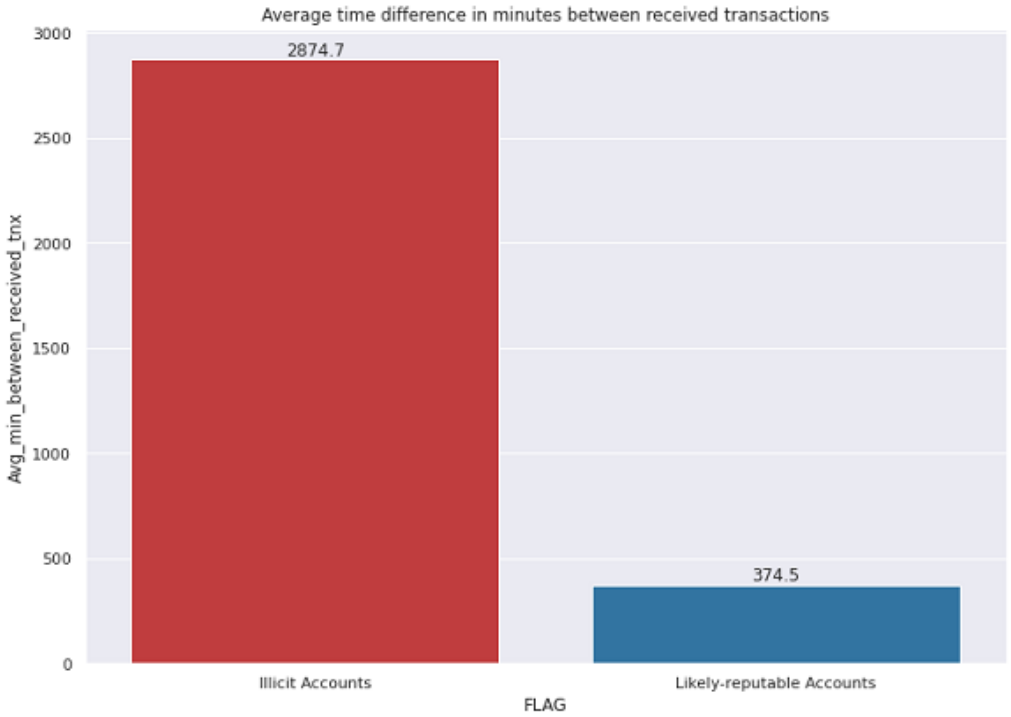


Figure 5.1 Average time difference between received transactions (in minutes) is examined for "likely-reputable" and "illicit" projects, denoted by blue and red, respectively.

To further illustrate the difference in reputable projects from CoinGecko and illicit projects, Fig. 5.2 shows the account balance in Ether for respective accounts. As

expected, likely-reputable accounts on average show a much greater Ether balance as compared to illicit accounts. In the opinion of the author, this tends to be since illicit accounts tend to have their funds withdrawn so that the proceeds can be laundered (i.e., disassociated with the illicit account); and also since they tend to be less successful than reputable projects.

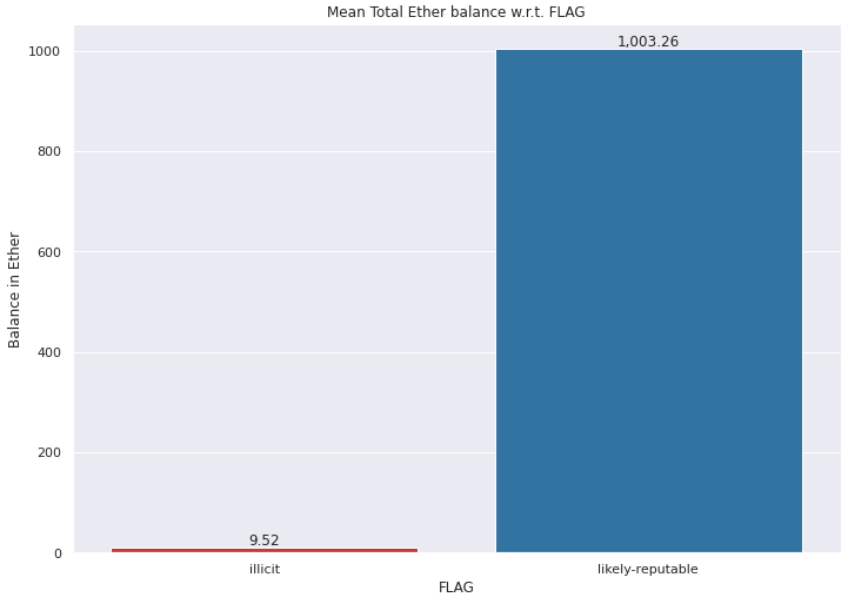


Figure 5.2 Mean total Ether balance with respect to the account flag.

More information on the dataset is provided in Appendix A.

5.2 T-distributed Stochastic Neighbor Embedding (T-SNE)

In order to visualise the data, 2D and 3D-t-SNE plots were generated as seen in Fig. 5.3 and Fig. 5.4 respectively.

The scatter plot displayed in Fig. 5.3 illustrates distinct clusters from both classes, with noticeable features such as three prominent red clusters located on the left side of the plot, and a central concentration of illicit accounts represented by the major blue cluster.. However, there is a small overlap of the two accounts classes which is more noticeable in Fig. 5.4.

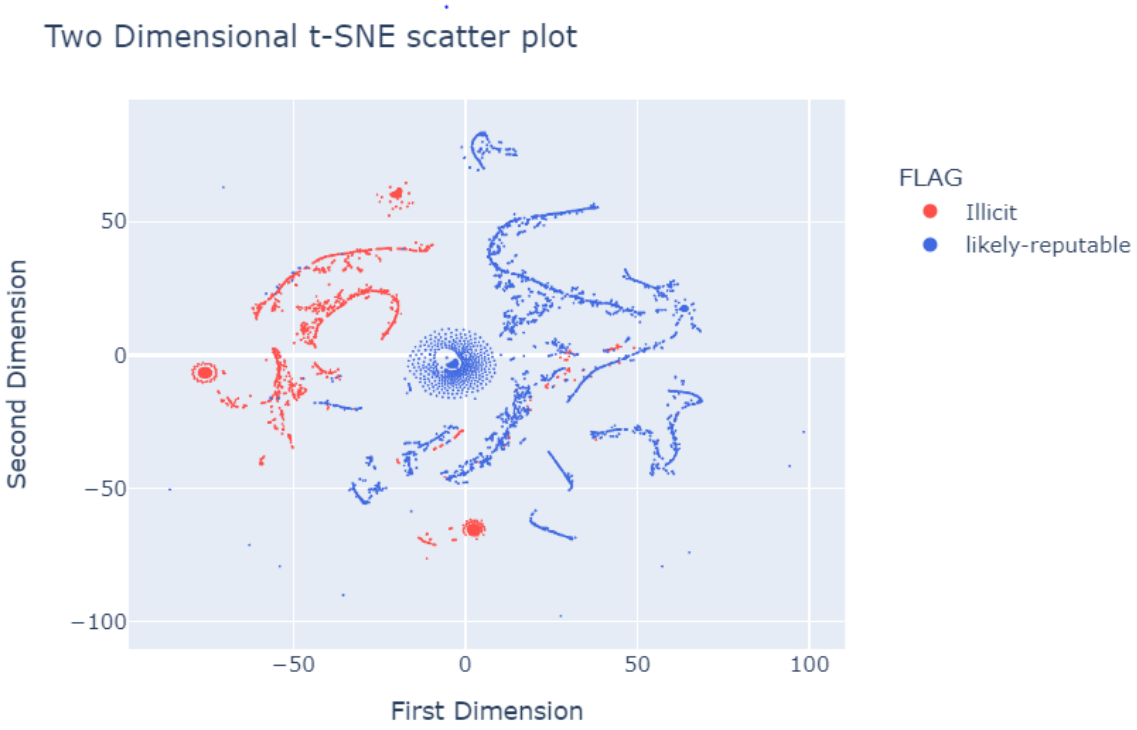


Figure 5.3 Two-dimensional Scatter plot visualizing the distribution of account classes after applying t-SNE transformation.

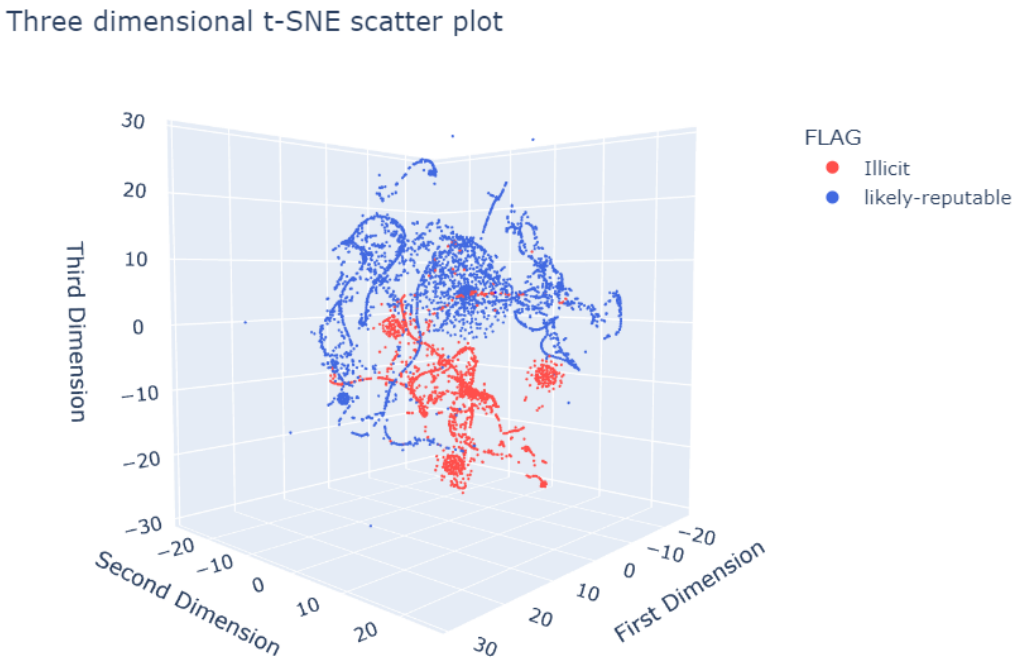


Figure 5.4 Three-dimensional Scatter plot visualizing the distribution of account classes after applying t-SNE transformation.

5.3 LightGBM

In order to determine the optimal values for our key parameters, namely the learning rate, number of trees (n_estimators), and maximum tree depth (max_depth), we employ grid search cross-validation. Additionally, the performance of the model was evaluated using the AUC measure to identify the best model performance based on the given parameter values.

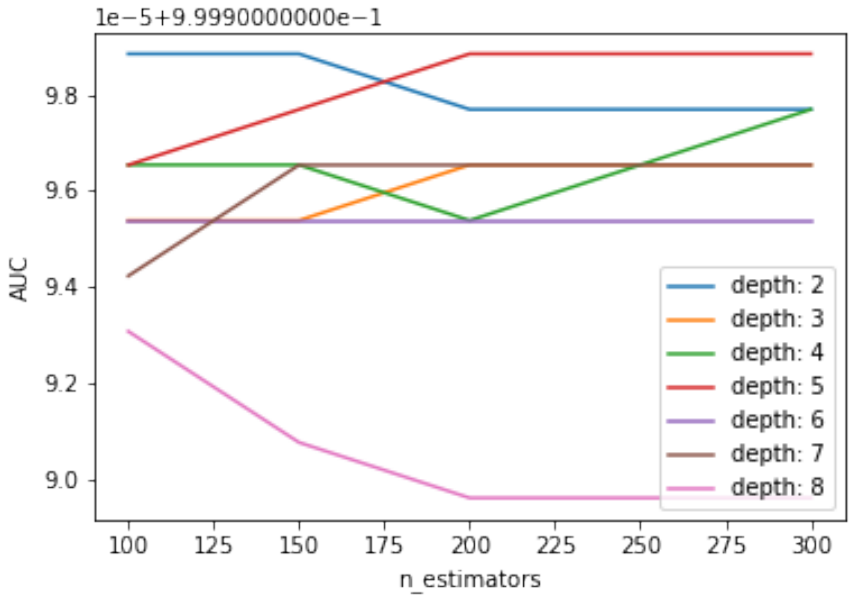


Figure 5.5 Evaluation of n_estimators and max_depth parameters using Grid-Search and 10-fold cross-validation.

The optimal parameters as highlighted by the blue line were identified to be a max depth of 2 and n_estimators 100 which led to a an AUC of **0.999**. The corresponding accuracy for these optimal parameters was **0.9984**. Through this experiment, and results, **(RQ2)** has been answered.

In order to gain deeper insights into the learning capabilities of the model, the logarithmic loss and classification error in relation to the number of iterations performed by LightGBM are presented in Fig. 5.6 and Fig. 5.7 respectively.

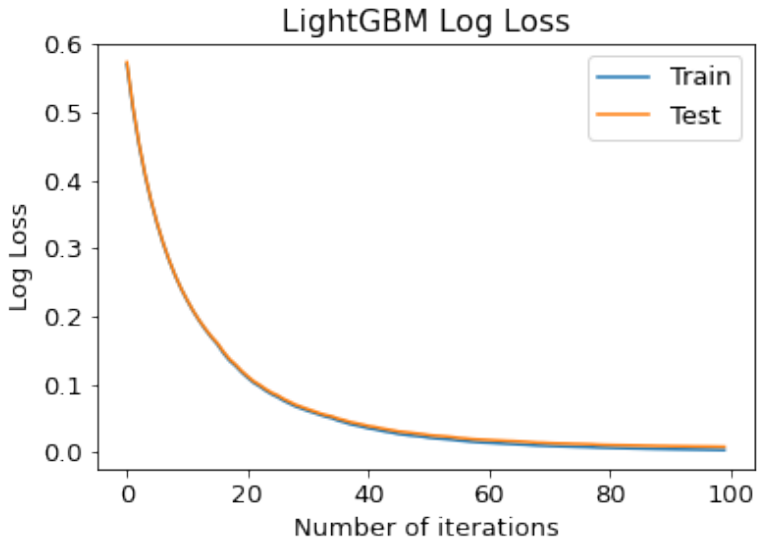


Figure 5.6 The average logarithmic loss, evaluated through 10-fold cross-validation, in relation to the number of iterations of LightGBM.

The logarithmic loss function measures the proximity between the predicted probability and the true value. Additionally, maximizing this likelihood is equivalent to minimizing the mean square error. Based on the illustration, it can be observed that the learning algorithm begins to converge around 100 iterations.

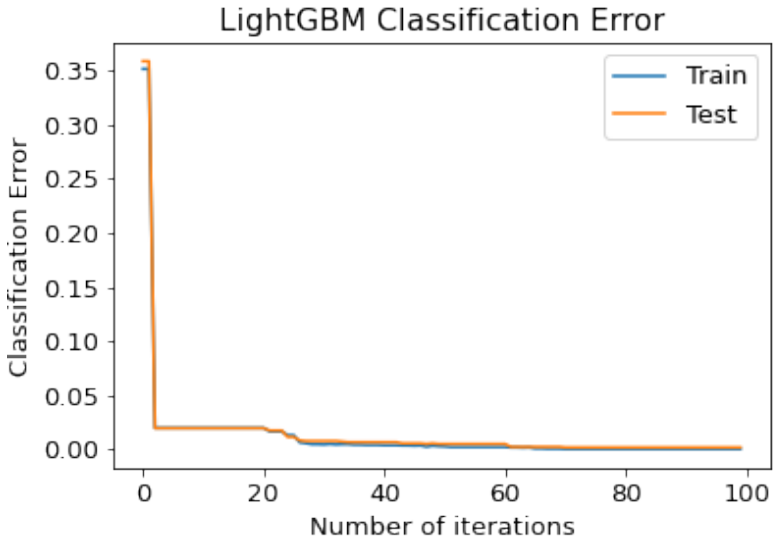


Figure 5.7 The average classification error, evaluated through 10-fold cross-validation, in relation to the number of iterations of LightGBM.

Moreover, the confusion matrix depicted in Figure 5.8 demonstrates that, on average, the model produces 2 False Positives (Type-2 errors) and 1 False Negative

(Type-1 error). These instances represent the projects that the model misclassified as likely-reputable instead of illicit or illicit instead of likely-reputable, respectively. While the misclassification of ”False Negatives” (illicit accounts being identified as likely-reputable accounts) is already explained in [3], the subsequent section provides further insights into aforementioned projects that were inaccurately classified.

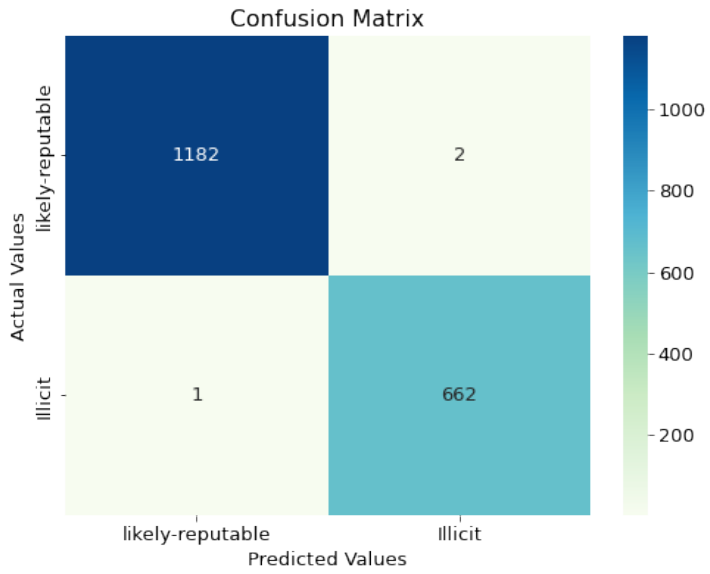


Figure 5.8 Confusion matrix obtained from average 10-fold cross-validation.

5.3.1 Misclassified project information

This section provides information relevant to the misclassified projects, namely the two of False Posiive (FP) and one of False Negative(FN) category. The specific associated accounts were investigated by studying the internal account and transactions details logged in the blockchain.

False Positivies	***False Negatives***
0xaa602de53347579f86b996d2add74bb6f79462b2	0x311f71389e3de68f7b2097ad02c6ad7b2dde4c71
0xc56c2b7e71b54d38aab6d52e94a04cbfa8f604fa	

Table 5.1 False Positivies and False Negatives

False Positivies - Type 1

- **Address** 0xaa602de53347579f86b996d2add74bb6f79462b2
  This Ethereum project corresponds to Zipmex Token[1], a regulated digital Assets Platform where users can buy, sell, and earn interest on digital assets. Though classified as being 'illicit', it is marked as reputable by CoinGecko. However, from Etherscan, it can be seen that the account's balance has always been 0 ETH and the only transactions the account is involved in are incoming token transactions. Moreover, the project was inactive from January 2020 till January 2022 conducting only 7 transactions. Whilst it seems to be the case that the project is indeed illicit, in the opinion of the author, such minimal activity should indeed not be listed as "likely-reputable", since there is minimal transaction information to base a project's reputability off of. Therefore, this false-positive has a valid reason for being marked so.

- **Address** 0xc56c2b7e71b54d38aab6d52e94a04cbfa8f604fa
  This Ethereum project corresponds to ZUSD Token[2]. Similar to Zipmex, the token's balance has maintained a constant value of 0 ETH from 11/12/2019 to 16/05/2023. Furthermore, the account seems to be involved in regular ongoing and incoming transactions however, the maximum number of transactions conducted in any month is 17. Finally, the account is involved in no outgoing transactions. As discussed above, in the opinion of the author, maintaining a zero balance in an account is often seen to demonstrate a project's lack of faith in the project itself and though the the project is listed as 'likely-reputable', this false-positive is reasonably predicted as 'illicit'.

False Negatives - Type 2

- **Address** 0x311f71389e3de68f7b2097ad02c6ad7b2dde4c71
  This project was misclassified to be likely-reputable even though it is illicit. The project is marked as Ponzi [3] on Etherscan. The account was involved in 84,181 transactions with the last transaction being made on 2023/03/051. Moreover, the account had a peak Ether balance of 7,694.31 ETH. Similar to most likely-reputable projects, this account address was not involved in any outgoing transactions and only incoming transactions, with all of these transactions being involved with other unique contract addresses perhaps to only conduct illicit activity with these unique addresses. In the opinion of the author, illicit actors tend to drain balances from such accounts and therefore, the fact that this

[1] https://etherscan.io/token/0xaa602de53347579f86b996d2add74bb6f79462b2
[2] https://etherscan.io/token/0xc56c2b7e71b54d38aab6d52e94a04cbfa8f604fa
[3] https://www.investor.gov/protect-your-investments/fraud/types-fraud/ponzi-scheme

account is leaving the funds in the account may be a reason why it is incorrectly marked as likely-reputable.

Through these results and justifications, (**RQ3)** has been answered.

5.3.2 Probability of being illicit and likely-reputable

For investors to place their own confidence in this model and make a decision whether they should invest in a given project, we provide a probability of being illicit and likely-reputable for each project in the dataset.This probability was generated using LightGBM's *predict_proba* function and is provided in the CSV file [4].

Furthermore, such scores should be presented together and made clear to users that these are indicators that help identify how similar projects are to previous ones that have been flagged as illict or likely-reputable, but ultimately investors would need to filter this information to make an informed decision, and the information in isolation does not definitively mean that a project marked as illicit is illicit, and neither one marked as likely-reputable is reputable – Yet, this is standard practice with respect to such financial indicators. Finally, by providing this probability, 2 new features namely, 'prob_illicit' and 'prob_reputable' which refer to probability of being illicit and/or likely-reputable respectively are supplementary features extracted from the existing data. Thus, answering (**RQ4)**.

5.3.3 Feature Importance

As discussed in section 4.5.4, we use a number of train-test splits to ascertain the stability of the features with respect to the models being built. We observe that 'Split' provides similar feature importance order, especially the top 10 features. The complete ranking is provided in Appendix B.

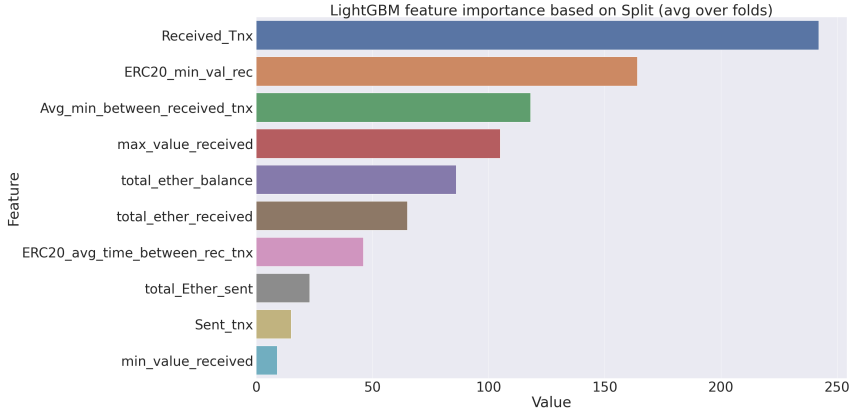


Figure 5.9 LightGBM Feature Importance based on 'Split' parameter.

[4] https://github.com/malikcyrus/eth-reputability-detection

Finally, the following table presents the classification report for the LightGBM model. Note that support refers to the number of actual occurrences of the class (likely-reputable or illicit) in the dataset.

	Precision	**Recall**	**f1-score**	**support**
Likely-reputable	0.9992	0.9992	0.9992	1184
Illicit	0.9970	0.9994	0.9982	663
accuracy			0.9984	1847
macro average	0.9981	0.9993	0.9987	1847
weighted average	0.9988	0.9988	0.9988	1847

Table 5.2 Classification Report of LightGBM model

5.4 Logistic Regression Implementation

We further assess the results obtained by a basic 'Logistic Regression model'. An average accuracy and AUC of **0.9827** and **0.9854** respectively over 10-fold cross-validation with stratification on the flag class. The table below demonstrates the performance of this model.

	Precision	**Recall**	**f1-score**	**support**
Likely-reputable	0.9991	0.9966	0.9979	1168
Illicit	0.9941	0.9985	0.9963	679
accuracy			0.9827	1847
macro average	0.9966	0.9976	0.9971	1847
weighted average	0.9973	0.9973	0.9973	1847

Table 5.3 Classification Report of Logistic Regression

From table 5.3 and the accuracy obtained by this model, we do not perform any further optimization and place our focus on the LightGBM model implemented. Thus answering (**RQ5)**, proving that LightGBM can effectively identify likely-reputable projects with an impressive accuracy.

6 Conclusion

Reputability over projects in the blockchain is crucial for investors to place their trust in aforementioned projects in order for investors to be aware of how trustworthy or reputable a given project is. However, Other studies in this field approach the issue from different angles and primarily focus on detecting illicit behavior within the blockchain network. For instance, [4] examine the identification of Ponzi schemes disguised as smart contracts, detection of anomalous transactions related to illegal activities, and uncovering money laundering schemes [3].

Though these strategies make valuable contributions to identifying illicit activities within blockchain networks, they specifically look at illicit behaviour and do not provide information with respect to a project's potential reputability and do not provide a comprehensive assessment of project reputability, especially when considering individual accounts, with a significant degree of precision.

By integrating diverse concepts and methodologies from related fields, particularly feature extraction and feature importance, we propose an innovative approach to assess the reputability of projects on the Ethereum network at an individual account level. This research utilizes the advanced LightGBM classification model. By employing a well-balanced dataset consisting of both ”illicit” and ”likely-reputable” accounts, the model demonstrated exceptional effectiveness, yielding an average AUC value of 0.999 and a standard deviation of 0.0003 across 10-fold cross-validation. It is worth noting that the model exhibited robust generalization capabilities and displayed resilience against overfitting. This observation is supported by the low standard deviations observed in both AUC and accuracy, as well as the consistent performance achieved even with a small number of k-folds during cross-validation. Despite efficiency not being the primary focus, the minimal resources utilized underscore the scalability of the system.

When examining the rank of feature importance, the average time difference between received transactions is the most influential factor in account classification.. Investigating this further, we observe that the average value for this particular feature is 165.5 seconds for likely-reputable accounts and 47 minutes for illicit accounts. This finding implies that likely-reputable accounts tend to receive multiple incoming transactions from unique addresses thus there is a huge discrepancy between this value for illicit and likely-reputable accounts. Additionally, it is worth noting that likely-reputable accounts refrain from engaging in Ether transactions, unlike illicit accounts. Additionally, likely-reputable accounts exhibit a significantly higher average balance of 1,003.28 Ether, whereas illicit accounts maintain an average balance of 9.82 Ether. Additionally, our analysis indicates that the average amount of Ether sent to

contracts and the involvement of addresses in ERC20 token transactions through standard contracts have less impact as features.

6.1 Limitations

Since the EtherscanAPI provides a maximum of 10,000 transactions per account, the model was trained only on these 10,000 transactions which was sufficient enough to produce the level of performance attained, however, it is unknown how acquiring all transactions for a given project could have an affect on the results achieved. Furthermore, this will come at a cost of time (more transactions require more time to retrieve and generate features).

Some of the accounts which are still listed on CoinGecko are inactive for a few months such as mentioned in section 5.3.1 which though are listed as likely-reputable, appear to be more illicit. And finally, the number of available document scams is rather limited in size; increasing only when the Ethereum community identifies a potential scam.

6.2 Future work

This study provides a foundational framework for future research in a related field. Potential avenues for further exploration include examining the applicability of our proposed approach in identifying likely-reputable projects on different blockchain networks. It would be useful to be able to build models to detect illicit and likely-reputable projects that are cross-blockcahin (E.g., some projects have tokens and smart contracts on different blockchains). Additionally, investigating the feasibility of adapting the described methodology to identify reputability at a granular ”transactional level” could be another promising direction for future investigations.

In addition, an alternative approach to assessing reputability involves analyzing the smart contract code of Ethereum projects to identify shared features among reputable projects. This information can be used to develop a model that incorporates these features and explores other potential indicators for improved classification results. Additionally, the analysis can extend beyond transactions recorded on the Ethereum blockchain to include internal transactions generated through contract execution. These additional features can be incorporated into the LightGBM model to enhance its classification capabilities.

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

References

[1] M. Rudlang, “Comparative analysis of bitcoin and ethereum,” M.S. thesis, NTNU, 2017.

[2] T. Chen *et al.*, “Understanding ethereum via graph analysis,” *ACM Transactions on Internet Technology (TOIT)*, vol. 20, no. 2, pp. 1–32, 2020.

[3] S. Farrugia, J. Ellul, and G. Azzopardi, “Detection of illicit accounts over the ethereum blockchain,” *Expert Systems with Applications*, vol. 150, p. 113 318, 2020.

[4] S. Yu, J. Jin, Y. Xie, J. Shen, and Q. Xuan, “Ponzi scheme detection in ethereum transaction network,” in *Blockchain and Trustworthy Systems: Third International Conference, BlockSys 2021, Guangzhou, China, August 5–6, 2021, Revised Selected Papers 3*, Springer, 2021, pp. 175–186.

[5] S. Sayadi, S. B. Rejeb, and Z. Choukair, “Anomaly detection model over blockchain electronic transactions,” in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, IEEE, 2019, pp. 895–900.

[6] V. Buterin, “Ethereum: Platform review,” *Opportunities and challenges for private and consortium blockchains*, vol. 45, 2016.

[7] D. Vujicic, D. Jagodic, and S. Randic, “Blockchain technology, bitcoin, and ethereum: A brief overview,” *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2018. DOI: 10.1109/infoteh.2018.8345547.

[8] M. Crosby, P. Pattanayak, S. Verma, V. Kalyanaraman, *et al.*, “Blockchain technology: Beyond bitcoin,” *Applied Innovation*, vol. 2, no. 6-10, p. 71, 2016.

[9] N. Atzei, M. Bartoletti, and T. Cimoli, “A survey of attacks on ethereum smart contracts (sok),” in *Principles of Security and Trust: 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings 6*, Springer, 2017, pp. 164–186.

[10] T. Aste, P. Tasca, and T. Di Matteo, “Blockchain technologies: The foreseeable impact on society and industry,” 2017.

[11] C. D. Clack, V. A. Bakshi, and L. Braine, “Smart contract templates: Foundations, design landscape and research directions,” *arXiv preprint arXiv:1608.00771*, 2016.

[12] S. Ammous, “Blockchain technology: What is it good for?” *Available at SSRN 2832751*, 2016.

[13] C. Dannen, *Introducing Ethereum and solidity*. Springer, 2017, vol. 1.

[14] V. Buterin *et al.*, “A next-generation smart contract and decentralized application platform,” *white paper*, vol. 3, no. 37, pp. 2–1, 2014.

[15] H. S. de Ocáriz Borde, “An overview of trees in blockchain technology: Merkle trees and merkle patricia tries,” 2022.

[16] L. Luu, Y. Velner, J. Teutsch, and P. Saxena, “Smart pool: Practical decentralized pooled mining.,” in *USENIX Security Symposium*, 2017, pp. 1409–1426.

[17] A. Fernández Anta *et al.*, “Miner dynamics on the ethereum blockchain,” in *Workshop on Complex Sociotechnical Systems*, 2019.

[18] D. Laney *et al.*, “3d data management: Controlling data volume, velocity and variety,” *META group research note*, vol. 6, no. 70, p. 1, 2001.

[19] I. Lee, “Big data: Dimensions, evolution, impacts, and challenges,” *Business horizons*, vol. 60, no. 3, pp. 293–303, 2017.

[20] K. Rabah, “Convergence of ai, iot, big data and blockchain: A review,” *The lake institute Journal*, vol. 1, no. 1, pp. 1–18, 2018.

[21] A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak, “Blockchain: A distributed solution to automotive security and privacy,” *IEEE Communications Magazine*, vol. 55, no. 12, pp. 119–125, 2017.

[22] S. Saberi, M. Kouhizadeh, J. Sarkis, and L. Shen, “Blockchain technology and its relationships to sustainable supply chain management,” *International Journal of Production Research*, vol. 57, no. 7, pp. 2117–2135, 2019.

[23] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*. Springer, 2008, vol. 207.

[24] T. Fawcett, “An introduction to roc analysis,” *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[25] K. N. Reddy and V. Kakulapati, “Classification of spam messages using random forest algorithm,”

[26] J. Vanerio and P. Casas, “Ensemble-learning approaches for network security and anomaly detection,” in *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, 2017, pp. 1–6.

[27] L. A. Clark and D. Pregibon, “Tree-based models,” in *Statistical models in S*, Routledge, 2017, pp. 377–419.

[28] D. M. Magerman, “Statistical decision-tree models for parsing,” *arXiv preprint cmp-lg/9504030*, 1995.

[29] L. Breiman, “Bagging predictors,” *Machine learning*, vol. 24, pp. 123–140, 1996.

[30] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[31] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.

[32] H. Deng, Y. Zhou, L. Wang, and C. Zhang, "Ensemble learning for the early prediction of neonatal jaundice with genetic features," *BMC medical informatics and decision making*, vol. 21, pp. 1–11, 2021.

[33] Y. Freund and R. E. Schapire, "Adaptive boosting," *Machine Learning*, vol. 23, no. 2, pp. 321–357, 1996.

[34] C.-N. Wang, F.-M. Hsu, and C.-J. Chang, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 58, no. 3, pp. 225–254, 2004.

[35] M. S. Mitchell, "Discipline without punishment: The PROACT approach to discipline," *Discipline Without Punishment: The ProAct Approach to Discipline*, vol. 7, no. 3, pp. 12–13, 2006.

[36] M. A. Alawal, "Understanding learning rates and how it improves performance in deep learning," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 2, pp. 181–187, 2018.

[37] G. Ke *et al.*, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in Neural Information Processing Systems*, vol. 30, pp. 3146–3154, 2017.

[38] W. Zhang, J. Wei, W. Zhou, Z. Sun, X. Yao, and M. Deng, "A comparison of gradient boosting machines and random forest for real-time landslide susceptibility mapping," *Geosciences*, vol. 10, no. 11, p. 447, 2020.

[39] H. Al-Khalidi, L. Sun, Z. Yang, H. Yuan, and J. G. Tchoupo, "A comparison of gradient boosting machines and deep learning for real-time control of supercritical co2 brayton cycles," *Applied Sciences*, vol. 9, no. 22, p. 4954, 2019.

[40] C. Seger, *An investigation of categorical variable encoding techniques in machine learning: Binary versus one-hot and feature hashing*, 2018.

[41] P. M. Pardalos, T. Mavridou, and J. Xue, "The graph coloring problem: A bibliographic survey," *Handbook of Combinatorial Optimization: Volume1–3*, pp. 1077–1141, 1998.

[42] A. S. Alzahrani and A. M. Khedr, "Predicting successful ethereum-based initial coin offerings using machine learning techniques," *PLOS ONE*, vol. 15, no. 4, e0231659, 2020.

[43] M. Zaremba and D. Al Qudah, “Ethereum smart contracts analysis: Detecting ponzi schemes,” *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, pp. 360–365, 2020. DOI: 10.14569/IJACSA.2020.0110148.

[44] M. Khodadadi, A. H. Lashkari, and A. Dehghantanha, “A machine learning-based approach for detecting ethereum smart contract vulnerabilities,” *Computers & Security*, vol. 92, p. 101 706, 2020. DOI: 10.1016/j.cose.2020.101706.

[45] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system bitcoin: A peer-to-peer electronic cash system,” *Bitcoin. org. Disponible en https://bitcoin. org/en/bitcoin-paper*, 2009.

[46] D. Schwartz, N. Youngs, A. Britto, *et al.*, “The ripple protocol consensus algorithm, 2014,” *URL: https://ripple. com/files/ripple_consensus_whitepaper. pdf*, 2014.

[47] M. S. Rahman, R. Islam, M. M. Islam, and K. M. Alam, “Blockchain forensics: A comprehensive survey, challenges, and open issues,” *IEEE Access*, vol. 9, pp. 21 869–21 890, 2021.

[48] R. K. Rai, P. P. Neema, A. K. Misra, and M. M. Hassan, “Blockchain-based fraud detection in cryptocurrencies: A systematic literature review,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 7, pp. 8483–8501, 2021.

[49] D. Vidal-Tomás, “Which cryptocurrency data sources should scholars use?” *International Review of Financial Analysis*, vol. 81, p. 102 061, 2022.

[50] B. Skvorc, *Ethereum: Internal transactions & token transfers explained*, Jul. 2018. [Online]. Available: https://www.sitepoint.com/ethereum-internal-transactions-token-transfers/.

[51] J. Hirshman, Y. Huang, and S. Macke, “Unsupervised approaches to detecting anomalous behavior in the bitcoin transaction network,” in *Technical report*, Stanford University, 2013.

[52] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne.,” *Journal of machine learning research*, vol. 9, no. 11, 2008.

[53] W. E. Arnoldi, “The principle of minimized iterations in the solution of the matrix eigenvalue problem,” *Quarterly of applied mathematics*, vol. 9, no. 1, pp. 17–29, 1951.

Appendix A In-depth Information on Dataset

figures A.1 and A.2 illustrate the discrepancy in average Ether sent or received. As shown by figure A.1, on average, illicit are more involved in sending of Ether across accounts whereas likely-reputable accounts collected from CoinGecko are not involved much over past 10,000 transactions in sending Ether.

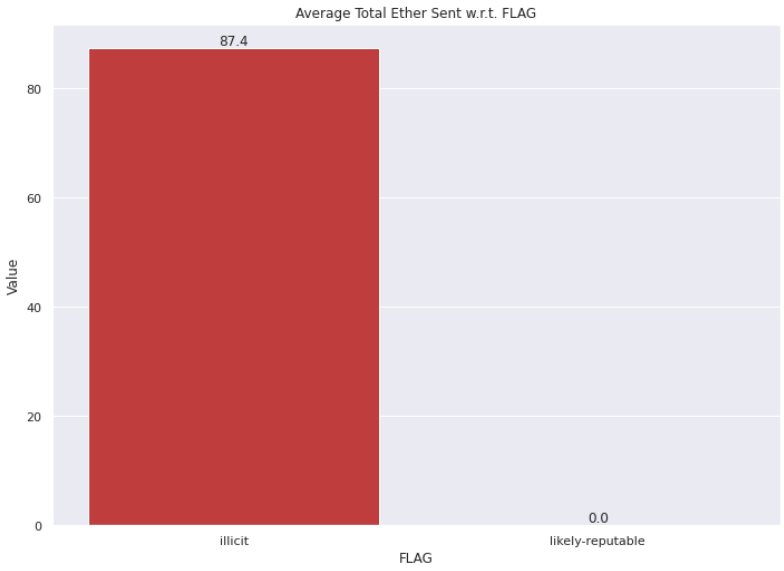


Figure A.1 Average amount of Ether sent with respect to the account class

On the other hand, likely-reputable accounts receive a lot more Ether in comparison to illicit projects. This can further be attributed to the fact that reputable projects such as Tether, Binance, BNB etc. are typically invested in and used by multiple users.

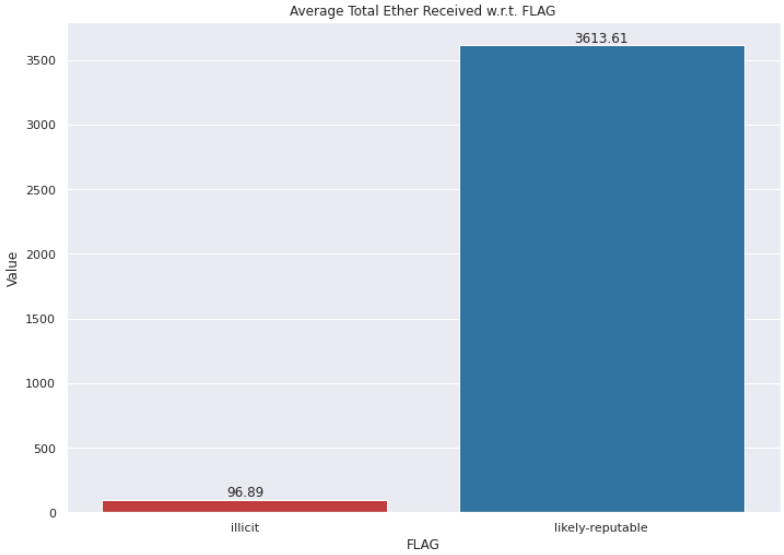


Figure A.2 Average amount of Ether sent with respect to the account class.

To study further this difference in activity, we look at the number of sent and received transactions conducted by an account with respect to its class as shown in figures A.3 and A.4.

As demonstrated by the figure below, illicit accounts are more involved in sending transactions from an account to another as compared to a likely-reputable account.

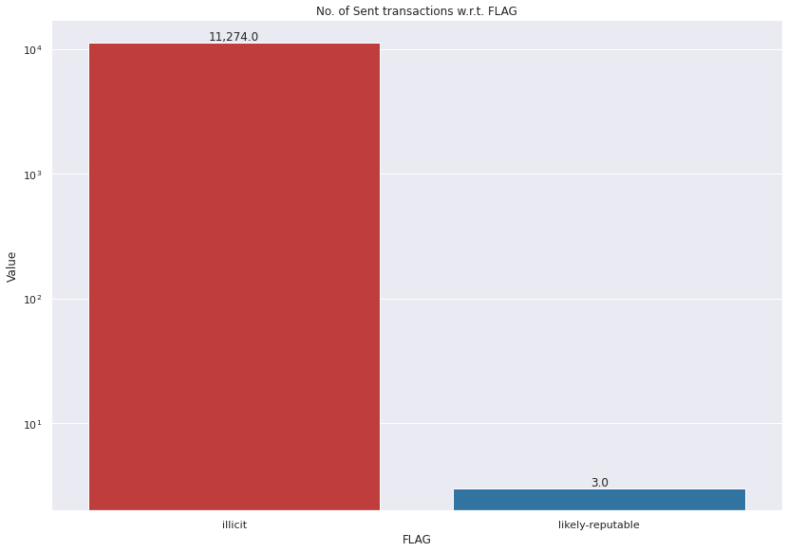


Figure A.3 Total Number of sent transactions w.r.t. Flag.

We note that on average, likely-reputable accounts receive a larger quantity of transactions along with Ether as compared to illicit accounts.

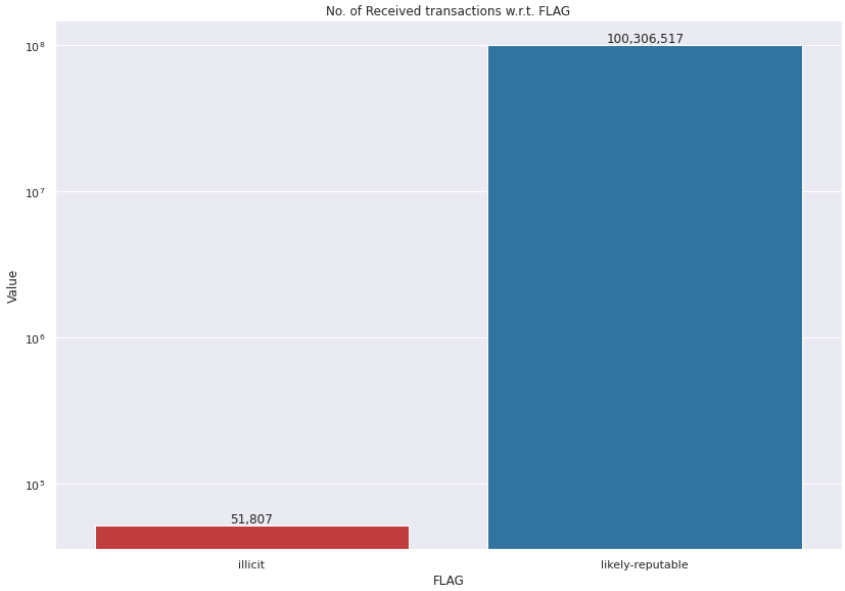


Figure A.4 Total Number of received transactions w.r.t. Flag.

A.1 Feature Importance

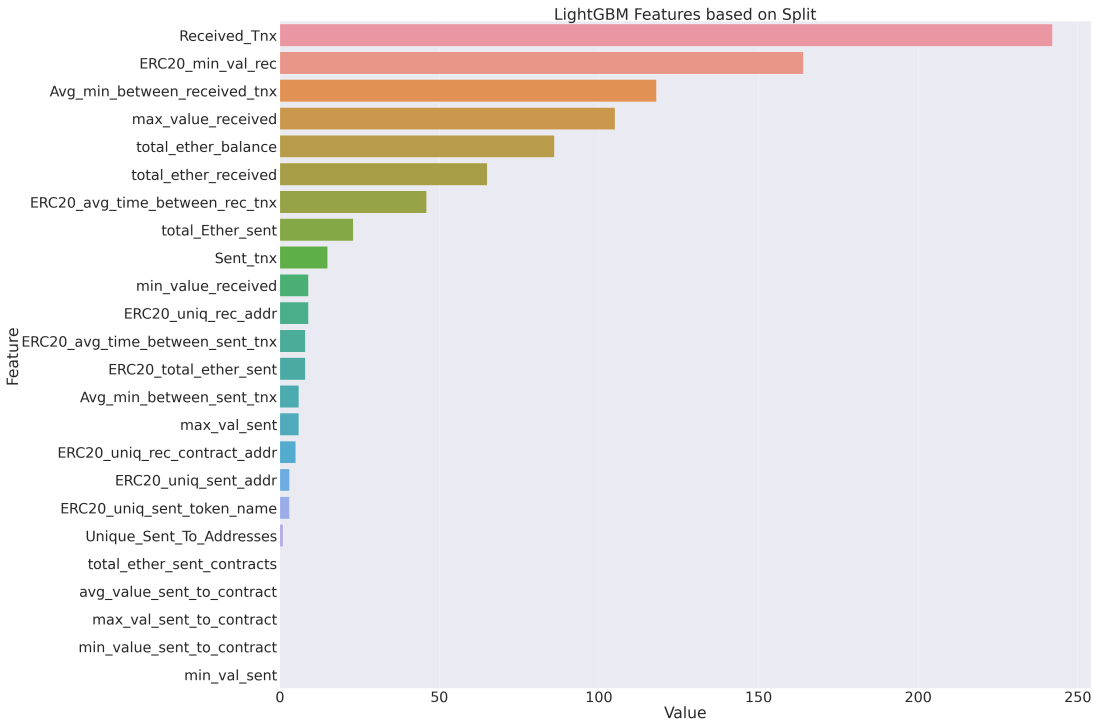


Figure A.5 Feature Importance based on the split parameter over average 10-cross-fold

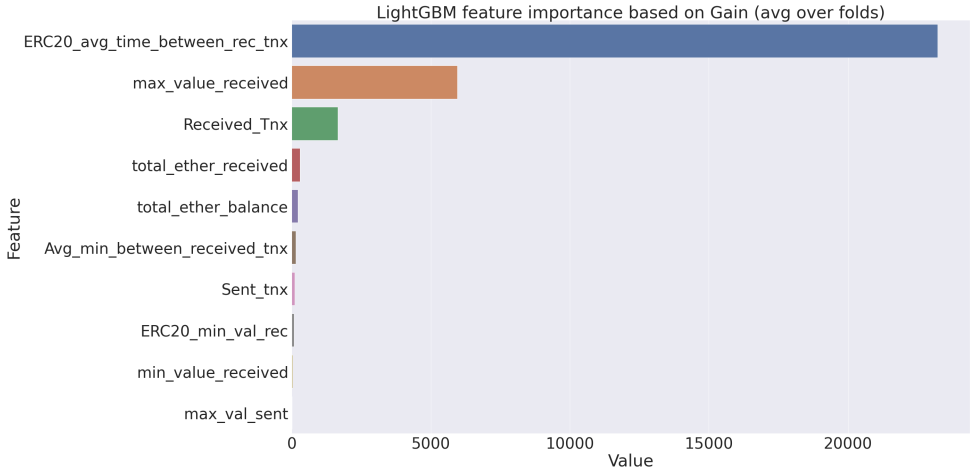


Figure A.6 Feature Importance based on the gain parameter over average 10-cross-fold

Appendix B Features of Dataset Utilized

	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

Table B.1 Complete list of features taken from [3]

Feature	*Description*
Time_Diff_between_first_and_last_(Mins)	Time difference between the first and last transaction
Number_of_Created_Contracts	Total Number of created contract transactions
total_transactions_(including_tnx_to_create_contract)	Total number of transactions
Total_ERC20_tnxs	Total number of ERC20 token transfer transactions
ERC20_min_val_sent_contract	Minimum value in Ether received from ERC20 token transactions
ERC20_max_val_sent_contract	Maximum value in Ether received from ERC20 token transactions
ERC20_avg_val_sent_contract	Average value in Ether received from ERC20 token transactions
ERC20_most_sent_token_type	Most sent token for account via ERC20 transaction
ERC20_most_rec_token_type	Most received token for account via ERC20 transaction

Table B.2 Missing features in likely-reputable projects dataset

Feature	*Description*
Unique_Received_From_Addresses	Total Unique addresses from which account received transactions
ERC20_total_Ether_received	Total ERC20 token received transactions in Ether
ERC20_total_Ether_sent_contract	Total ERC20 token transfer to other contracts in Ether.
ERC20_avg_time_between_contract_tnx	Total ERC20 token transfer to other contracts in Ether.
ERC20_max_val_rec	Maximum value in Ether received from ERC20 token transactions for account
ERC20_avg_val_rec	Average value in Ether received from ERC20 token transactions for account
ERC20_min_val_sent	Minimum value in Ether received from ERC20 token transactions for account
ERC20_max_val_sent	Maximum value in Ether sent from ERC20 token transactions for account
ERC20_avg_val_sent	Average value in Ether sent from ERC20 token transactions for account
ERC20_uniq_rec_token_name	Number of Unique ERC20 tokens received

Table B.3 Features with a high a correlation of more than 95%

B.1 Normal Transactions

Normal transactions are identified by the presence of an 'ISERROR' field in normal transactions. This field indicates whether a transaction was executed without any issues. Unsuccessful transactions can occur for various reasons, such as insufficient funds, exceeding the gas limit, or providing an incorrect recipient account address. In this study, our focus is solely on the transactions that were successfully executed, which is determined by verifying that the 'ISERROR' field is set to 0. Any transactions with a non-zero value in the 'ISERROR' field are excluded from our analysis. 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 B.4 Normal transaction fields of interest

B.2 ERC20 Transactions

ERC20 transactions play a crucial role in the Ethereum ecosystem as they enable the seamless transfer and exchange of tokens between different users and decentralized applications (dApps) built on the Ethereum platform. These transactions are recorded on the Ethereum blockchain, providing a transparent and immutable ledger of token transfers. 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	New address for contract created	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 B.5 ERC20 transaction fields of interest

Appendix C Python Libraries

Library	Version
etherscan-python	2.1.0
google-api-core	2.11.0
google-auth	2.16.2
google-cloud-bigquery	3.6.0
google-cloud-core	2.3.2
lightgbm	3.3.5
matplotlib	3.5.1
numpy	1.21.5
pandas	1.4.2
pingouin	0.5.3
seaborn	0.11.2
sklearn	1.0.2

Table C.1 Python Libraries Utilized