University of Groningen

Detection of illicit accounts over the Ethereum blockchain

Farrugia, Steve; Ellul, Joshua; Azzopardi, George

Published in:
Expert systems with applications

DOI:
[10.1016/j.eswa.2020.113318](https://doi.org/10.1016/j.eswa.2020.113318)

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2020

[Link to publication in University of Groningen/UMCG research database](#)

Citation for published version (APA):

Farrugia, S., Ellul, J., & Azzopardi, G. (2020). Detection of illicit accounts over the Ethereum blockchain. *Expert systems with applications*, 150, [113318]. <https://doi.org/10.1016/j.eswa.2020.113318>

Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Detection of illicit accounts over the Ethereum blockchain

Steven Farrugia^a, Joshua Ellul^{b,c,*}, George Azzopardi^a

^a Faculty of Science and Engineering, University of Groningen, Bernoulliborg, Nijenborgh 9, 9747AG, Groningen, Netherlands

^b Centre for Distributed Ledger Technologies, University of Malta, Msida, MSD 2080, Malta

^c Department of Computer Science, University of Malta, Msida, MSD 2080, Malta

ARTICLE INFO

Article history:

Received 1 October 2019

Revised 24 January 2020

Accepted 16 February 2020

Available online 17 February 2020

MSC:

00-01

99-00

Keywords:

Blockchain

Ethereum

Fraud detection

Machine learning

XGBoost

ABSTRACT

The recent technological advent of cryptocurrencies and their respective benefits have been shrouded with a number of illegal activities operating over the network such as money laundering, bribery, phishing, fraud, among others. 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 (± 0.006) with an average AUC of 0.994 (± 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 value received'. Based on the results we conclude that the proposed approach is highly effective in detecting illicit accounts over the Ethereum network. 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 compiled data set as a benchmark for future related works.

© 2020 Elsevier Ltd. All rights reserved.

1. Introduction

1.1. Detection of illegal activity

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 (Godspower-Akpomiemie & Ojah, 2019). 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 (Levi, 2002).

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 (Bartoletti, Carta, Cimoli, & Saia, 2020). 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 (Bartoletti et al., 2020).

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 to-

* Corresponding author.

E-mail addresses: joshua.ellul@um.edu.mt (J. Ellul), g.azzopardi@rug.nl (G. Azzopardi).

tal of 0.1 ETH (Garg, 2019). Such activity has raised concerns on the actual intentions of the executed transactions; which raises the question on whether they were 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.

1.2. Ethereum

The recent shift towards decentralised digital assets has led to a rapid development in cryptocurrencies seeking to offer something other than just transferring funds between accounts. Vitalik Buterin earlier in 2013 proposed the Ethereum blockchain, which addressed this desire through the introduction of smart-contracts (Rudlang, 2017). When coupled with enhanced security through hash functions, decentralisation of computational logic and the largest ensemble of developers in the cryptocurrency domain (Porru, Pinna, Marchesi, & Tonelli, 2017), Ethereum manifests the potential for widespread applicability.

Ethereum, like other cryptocurrency networks, has been subject to a number of entities/users carrying out illegal activity over the network, some of which includes smart-Ponzi schemes, phishing, money laundering, fraud and crime related activity (Juels, Kosba, & Shi, 2016). Even though 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. 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 (Ermakova, Fabian, Baumann, Izmailov, & Krasnova, 2017). This has inadvertently led to the development of methodologies and techniques to detect such operations.

The distributed ledger made publicly available by the Ethereum blockchain network supports more than a total of 400M transactions and 7.3M blocks as of March 2019.¹ 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 (Bartoletti et al., 2020). An example of such deployed contracts include smart contract Ponzi schemes - solely created with the intent of benefiting the contract creator and possibly a few users who invested in the scheme initially (Iyer & Dannen, 2018). 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 (Vasek & Moore, 2019). This may all be done while the identity of the aforementioned smart contract creator remains anonymous.

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 (Meng, Tschhäuser, Wang, Wang, & Han, 2018). 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 naturally suggest the implementation of machine learning (ML) techniques to help identify any patterns related to anomalous behavior.

Given that labelled data was available for this work and keeping in mind the computational resources available, future scalability, ability to rank the features by order of their importance and necessity to achieve a high level of accuracy, we utilize a supervised learning classifier known as Extreme Gradient Boosting (XGBoost). The XGBoost classifier, through the use of boosting, is capable of integrating a high quantity of tree models with weak performance (base classifiers) resulting in a remarkably accurate classification model (Chen, Jiang, et al., 2018). With automatic parallel computation capabilities, XGBoost has shown to be 20 times faster than gradient boosting machines (GBM) with the same architecture (Chen, He, Benesty, Khotilovich, & Tang, 2015). This is not to mention the additional efficiency achieved through translating data into a D-matrix format during the pre-processing stage. The algorithm is also fully capable of scaling to billions of instances while keeping the required resources sparse relative to other existing systems (Chen & Guestrin, 2016). During a machine learning competition made available by Kaggle in 2015, XGBoost was used throughout more than half of the winning solutions (Zhang et al., 2018).

1.3. Practical implications and contributions

Whilst in this work we have focused on demonstrating the applicability of such algorithms towards the detection of illicit activity (given the dataset available which provides for known illicit accounts), this work could be applied to specific finer grained illicit activities, such as detecting money laundering specifically (this would require such a dataset to be built and/or be available).

The application of the work presented herein can be used by investigative and enforcement agencies and authorities to detect illicit activity. More so, it can be applied to any specific finer grained activity, say to combat money laundering and funding of terrorism. It is not just a *nice to have* feature, but a mandate by the European Union to enforce and combat such illicit activity including money laundering and funding of terrorism – even if using cryptocurrencies (or 'virtual' currencies).² Besides the work that regulators, enforcement and investigative agencies must undertake, due diligence on transactions must be undertaken by all subject persons associated with such economic activity including banks, corporate and financial service providers, and any intermediaries connected to the setup, oversight or operational aspects of associated transactions. Banks and cryptocurrency exchanges are required to undertake such due diligence to ensure that their clients' transactions are in no way related to such illicit activity – and could be held liable if they fail to detect such activity. Similar requirements are being laid out by the Financial Action Task Force (FATF)³ which aim to mandate global implementation.

We contribute to this area of research by investigating whether illicit accounts may be detected based on their transaction history using the XGBoost classification model. The proposed mechanism may be utilized by government entities, developers, researchers, news agencies as well as private entities/corporations.

¹ <https://etherscan.io/>.

² https://ec.europa.eu/info/policies/justice-and-fundamental-rights/criminal-justice/anti-money-laundering-and-counter-terrorist-financing_en.

³ <http://www.fatf-gafi.org/media/fatf/documents/recommendations/RBA-VA-VASPs.pdf>.

2. Related work and XGBoost model background

2.1. Related work

In 2013, Henderson et al. (2012) using K-means and Role Extraction (RoLX) were able to identify users which conducted transactions over the Bitcoin network in an abnormal manner which is typically associated with money laundering. Having acquired all transactions enacted from the networks inception till May 2013, 15 features pertaining to interactions and transaction values were extracted on accounts having a minimum of 650 transactions. Following 1000 iterations of random cluster sizes, $k = 5$ was deemed to be the optimal number of groups with respect to the error produced. The mean node degree and variance node degree, determined by accounts which transacted with one another, were identified as the most two important features.

Delving deeper into the 5 main clusters, the least common contained a large variance in transaction amount. This implied that these accounts participated in transactions worth small values up to thousands of bitcoins. Using RoLX, the authors were able to provide a visual depiction of the account interactions in an apparent money laundering scheme, highlighting how the bitcoins are recursively split and directed to two new addresses.

Seeking to mitigate illegal activity over the Ethereum network through the identification of Ponzi schemes deployed as smart contracts, Chen, Zheng, Ngai, Zheng, and Zhou (2019) classified accounts based on their operation code (Opcode) using supervised learning algorithms. Following parameter tuning, the superior Random Forest was able to classify 305 out of 394 identified smart-Ponzi schemes with a probabilistic certainty of over 90% - taking into consideration that these may include experimentally deployed contracts.

Analyzing 200 smart contracts, 'CREATE' was identified as the least used opcode, appearing in only 20 of them. This 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. Considering the recall and precision achieved by their model, the authors estimate that over 500 smart-Ponzi schemes are available on the Ethereum network.

In order to detect fraudulent activity on the Ethereum network, OKane (2018) examined transactions enacted by 4000 scams available on Etherscamdb along with token and exchange addresses. The scam addresses, falling under the phishing scam category, were either the sender or receiver of a transaction. The most prevalent was identified as the OmiseGo site asking for private keys. In less than 36h, between 5:50 p.m. on the 9th and 11:30 p.m. on the 10th 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 '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.

2.2. XGBoost Model background

XGBoost is a decision-tree ensemble Machine Learning (ML) algorithm which utilizes the gradient boosting architecture. It's foundation stems from the well-known structured *decision tree* methodology. Bootstrap aggregating or *bagging*, similar to boosting, exploits the training dataset available so as to generate varying classifiers (Quinlan, 1996). In simple terms, bagging generates diverse training sets through sampling with replacement from the available training examples - also known as bootstrapping. Using

the training set samples, the learning systems allows for an ensemble of classifiers. A given test sample can then be processed by all classifiers and the decisions are aggregated, for instance, by majority vote.

The next 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 (Liaw, Wiener et al., 2002), 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. Last but not least, the final output provided new data is obtained by aggregating the predictions provided by each 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 (Bishop, 2006). The error function corresponding to the training of a specific model 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.

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 (Boehmke, 2018). Given the loss function, gradient boosting seeks to minimize the overall error through measuring the local gradient of the loss function for the available set of parameters, iteratively tweaking them in the direction of the descending gradient.

Short for Extreme Gradient Boosting, XGBoost offers a scalable and efficient implementation of the aforementioned gradient boosting framework (Chen et al., 2015). 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 (Torlay, Perrone-Bertolotti, Thomas, & Baci, 2017) and (v) rank the features by order of their importance.

These benefits are mostly attributed to a number of system and algorithmic optimizations (Chen et al., 2015). System improvements are primarily attributed to three main enhancements which include (i) parallelizing the building of the XGBoost across multiple threads for an improved overall computation time, (ii) pruning backwards using the max-depth parameter to limit the tree depth instead of the greedy stopping criterion which is used in Gradient Boosting Models (GBM) to determine the features to be split on (tree spitting) and (iii) optimizing hardware utilisation of resources - specifically cache use for storing gradient related data. The algorithm also introduces a number of enhancements, namely (i) automatically catering for different variations in sparsity patterns presently available within the dataset in an efficient

Table 1
Normal transaction fields of interest.

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

manner, (ii) out-of-the-box cross-validation to enhance the models capability and reduce development time as well as (iii) penalizing more complex models via L1 and L2 regularization to avoid overfitting.

3. Methodology

3.1. Ethereum account dataset

The dataset was obtained through a combination of two sources; the Etherscan database and a local Geth client connected to the Ethereum network for scam and normal accounts, respectively.

Accounts were flagged by the Ethereum community for illicit behavior for a number of cases, some of which include (i) trying to imitate other contract addresses providing tokens, (ii) scam lotteries, (iii) fake initial coin offerings (ICO), (iv) imitating other users, (v) Ponzi schemes, (vi) phishing and (vii) mirroring websites. All illicit accounts listed on the 17th of April 2019 were collected from Etherscan database,⁴ a publicly available dataset containing community provided Ethereum account IDs deemed to have carried out illicit activity, via a GET Request, parsing the addresses available in the returned JSON object. A total of 2179 fraudulent accounts were acquired. No preprocessing was necessary during the data preparation stage since no missing values and no transformations were required. The full list of attributes is listed in the Appendix.

Using the functional tool developed by Sokolowska (2018), we randomly selected 3000 normal accounts which conducted transactions during block range 3,800,000 and 3,805,000.⁵ After filtering out non-unique accounts, the final count was that of 2502 unique ones. These accounts were active during the period of mid-July 2017 - a period which saw an increase in Ethereum network usage. All accounts were cross-referenced with the illicit account IDs to ensure that no documented illicit activity has been executed by any of the obtained accounts.

All normal and illicit account IDs, resulting in a total of 4681 accounts combined, were then passed to the Etherscan application programming interface (API) to obtain all the corresponding transactions the account engaged in. The API supported the retrieval of the latest 10,000 transactions carried out by an ERC20 contract account (CA) (CA following the ERC20 interface standard) or externally owned account (EOA). The API takes the following structure for (1) normal transactions and (2) internal transactions, respectively;

1. <http://api.etherscan.io/api?module=account&action=txlist&address=0xddbd2b932c763ba5b1b7ae3b362eac3e8d40121a&startblock=0&endblock=999999999&sort=asc&apikey=YourApiKeyToken>.
2. <http://api.etherscan.io/api?module=account&action=txlistinternal&address=0x2c1ba59d6f58433fb1eae7d20b26ed>

[83bda51a3&startblock=0&endblock=2702578&sort=asc&apikey=YourApiKeyToken](http://api.etherscan.io/api?module=account&action=txlist&address=83bda51a3&startblock=0&endblock=2702578&sort=asc&apikey=YourApiKeyToken).

The API parameter *address* was set to the 4681 account identifiers collected for which the respective transactions were retrieved for all accounts. Features were extracted on all transactions executed by an account prior to being appended to the CSV file for model classification.

The simplest form of transactions within the Ethereum network are between EOA. These executed transactions involve the transfer of Ether from the sender to the recipient (Cao, Zhang, Liu, Zhang, & Neri, 2019). Other plausible transactions may involve contributions towards a Crowdsale of contract tokens in exchange for a quantity of the respective tokens (Rouhani & Deters, 2017). 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 the lack of necessary funds, exceeding the gas limit or incorrect recipient account address (Sturm, Schnig, Szalanczi, & Jablonski, 2019). 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. For these successful transactions, we acquire the respective fields identified in Table 1 in order to carry out feature extraction.

ERC20 transactions are solely related to the transfer of tokens which support/fall under the ERC20 interface standard (Ikeda & Hamid, 2018). 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. Acquiring the transaction fields mentioned in Table 2 per transaction, allowed for feature extraction to take place for the provided account IDs.

A total of 42 features were extracted per account using the transactions gathered.⁶ A small subset of extracted features were similar to those acquired in similar work using graph analysis (Chen, Zhu, et al., 2018; Hirshman, Huang, & Macke, 2013). Feature extraction was carried out in conjunction with the Etherscan transaction request API; generating the respective fields for each account. We have made the extracted dataset and code publicly available⁷

3.2. Data visualization and ML algorithms

Figs. 1 and 2, corresponding to the non-linear transformation applied to the original feature space using t-SNE, were generated with respect to the account class label (normal or illicit) showing the 2D and 3D t-SNE plots, respectively.

⁴ <https://etherscan.io/scams>.

⁵ Original code: <https://bit.ly/2zGhpps>.

⁶ Complete list is available in the Appendix, Table A1.

⁷ <https://bit.ly/32pmC2A>.

Table 2
ERC20 token transaction fields of interest.

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

Fig. 1. 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. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

From the 2D scatter plot in Fig. 1, 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° tilt representation in a three-dimensional scatter plot, Fig. 2 provides an alternative representation of the 'illicit' account clusters which are evident in Fig. 1. 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.

This was then followed by the implementation of the XGBoost classification model involving a sequential three-step process; (i) declaring the model along with all the respective parameters to be considered, (ii) training the classifier on the training set and (iii)

predicting on the test data using the model trained in the previous step. Cross-validation was used to assess the model's performance.

We assessed the following three main XGBoost parameters using grid-search; (i) learning rate, (ii) the number of estimators ($n_estimators$) and (iii) the maximum depth (max_depth).

The best parameters were determined using the Receiver Operating Characteristic (ROC) curve which presents us with the area under the curve (AUC) measure (Bradley, 1997). Classifiers during the prediction phase 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 model's performance, more meaningful measures may be formulated to highlight the performance of the model in

Three-dimensional t-SNE scatter plot

Fig. 2. 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. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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 the performance of a classification model. In simple terms, it's the number of correct predictions with respect to all the available examples being tested.

$$\text{Accuracy} = \frac{TP + TN}{CP + CN} \quad (1)$$

where $CN = TN + FP$ and $CP = FN + TP$.

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 (Altman & Bland, 1994), respectively.

$$\text{Sensitivity} = \frac{TP}{CP} \quad (2)$$

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

$$\text{Specificity} = \frac{TN}{CN} \quad (3)$$

Using the specificity to calculate the false positive rate (FPR) which is equal to $1 - \text{Specificity}$, we are able to plot the ROC curve

Table 3

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.

# of Folds	Optimal Depth	Optimal # estimators	Accuracy	F1-score	AUC	Execution time (s)
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

and obtain the AUC measure (Fan, Upadhye, & Worster, 2006). 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 the curve (AUC) (Choi et al., 2018). 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 model's performance, we deem the AUC as the preferred measurement for identifying the optimal model of XGBoost with different configurations.

3.3. Feature importance

XGBoost provides three main importance metrics to quantify the importance of all features trained on by the model, namely; (i) Weight, (ii) Gain and (iii) Cover. Weight specifies the number of times a particular feature crops up in the trees defined by the model (Shi, Wong, Li, Palanisamy, & Chai, 2019). Gain corresponds to the relative improvement in accuracy achieved by a feature to the branches it forms part of. Cover or coverage relates to the number of observations with respect to the feature in question. The resultant output is a matrix; with the first column listing the features and the other columns defining the importance value with regards to the importance metric provided.

Since we are also interested in the overall model rigidity/stability, we also assessed how the model performed when provided with varying training/testing sizes. Therefore, we analyze how the features vary in their rank with respect to the three importance metrics on the train-test splits executed.

4. Results

4.1. 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. Table 3 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. From Fig. 3 we may note that the boosted trees performance is sub-optimal for combinations of low depth and *n_estimators* when assessed over 10-folds.

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 Figs. 4 and 5, respectively. 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

Fig. 3. Grid-Search 10-fold cross-validation evaluation of *n_estimators* and *max_depth* parameters.

Fig. 4. Average logarithmic loss over 10-fold cross-validation sets with respect to the number of XGBoost iterations.

and the target function. From the two figures we may note that the learning algorithm is converging after approximately 100 iterations.

The confusion matrix in Fig. 6, produced from the execution of 10-fold cross-validation, states 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 the model classified as normal instead of illicit or illicit instead of normal, respectively. In retrospect, although not ideal, we are mostly interested in ensuring that illicit accounts are detected; therefore the 'False Negatives'.

Comparing the three feature importance parameters mentioned in Section 3.3, we conclude that the 'Weight' parameter delivers

Table 4

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

False Positives (FP)	False Negatives (FN)
0x4ba0166ed4522e4e8452fad4850e82e6f905bb64	0x4ff4c15af13f6d4950448104d9b97387a03da6f9
0xafc8a3fb05567fcca7b88fcced6886434850e1bf	0x15f4a5d5bbc071fc20bd60b9b7d81aaa8a1142ab
0xa38d8109127028d3e1774413ef28db70c471cb54	0x0d4f74c538613ed6e6c8c1bc8896ecfd45f5ef23
0x5dc9378cd4be85b41699e9cdf25cc669c0696880	0x49311a81f5f5df6c109599ed5d7413d85b8bca18
0xcc6443b0f0a8cf107cb3350e8eae4b771158e652	0xbbeebe66b6a511dd2accbf7f258554dbe97f1294
0x1c3f580daaeac2f540c998c8ae3e4b18440f7c45	0x2d791afb30b48a0ac39e6b416d2f46aa095b1704
	0xd4c9f4dbd26ae8c9b87a0d9b3368c0096241a162

Fig. 5. Average classification error over 10-fold cross-validation with respect to the number of XGBoost iterations.

True Label	Normal	245	6
	Illicit	7	211
		Normal	Illicit
		Predicted label	

Fig. 6. Average 10-fold cross-validation confusion matrix.

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 9th or 10th usually, they were present in the top 15 features.

The feature pertaining to the time difference between the first and large transaction consistently held the largest frequency score. This was followed by the 'Total Ether balance' field determined by the total available Ether value present in the account following the enacted transactions as well as the minimum value in Ether received (Fig. 7).

4.2. Incorrectly classified accounts

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. Using the Etherscan.io search functionality we may present readily available analytics and information on any existing Ethereum 20-byte account address.

4.2.1. False negatives - Type 2

- '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).
- '0x15f4a5d5bbc071fc20bd60b9b7d81aaa8a1142ab' Account was said to be a phishing scam for a startup called OneLedger. The account was active for 53 days between February 8th, 2018 and April 3rd, 2018 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.
- '0x0d4f74c538613ed6e6c8c1bc8896ecfd45f5ef23' Account was said to have partaken in a Phishing scam concerning the Origin Protocol (allowing for a decentralized marketplace for tokens⁸). 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 h on the 14th April, 2018.

4.2.2. False positives - Type 1

- '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 - having been 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.
- '0xAfc8a3fb05567fcca7B88FccEd6886434850E1Bf' In all a total of 12 transactions were executed; with one transaction incoming and 11 outgoing. All transactions were made to one-time addresses up until all the initial balance of 3.9964476 Ether purchased on June 1st, 2017 was distributed among the 11 unique accounts.
- '0xa38d8109127028d3E1774413ef28Db70C471CB54' A contract account which participated in 144 transactions from the 8th June 2017 till the 27th of June 2017. Titled as PaymentForwarder, the contract has seen payments above 1000 Ether (transaction hash: 0x3888e8a643a0d0b5e3240872d8714aabcfc67c8edb6e142e50e04360d0141893) in a single transaction. The majority of all executed transactions were from/to new accounts.

⁸ <https://www.originprotocol.com/en>.

Fig. 7. The 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.

5. 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 number of created contracts by an account as well as the number of unique address to which tokens were sent via ERC20 standard contracts were the least important features.

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).

Account level detection mechanisms, however, may fall short of detecting smart contract computational logic that attempts to hide its activity from such algorithms. Therefore, future work should be undertaken to perform dynamic analysis of smart contract code in aim of detecting illicit smart contracts that both are static in nature, as well as those that aim to hide away from such detective measures. Such future work is complementary to the work undertaken in this paper.

The relevant authoritative bodies are also placing a higher emphasis on activity at an account level. Authorities may also like to focus on certain types of accounts and the activity of such accounts; (i) potentially looking into dormant account activity, (ii) acquisition streams and distribution of funds and (iii) comparisons with other accounts which exhibit similar transaction patterns. Retrospectively transaction level analysis can not justify or provide sufficient reasoning towards this, and may therefore be an alternate/supportive approach to detect such illicit/abnormal activity.

6. Conclusion

Based on the numerous tests conducted, we deduce that the illicit behavior over the Ethereum network at an 'account level' may be detected efficiently and effectively via the XGBoost classification model that takes input from past transactions and account properties. The proposed approach also presents us with insight on the importance of the input features. The time difference between the first and last transaction, the total available Ether balance and the minimum value in Ether received by an account were established to be the most important three features. Moreover, the proposed approach is conceptually simple and may be adapted to other similar problems.

Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Credit authorship contribution statement

Steven Farrugia: Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization, Funding acquisition. **Joshua Ellul:** Conceptualization, Methodology, Formal analysis, Resources, Writing - original draft, Writing - review & editing, Supervision. **George Azzopardi:** Conceptualization, Methodology, Formal analysis, Resources, Writing - original draft, Writing - review & editing, Supervision, Project administration.

Acknowledgments

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.

Appendix A

Table A1
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	Integer
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

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.eswa.2020.113318.

References

- Altman, D. G., & Bland, J. M. (1994). Diagnostic tests. 1: Sensitivity and specificity. *BMJ: British Medical Journal*, 308(6943), 1552.
- Bartoletti, M., Carta, S., Cimoli, T., & Saia, R. (2020). Dissecting Ponzi schemes on Ethereum: Identification, analysis, and impact. *Future Generation Computer Systems*, 102, 259–277.
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
- Boehmke, B. (2018). Gradient boosting machines. http://uc-r.github.io/gbm_regression.
- Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7), 1145–1159.
- Cao, S., Zhang, G., Liu, P., Zhang, X., & Neri, F. (2019). Cloud-assisted secure health systems for tamper-proofing EHR via blockchain. *Information Sciences*, 485, 427–440.
- Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining KDD '16* (pp. 785–794). New York, NY, USA: ACM.
- Chen, T., He, T., Benesty, M., Khotilovich, V., & Tang, Y. (2015). Xgboost: extreme gradient boosting (pp. 1–4). R Package Version 0.4-2.
- Chen, T., Zhu, Y., Li, Z., Chen, J., Li, X., Luo, X., ... Zhang, X. (2018). Understanding ethereum via graph analysis. In *IEEE INFOCOM 2018 - IEEE conference on computer communications* (pp. 1484–1492).
- Chen, W., Zheng, Z., Ngai, E. C., Zheng, P., & Zhou, Y. (2019). Exploiting blockchain data to detect smart Ponzi schemes on Ethereum. *IEEE Access*, 7, 37575–37586.
- Chen, Z., Jiang, F., Cheng, Y., Gu, X., Liu, W., & Peng, J. (2018). Xgboost classifier for DDoS attack detection and analysis in SDN-based cloud. In *2018 IEEE international conference on big data and smart computing (bigcomp)* (pp. 251–256).
- Choi, H.-S., Kim, S., Oh, J. E., Yoon, J. E., Park, J. A., Yun, C.-H., & Yoon, S. (2018). 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 BCB '18* (pp. 111–121). New York, NY, USA: ACM.
- Ermakova, T., Fabian, B., Baumann, A., Izmailov, M., & Krasnova, H. (2017). Bitcoin: Drivers and impediments. *SSRN Electronic Journal*.
- Fan, J., Upadhye, S., & Worster, A. (2006). Understanding receiver operating characteristic (ROC) curves. *Canadian Journal of Emergency Medicine*, 8(1), 19–20.
- Garg, P. (2019). Mistake or money laundering scheme? Someone pays \$450,000 in fees to send 0.1 eth.
- Godspower-Akpomimie, E., & Ojah, K. (2019). Money laundering, tax havens and transparency. *Enhancing Board Effectiveness*, 248–266. doi:10.4324/9781315169477-15.
- Henderson, K., Gallagher, B., Eliassi-Rad, T., Tong, H., Basu, S., Akoglu, L., ... Li, L. (2012). Rolx: Structural role extraction & mining in large graphs. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining*.
- Hirshman, J., Huang, Y., & Macke, S. (2013). Unsupervised approaches to detecting anomalous behavior in the bitcoin transaction network. *Technical report*. Stanford University.
- Ikeda, K., & Hamid, M.-N. (2018). Chapter four- Applications of blockchain in the financial sector and a peer-to-peer global barter web. In P. Raj, & G. C. Deka (Eds.), *Blockchain technology: Platforms, tools and use cases*. In *Advances in Computers: Vol. 111* (pp. 99–120). Elsevier.
- Iyer, K., & Dannen, C. (2018). *Ponzi and pyramids* (pp. 143–170). Springer.
- Juels, A., Kosba, A., & Shi, E. (2016). The ring of gyges. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security - CCS16*. doi:10.1145/2976749.2978362.
- Levi, M. (2002). Money laundering and its regulation. *The ANNALS of the American Academy of Political and Social Science*, 582(1), 181–194.
- Liaw, A., Wiener, M., et al. (2002). Classification and regression by randomforest. *R News*, 2(3), 18–22.
- Meng, W., Tischhauser, E. W., Wang, Q., Wang, Y., & Han, J. (2018). When intrusion detection meets blockchain technology: A review. *IEEE Access*, 6, 10179–10188.
- OKane, E. (2018). Detecting patterns in the Ethereum transactional data using unsupervised learning. Master's thesis. University of Dublin, Trinity College.
- Porru, S., Pinna, A., Marchesi, M., & Tonelli, R. (2017). Blockchain-oriented software engineering: Challenges and new directions. In *2017 IEEE/ACM 39th international conference on software engineering companion (ICSE-C)* (pp. 169–171).
- Quinlan, J. R. (1996). Bagging, boosting, and c4.5. In *Proceedings of the thirteenth national conference on artificial intelligence - volume 1 AAAI96* (pp. 725–730). AAAI Press.
- Rouhani, S., & Deters, R. (2017). Performance analysis of Ethereum transactions in private blockchain. In *2017 8th IEEE International conference on software engineering and service science (ICSESS)* (pp. 70–74).
- Rudlang, M. (2017). Comparative analysis of bitcoin and Ethereum. Master's thesis. NTNU.
- Shi, X., Wong, Y. D., Li, M. Z.-F., Palanisamy, C., & Chai, C. (2019). A feature learning approach based on xgboost for driving assessment and risk prediction. *Accident Analysis & Prevention*, 129, 170–179.
- Sokolowska, A. (2018). How to interact with the Ethereum blockchain and create a database with python and sql.
- Sturm, C., Schnig, S., Szalanczi, J., & Jablonski, S. (2019). A blockchain-based and resource-aware process execution engine. *Future Generation Computer Systems*, 100, 19–34.
- Torlay, L., Perrone-Bertolotti, M., Thomas, E., & Baci, M. (2017). Machine learning-xgboost analysis of language networks to classify patients with epilepsy. *Brain Informatics*, 4(3), 159.
- Vasek, M., & Moore, T. (2019). Analyzing the bitcoin Ponzi scheme ecosystem. In A. Zohar, I. Eyal, V. Teague, J. Clark, A. Bracciali, F. Pintore, & M. Sala (Eds.), *Financial cryptography and data security* (pp. 101–112). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Zhang, D., Qian, L., Mao, B., Huang, C., Huang, B., & Si, Y. (2018). A data-driven design for fault detection of wind turbines using random forests and xgboost. *IEEE Access*, 6, 21020–21031.