Detecting Financial Bots on the Ethereum Blockchain

Thomas Niedermayer
thomas.niedermayer@ikna.io
Iknaio Cryptoasset Analytics
Vienna, Austria

Pietro Saggese
pietro.saggese@imtlucca.it
IMT School for Advanced Studies
Lucca, Italy

Bernhard Haslhofer
haslhofer@csh.ac.at
Complexity Science Hub
Vienna, Austria

ABSTRACT

The integration of bots in Distributed Ledger Technologies (DLTs) fosters efficiency and automation. However, their use is also associated with predatory trading and market manipulation, and can pose threats to system integrity. It is therefore essential to understand the extent of bot deployment in DLTs; despite this, current detection systems are predominantly rule-based and lack flexibility. In this study, we present a novel approach that utilizes machine learning for the detection of financial bots on the Ethereum platform. First, we systematize existing scientific literature and collect anecdotal evidence to establish a taxonomy for financial bots, comprising 7 categories and 24 subcategories. Next, we create a ground-truth dataset consisting of 133 human and 137 bot addresses. Third, we employ both unsupervised and supervised machine learning algorithms to detect bots deployed on Ethereum. The highest-performing clustering algorithm is a Gaussian Mixture Model with an average cluster purity of 82.6%, while the highest-performing model for binary classification is a Random Forest with an accuracy of 83%. Our machine learning-based detection mechanism contributes to understanding the Ethereum ecosystem dynamics by providing additional insights into the current bot landscape.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **General and reference** → **Empirical studies**.

KEYWORDS

Bots, Ethereum, Blockchain, DeFi, Machine Learning

ACM Reference Format:
Thomas Niedermayer, Pietro Saggese, and Bernhard Haslhofer. 2024. Detecting Financial Bots on the Ethereum Blockchain. In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3589335.3651959

1 INTRODUCTION

The algorithmic automation of financial activity is among the most fundamental innovations harnessed by Decentralized Finance (DeFi), and it is integral to achieving efficiency within this framework [3]. In Distributed Ledger Technologies (DLTs) like Ethereum, bots facilitate the automated execution of software programs, known as smart contracts, that encapsulate the logic of deterministic functions such as conventional financial operations. Bots typically manage externally owned accounts (EOAs), often referred to as wallets, which are identified by a hexadecimal address. As there is significant incentive to automate EOAs in the form of financial gain, time saving, or increased fault tolerance, bots have emerged as impactful agents on DLTs, and their relevance has become more prominent over time. For instance, bots have been increasingly exploited to extract more than $1 billion profits from the Ethereum ecosystem to date [2, 16, 28].

The deployment of bots can be beneficial to DLT-based ecosystems. Bots can be used to provide critical infrastructure, e.g., enabling interfaces between centralized exchanges and the blockchain by managing assets automatically [32]. Furthermore, layer 2 scaling solutions like roll-ups automatically deploy transaction information on Ethereum for persistence [37]. Bots also contribute to market stability through arbitrage [41].

However, bots are also exploited for predatory trading and market manipulation [28], posing financial threats to unwary users [42] and potential systemic threats to the network's integrity [5]. In cases of profit-optimizing bots, side-effects on the ecosystem can be the exploitation of users interacting with smart contracts [38], exploitation of smart contracts [46], inflated gas prices through increased traffic, and the destabilization of the network [13].

Whilst understanding the scale of bot adoption in DLTs is essential for assessing risks to human users, ensuring their protection, and informing policy, it is hard to estimate the extent of this phenomenon, as to date there are no dedicated bot detection systems. Existing software such as MEV-inspect[1] and Eigenphi[2], that focus on profit-optimizing transactions often carried out by bots, may be repurposed for bot detection. However, they are rule-based and are not built for capturing evolving bot behaviors. It is therefore important to devise bot detection methods that are more flexible and operate in a data-driven manner.

This study provides a machine learning-based method to detect financial bots deployed on Ethereum and aims to identify key bot predictors, in an effort to remove the prevalent reliance on mechanism-specific and rule-based systems. Figure 1 provides an overview of the pipeline of our work. First, we complement existing literature with anecdotal evidence to propose a taxonomy for financial bots on Ethereum. Second, we process publicly available Ethereum blockchain data and additional information from the websites Etherface.io and Etherscan.io, and we create a ground-truth dataset by annotating 270 EOAs based on the assessment of three independent annotators. We then extract features to accurately represent EOA behavior for testing bot detection using machine

WWW '24 Companion, May 13–17, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0172-6/24/05
https://doi.org/10.1145/3589335.3651959

[1] https://github.com/flashbots/mev-inspect-py
[2] https://eigenphi.io/

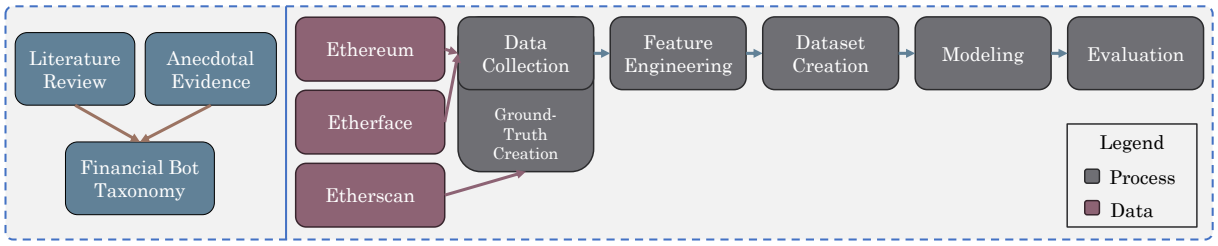

Figure 1: Overview of the methodology.

learning algorithms. Finally, we evaluate the effectiveness of these models. We utilize both unsupervised and supervised techniques to respectively group and classify EOAs as either "Bot" or "Human". Additionally, we focus on a specific subset of asset-accumulating bots, and measure algorithm performance in a multiclass classification context. The research questions and related contributions of this work can be summarized as follows:

- RQ1 **What types of bots are active on distributed ledger technologies such as Ethereum?** We propose a taxonomy for financial bots on Ethereum comprising 7 bot categories and 24 subcategories. Furthermore, we create a ground-truth dataset of 133 human and 137 bot addresses.
- RQ2 **To what extent can we detect bots automatically?** We deploy both supervised and unsupervised machine learning models and measure their performance. The highest-performing clustering algorithm is a Gaussian Mixture Model with an average cluster purity of 82.6%, evaluated on a balanced two-class dataset. The highest-performing binary classification model is a Random Forest with an accuracy of 83% on the same dataset.
- RQ3 **Which features are most informative to a model with high performance?** Using explainable AI techniques, we find that features based on the time, frequency, gas price, and gas limit of outgoing transactions are the most influential.

We believe that our work helps better understand the Ethereum ecosystem dynamics by shedding more light on the existing bot landscape and by proposing a novel ML-based detection mechanism. As these bots can significantly influence market dynamics, liquidity, and the overall network safety, it is important to monitor their prevalence and impact.

All data and source code used in this study are publicly available for reproducibility purposes[3].

2 BACKGROUND

This section provides a background of the technologies used by financial bots, defines what a bot is in a blockchain context, and summarizes the existing literature on bot detection.

2.1 Decentralized Finance (DeFi)

Ethereum's smart contract support allows the execution of programs in a decentralized manner. This gives rise to complex functionalities involving cryptoassets, i.e., digital assets that represent some economic resource or value to someone and are based on cryptographic primitives and a DLT [3]. Decentralized Finance (DeFi) enables trustless borrowing, lending, or investing on blockchain systems like Ethereum [43]. The following are core components of DeFi relevant to this work.

Tokens[4] are cryptoassets created through smart contracts that adhere to specific standards to implement core functionalities such as the transferability between accounts [39]. Tokens following the ERC-20 standard are fungible, i.e. each unit is identical to any other deployed by the same contract. Smart contracts implemented according to the ERC-721 standard are Non-Fungible Tokens (NFTs). This means they are not interchangeable, which makes each NFT unique, allowing them to be used as proof of ownership of a digital asset. Tokens have been used to represent Dollar equivalents on a blockchain (stablecoins) and NFTs have been used to represent artworks, tickets, or characters in video games [40].

Decentralized exchanges (DEXes) on Ethereum are smart contracts that create a market to exchange cryptoassets against each other. Their off-chain counterparts, i.e. centralized exchanges (CEX), are commonly used as an interface between fiat currencies and cryptoassets and often enable trading through an order book. DEXes on the other hand often employ Automated Market Makers (AMMs), that allow for trading without an order book [25]. For these AMMs to work, prevalent mechanisms require liquidity to enable trading and users are financially incentivized to provide it [20].

A relevant concept in DeFi is Maximal Extractable Value (MEV), initially termed miner-extractable value in the seminal work of Daian et al. [13]. It refers to profits that can be extracted from block production in excess of the standard block reward and transaction fees. This is achieved by including, excluding, or re-ordering transactions within a block [36]. The most prominent examples of MEV are arbitrage, sandwich attacks, and liquidations. Arbitrage exploits price differences on DEXes to generate profit. Sandwich attacks send a transaction to drive up the price, let the victim buy at the high price, to then send another transaction to sell what was bought earlier at a profit. Liquidations involve the sale of collateralized assets to repay a debt at a favorable rate for the buyer [27].

2.2 Bots

The potential for automation in Ethereum's ecosystem and the amount of assets involved in DeFi attract a diverse set of actors categorized as bots. In the context of account-based blockchain systems such as Ethereum, Zwang et al. [48] defined bots as software

[3] https://github.com/Tommel71/Ethereum-Bot-Detection

[4] https://ethereum.org/en/developers/docs/standards/tokens

robots. In this work, we describe them more precisely by defining the two mutually exclusive classes *human* and *bot* and establish the following Definition 2.1:

Definition 2.1 (Bot). A bot is an EOA that has sent at least one transaction that was compiled and sent by a software program without human oversight, i.e. a bot transaction.

2.3 Bot Detection and Related Literature

Software programs such as MEV-inspect and Eigenphi can discern transactions related to MEV. As these transactions are highly time-critical, bots are employed to execute them; we can therefore interpret these programs as bot detection systems, by recognizing the sender of detected transactions as bots. Currently, these systems primarily focus on transaction patterns in a rule-based manner and are focused on a specific set of transactions modeled in their code. In addition to such programs, research related to the identification of bots on Ethereum has been conducted. Studies leverage heuristic methods, which allow to analyze bots without expensive annotation efforts. Graph-based heuristics have shed light on bot types and the prevalence of private transaction use in MEV exploitation [26]. Time-based heuristics have also been instrumental in identifying automated wallet behaviors, revealing patterns in transaction timings [48]. Further, a categorization of Ethereum wallets using transaction-based and graph-theoretic metrics has been carried out [7]. An analysis of bots involved in sniping, a subset of MEV that aims to acquire cryptoassets as fast as possible to profit from other users driving up the price, highlights their operational tactics and prevalence through heuristic examination of transaction timings [9]. Overall, these heuristic-based studies provide a solid basis for bot detection and motivate features that could be implemented in a more flexible ML-based system.

MEV bots have received particular attention because of MEV's impact on Ethereum security and economy, as highlighted by recent research [16, 28]. Because time is critical to exploit MEV opportunities, bots are employed to automatically find them and assemble and send transactions that act upon them. Key studies have shown the practical threats posed by MEV, focusing on real-time pending transactions data analysis [13]. There is a systematization of knowledge around front-running on blockchains, drawing parallels with traditional finance and classifying front-running based on adversarial intent [15].

Whilst we are not aware of any work that uses ML for bot detection specifically, several works used ML approaches for the broader task of behavior-based wallet classification. Unsupervised learning, specifically using expectation-maximization algorithms coupled with Random Forest models, has been applied for clustering Ethereum wallets and detecting anomalies [4]. Key statistical measures have been calculated over data grouped by Ethereum addresses and effectively used for machine learning-based classification of accounts [4, 17]. Supervised and unsupervised techniques have been combined to classify attack instances, utilizing deep neural networks for attack detection [29]. Ponzi scheme detection in Ethereum, using a combination of manual labeling and feature extraction, exemplifies the application of machine learning in identifying fraudulent activities [11, 17]. Additionally, anomaly detection in Bitcoin transactions, employing methods like one-class support vector machines and k-means clustering have been surveyed [33].

In conclusion, we found software applicable for bot detection, research employing rule-based heuristics to detect bots, and more generally, research in the use of ML to detect certain behaviors of accounts. To the best of our knowledge, however, there is no research on ML-based bot detection on Ethereum. This serves as a rationale to motivate our study.

3 BOT CATEGORIES

In an effort to provide a comprehensive representation of the DeFi bot landscape in Ethereum, we complement the existing scientific literature with anecdotal evidence of existing financial bots that we collected from GitHub repositories, sample transactions, or sample accounts (see Table 5 in Appendix for further details). We report our taxonomy in Figure 2, divided into 7 categories and 24 subcategories, based on the collected evidence. Below we outline the characteristics of the seven categories identified.

MEV bots exploit a blockchain state for profit by rapidly detecting and executing profitable opportunities, raising concerns about fairness and protocol integrity. *Centralized Exchange (CEX) bots* handle interactions between centralized exchanges and blockchains. They

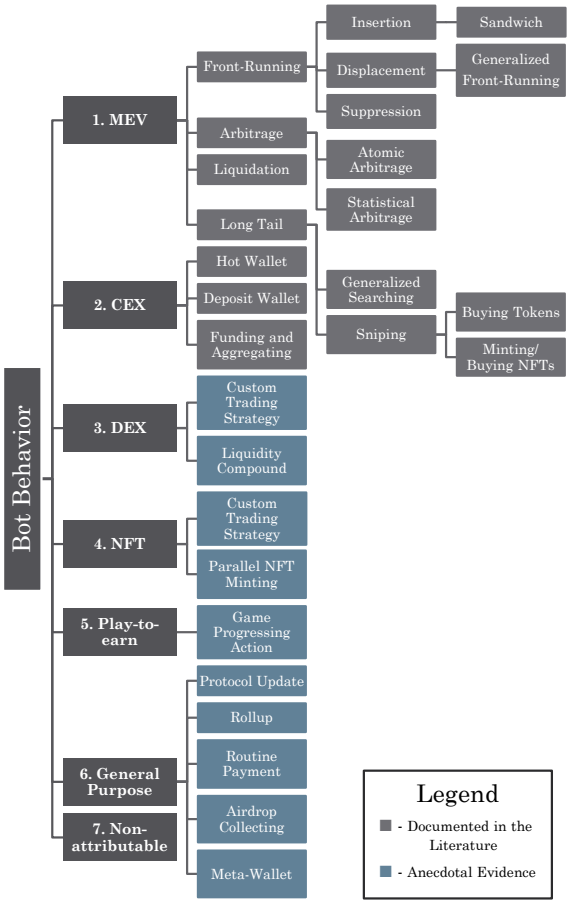

Figure 2: Taxonomy of Bots on Ethereum.

include hot wallets for daily operations, deposit wallets that manage user funds, and funding wallets that provide gas-fee-covering liquidity. *Decentralized Exchange (DEX) bots* engage in custom trading strategies and liquidity compounding. They use automated mechanisms to trade tokens or provide liquidity to pools, often for staking rewards. *Non-Fungible Token (NFT) bots* engage in trading strategies and minting practices. This includes automated trading based on pricing models and parallel minting to circumvent bulk limits, imposed by smart contracts to prevent exploitation. *Play-to-Earn (P2E) bots* automate game-progressing actions in blockchain-based games for rewards. Common actions include questing, combat, crafting, and market interactions. Automated processes enable constant game progression and asset accumulation. *General Purpose bots* execute routine tasks like protocol updates, rollups, payments, and airdrop collecting. They automate repetitive or resource-intensive tasks. Finally, *Non-Attributable bots* exhibit automated behavior without a clear purpose and are therefore categorized as non-attributable.

4 DATA AND METHODS

In the previous sections, we have defined bots and given an overview of the bot landscape. We now describe how we process publicly available blockchain data and smart contract information to create our datasets.

4.1 Data Sources

We gathered data from the following sources:

Ethereum blockchain: We ingest transaction data from an Erigon archive node[5] using *Graphsense-lib*[6] [18]. We investigate the block range of 15,500,000 (September 9, 2022) until 15,599,999 (September 24, 2022), i.e. 100,000 blocks. This limited range allows our method to run on consumer hardware. Furthermore, the range is defined so that it contains no significant hype-phase which might bias the analysis, and all major features of DeFi as we know it are already established, which is relevant for many bot categories. While the method described here uses 100,000 blocks, the block range can be extended or restricted arbitrarily to include more information in the features or allow for faster processing, respectively.

Etherface.io: We use smart contract function and event descriptions from Etherface.io[7] to decode smart contract functions and events and make data in swaps and token transfers accessible.

Etherscan.io: Using the API of Etherscan.io[8], we provide additional transaction data about addresses annotated to allow annotators a more comprehensive analysis.

4.2 Ground-Truth Creation

We call the last two blocks of our block range the test blocks and annotate them to yield a labeled dataset. To manually annotate all 270 EOAs sending transactions in the test blocks, we employ two independent annotators A and B. Based on information documented on the web, indexing services, and custom aggregate metrics and graphs, both annotators attach to all EOAs one of the labels "Bot" and "Human". In addition to the binary labels, annotator A adds fine-grained labels, indicating the specific use of a bot[9]. In the case of a disagreement, a third annotator C breaks the tie based on a description noted down by annotators A and B. In the case of a tie-breaker where A picks Human (and therefore provides no fine-grained bot label) and C chooses B, then B provides the fine-grained label. The result is a dataset of 137 bot and 133 human addresses, where 31 addresses required a tie-breaker. The Cohen's Kappa of the first two annotators is 0.77, which indicates substantial agreement. Finally, to enable the analysis of a classifier discerning multiple bot classes, we gather additional data about short-tail MEV bots. We run MEV-inspect on the entire block range to obtain 111 bots for each category *Arbitrage*, *Sandwich*, and *Liquidation* bots. The number of bots is the same for all categories by design and is limited by the lowest number of bots found in the category *Liquidation*.

4.3 Feature Engineering

We perform feature engineering by transforming the transaction history of an address into aggregate metrics. This is mostly done for outgoing transactions or token transfers and in some cases for incoming ones. We calculate a total of 83 features per wallet. These metrics are in some cases informed by knowledge of trading bots or human behavior while in other instances, they reflect the address' overall activity to provide a subsequent model with highly informative features. Figure 6 in the Appendix shows an overview of the features, which are divided into four main categories: address-based features (based on the 40-character-long hexadecimal representation of an address and requires no transaction history), transaction-based features (based on transaction data), function-call-based features (based on input data sent in transactions to smart contracts), and event-based features (based on data smart contracts produce). The general strategy is to first extract specific data from the transaction history of an account to subsequently apply aggregate functions that produce one or more features. We first present such functions that are used multiple times, and then describe the features, which may require specific functions directly defined with the feature.

General Functions. The function *BenfordsLaw*, applied on a given set of values, leverages Benford's law, which predicts the distribution of the first significant digits in natural datasets. The function applies Pearson's Chi-squared test to assess how well first-significant-digit distributions align with Benford's law, generating a p-value as a feature [6, 12]. To capture the tendency of human traders to use round numbers as cognitive reference points we define *TradeValueClustering* as the ratio of round and non-round numbers in the grouped data [12]. We call a number round if its digits after the 7th significant digit are zero. *GapBasedSleepiness* calculates time deltas for a list of timestamps $ts = [t_1, t_2, \ldots, t_k]$ as $\mathit{TimeDelta}(ts) = [t_2 - t_1, t_3 - t_2, \ldots, t_k - t_{k-1}]$. Timestamps are divided into two-day intervals to minimize time-zone effects and account for human sleep patterns. In each interval, the maximum time delta is determined and then averaged across intervals to produce a single value. *Categorical* takes a group of categorical values

[5] https://erigon.tech/
[6] https://github.com/graphsense/graphsense-lib
[7] Etherface.io
[8] https://etherscan.io/

[9] We did not provide the annotators with a preset list of bot categories to prevent limiting them to our initial findings. Additionally, our annotated block range is too small to cover all classes in the taxonomy. While these circumstances partly disconnect our bot taxonomy from the fine-grained labels, there is still notable overlap.

and returns the entropy of the distribution of class occurrences, for each class the percentage of its occurrence, and the mode. *Numerical* takes a group of numerical values and returns mean, mode, standard deviation, minimum, maximum, and the 95% quantile.

4.3.1 Address-based features. The feature *NLeadingZeros* counts leading zeros in Ethereum's hexadecimal addresses. Addresses with more leading zeros, often mined for efficiency, indicate deliberate creation to reduce gas fees and increase smart contract efficiency [1]. *DigitEntropy* measures the entropy of digits in an Ethereum address, capturing randomness or patterns which could be expressed differently in bots and therefore contain predictive power.

4.3.2 Transaction-based features. *TxPerBlock* calculates the number of transactions sent divided by the total number of blocks (100,000). *StatisticsTxPerActiveBlock* calculates the number of transactions sent for each block, removes zeros, and applies *Numerical*. The resulting metrics inform about the intensity of activity and its distribution in blocks where transcations were sent.

For time-based features, *GapBasedSleepiness* is applied to the timestamps of in- and outgoing transactions separately. *TransactionFrequency* calculates for both in- and outgoing transactions separately the total number of transactions divided by the number of seconds between first and last transaction made. *StatisticsTimeDifference* represents the features returned by the *Numerical* function applied to the group of time differences between consecutive transactions sent. *EntropyTime* is done for both in- and outgoing transactions separately and calculates the entropy (see Appendix A) of the timestamps after transforming them to represent the hours $h \in [1, 2, \ldots, 23, 24]$ of the day they were sent in [21]. For value-based features, *BenfordsLaw* and *TradeValueClustering* are calculated for the ETH values transferred in outgoing transactions. *Standard* fields are the categorical "type" and "status", and the numerical "value", "gasLimit", and "gasPrice", together with the index of a transaction[10] normalized by the number of transactions in the block named "indexRelative". For each data group of a standard field, *Categorical* or *Numerical* is applied according to the data type.

4.3.3 Function-Call-based features. To filter blockchain data for specific smart contract function calls or emitted events related to DEXs, we modeled them based on the Uniswap protocol because it is pioneering the field of DEXs with an open-source code base. This has led to many other protocols copying Uniswap's code and interfaces allowing us to capture their functionality by modeling functions and events of Uniswap only.

We filter the outgoing transaction history for swaps where we modeled the contract. Swaps are especially relevant for financial bots as they represent a basic functionality many of them use and can therefore help in creating better representations of account behavior in the form of features. We calculate *BenfordsLaw* and *TradeValueClustering* for the "amountIn" or "amountOut" field denominated in the swap because in the case of a human using the website of a DEX, these fields are provided by the user while other fields in the smart contract call are often automatically calculated by the website before the transaction is sent. Automatically calculated values are less suited for metrics based on cognitive rules like *TradeValueClustering*. *StatisticsPathLength* takes for all outgoing transactions the path parameters which determines if the input asset is swapped directly into the output asset or if there are other swaps between, takes their length (two for a direct swap), and applies the *Numerical* function. Finally, *SwapsPerBlock* calculates the number of swap transactions divided by the total number of blocks.

4.3.4 Event-based features. For swaps based on UniswapV2, we get the input amount by adding up the fields "amount0In" and "amount1In" and for UniswapV3, we get the input amount by taking the maximum of the fields "amount0" and "amount1". We group the data by the recipient of the swap and not the sender in the event data because the senders are usually smart contracts. We calculate *BenfordsLaw* and *TradeValueClustering* for the input amount denominated in the swaps. Additionally, in *SwapsPerBlock*, the number of swap events divided by the total number of blocks is given. Similarly, *BenfordsLaw* and *TradeValueClustering* are calculated for the value of transfer events with the standard ERC-20 transfer event signature. Finally, *TransferEventsPerBlock* gives the number of transfer events divided by the total number of blocks.

For features based on swaps, we carry out a dimensionality reduction step that averages the respective features over all the functions and events modeled to reduce 128 features to 14. StatisticsPathLength features are only averaged over the functions modeled.

4.4 Dataset Creation

Lastly, we use the 100,000 blocks described in Section 4.1 to create three datasets of different sizes with each row representing one address as a set of features.

(1) **Binary Bot Dataset**: We collect all 270 addresses sending in the test blocks to produce this dataset. As described in Section 4.2, these addresses are associated with the binary labels "Bot" and "Human".
(2) **Clustering Dataset**: We collect all addresses sending transactions in the 100,000 blocks window and exclude the ones sending in the test blocks. This dataset is unlabeled.
(3) **Multiclass MEV Dataset**: We randomly sample 111 addresses from the test blocks that are not involved with MEV and combine them with 111 addresses of each short-tail MEV category from the automatically annotated addresses. This results in a labeled dataset with 4 classes, each of size 111 with the corresponding labels "Arbitrage", "Sandwich", "Liquidation", and "non-MEV".

5 RESULTS

In this section, we analyze the effectiveness of clustering and classification to detect bots on Ethereum. Furthermore, we investigate which features are most influential in the decision of our best performing classifier.

5.1 Clustering

We begin by using clustering techniques to detect bots on Ethereum and explore how well bots and humans can be grouped in an unsupervised environment. To assess the quality of a clustering, we measure entropy and purity (see Appendix A) with respect to the two classes Bot and Human in all individual clusters. Purity is a measure of homogeneity and a higher value means the cluster is

[10] All these values are found in Erigon transaction and receipt data.

more dominated by data points from a single class. Entropy is a measure of disorder in a cluster and is one for equal representation of all classes and zero for a cluster containing only one class. Therefore, for this analysis, we first utilize the unlabeled clustering dataset for fitting and next employ the binary bot dataset to compute purity and entropy.

After surveying multiple clustering methods, k-means and Gaussian Mixture Model (GMM) were chosen because they are inductive and can therefore be used to cluster new instances in the binary bot dataset, and because they run acceptably fast. For both algorithms we use the scikit-learn implementation, whereby the number of clusters has to be provided. We tried different approaches to find the optimal number of clusters, such as minimization and the elbow method [23] on the Bayesian Information Criterion (BIC) [34], and the silhouette score [31]. While for k-means, minimizing the BIC yields more than 100 clusters and using the elbow method on BIC or silhouette scores yields 5 to 10 clusters, minimizing BIC for GMM yields 10 to 30 clusters. This wide range of cluster sizes suggested by common methods stands in conflict with the evaluation metrics we use. Since we have annotated data, we can calculate purity and entropy regarding the distribution of Bot and Human addresses within a cluster. These are appealing metrics because they tell whether or not a certain cluster can be used for selecting a higher percentage of either Bots or Humans. The conflict arises since a higher number of clusters, all else being equal, leads to higher purity because more clusters allow for fewer addresses per cluster on average, which in turn leads to a higher chance that the distribution in a cluster is pure just by chance. This effect becomes evident when a cluster may fit around one single address which leads to a purity of 100%. For this reason, we evaluate the algorithms at three fixed cluster sizes of 5, 15, and 30, to obtain results at different levels of model complexity. Additional results based on BIC optimization and the 'elbow' method are reported in the Appendix, Table 6.

For both GMM and k-means, we preprocess the features in the clustering dataset by Min-Max-scaling to the $[0, 1]$ range, ensuring equal importance across features. We also examine whether UMAP embedding [24] to two dimensions enhances performance and if imputing missing values with the column mean or -1 yields better outcomes. Imputing missing values with -1 post-scaling might be advantageous, as it distinctly separates addresses with missing values from those without in the feature space.

Finally, to calculate purity and entropy, all combinations of preprocessing methods, clustering methods, and number of clusters are fitted on the clustering dataset and tested by assigning clusters to the addresses in the binary bot dataset. For an entire clustering, we define the two metrics as the weighted average of the metrics of the individual clusters, weighted by the ratio of the samples in the respective cluster.

Table 1 shows the results for different algorithms, preprocessing methods, and cluster sizes. GMM with 30 clusters, mean-imputation and no dimensionality reduction scores best in entropy and purity; we name it the top clustering algorithm. In general, UMAP-embedding is beneficial for a small number of clusters and detrimental in runs with large numbers of clusters. For smaller models, UMAP embedding even resulted in the best-performing clusterings. This could be due to embedding losing information that could be leveraged by larger models while helping to separate clusters effectively only using a few clusters by simplifying the data.

Table 1: Average entropy and purity of the combinations of preprocessing methods and clustering algorithms explored with different cluster sizes.

Algorithm	Imputation	Dimension. Reduction	Purity 5 Clusters	Purity 15 Clusters	Purity 30 Clusters	Entropy 5 Clusters	Entropy 15 Clusters	Entropy 30 Clusters
GMM	-1	UMAP	**0.726**	0.744	0.756	0.842	0.795	**0.749**
		non-UMAP	0.581	0.670	0.789	0.934	0.791	0.606
	mean	UMAP	0.544	0.741	0.770	**0.972**	0.774	0.724
		non-UMAP	0.552	0.770	**0.826**	0.954	0.725	0.563
kmeans	-1	UMAP	0.704	0.737	0.770	0.868	0.796	0.738
		non-UMAP	0.633	0.674	0.711	0.906	**0.801**	0.742
	mean	UMAP	0.652	0.748	0.778	0.923	0.777	0.700
		non-UMAP	0.585	**0.774**	0.800	0.970	0.713	0.649

Table 2: Individual-cluster-level metrics of the largest clusters of the top clustering algorithm.

Cluster	Purity	Entropy	Size	Majority
2	0.667	0.918	18	Human
5	0.927	0.376	55	CEX
6	0.842	0.629	19	Human
13	0.929	0.371	28	Deposit Wallet
14	0.579	0.982	19	Human
19	0.872	0.552	39	Human

Table 2 reports the 6 clusters produced by the top clustering algorithm that contain at least 15 addresses. The majority class within a cluster is based on the fine-grained label of the binary bot dataset. Clusters 6 and 19 have the majority class "Human" and show high purity. Clusters 5 and 13 also show exceptionally high purity with the majority class CEX and Deposit Wallet respectively. Clusters 2 and 14 have low quality, identifiable by an entropy close to 100%. Conclusively, there are easily discerned fine-grained bot classes such as Deposit Wallet and CEX, and with 30 clusters, a purity of 82.6% can be attained.

5.2 Classification

Following is an analysis of the effectiveness of different supervised ML algorithms in discerning bots from humans and other types of bots. To this end, we evaluate binary classification algorithms on the binary bot dataset and multiclass classification algorithms on the multiclass MEV dataset. Note that the latter was partially generated by MEV-inspect and therefore does not generalize readily beyond the system's capabilities in a supervised setting. However, it allows us to further test our method and the utility of the features. To effectively use all labeled data, our approach deviates from the typical data science workflow. Instead of iteratively refining a model on a training subset before evaluating on a holdout set, we select methods that generally work well on small tabular datasets and evaluate them only once on the entire labeled data. This allows us to use more data to present our results but limits optimization.

For our binary and multiclass classifiers, we use the scikit-learn implementations of Random Forest and AdaBoost, and XGBoost

from the package XGBoost [10]. We set the number of estimators for the Random Forest at 400 and leave the rest of the parameters at their default values. We chose these methods because they are known to work well for small tabular datasets [45]. Even though tree-based methods are not sensitive to scale, we apply standardization and mean-imputation in a preprocessing step.

To measure the performance of classifiers, we employ the standard binary classification metrics accuracy, recall, precision, and F1 where Bot represents the positive class. For multiclass problems, we use the macro average of the respective metric. We model the binary bot dataset with the binary version of the chosen supervised methods and the multiclass MEV dataset with their multiclass equivalent. To increase the utilization of the binary bot and multiclass MEV datasets, we use 20-fold cross validation and calculate an average of the corresponding metric, and further report a confidence interval for the mean based on the 20 values.

Table 3: Evaluation metrics of binary classifiers on the binary bot dataset.

Algorithm	Accuracy	Precision	Recall	F1
Random Forest	**0.83** (0.77, 0.88)	**0.87** (0.77, 0.97)	0.77 (0.68, 0.86)	0.80 (0.72, 0.88)
GradientBoosting	0.82 (0.77, 0.86)	0.85 (0.77, 0.92)	0.78 (0.70, 0.87)	0.80 (0.73, 0.87)
AdaBoost	**0.83** (0.78, 0.87)	0.84 (0.75, 0.94)	**0.80** (0.71, 0.88)	**0.81** (0.73, 0.88)

Testing the utility of the three classifiers Random Forest, Gradient Boosting, and AdaBoost on the binary Bot / Human problem on the binary bot dataset, we show in Table 3 that the algorithms perform similarly to each other, especially considering the difference in mean accuracy and the size of the confidence intervals.

Table 4: Evaluation metrics of multiclass classifiers on the multiclass MEV dataset.

Algorithm	Accuracy	Precision	Recall	F1
Random Forest	**0.77** (0.73, 0.81)	**0.77** (0.72, 0.82)	**0.77** (0.72, 0.81)	**0.75** (0.70, 0.80)
GradientBoosting	0.76 (0.72, 0.81)	0.76 (0.71, 0.82)	0.76 (0.70, 0.81)	0.74 (0.68, 0.80)
AdaBoost	0.50 (0.44, 0.56)	0.54 (0.47, 0.61)	0.50 (0.44, 0.56)	0.49 (0.43, 0.54)

In Table 4, the performance of the surveyed methods on the four-class classification problem Arbitrage / Sandwich / Liquidation / non-MEV is displayed. Random Forest performed best and we call it the top classifier. More in-depth, for the top classifier, liquidations are the easiest to discern with an accuracy of 93%: features capturing gas usage allow an easy distinction as liquidation addresses show unusually high values in them (as illustrated in Figure 5 in the Appendix). Notably, sandwich addresses are difficult to detect with 16% of them misclassified as Arbitrage and 13% as non-MEV and a low accuracy of 68%. Furthermore, 17% of non-MEV is mistakenly classified as a sandwich bot. Again, Random Forest performs similar to Gradient Boosting, but AdaBoost performs significantly worse than in the binary Bot / Human problem. In conclusion, our features are not only suitable to distinguish general bots from humans, but also to differentiate between more specific classes of bots. While the generation of the multiclass MEV dataset does not allow for generalization beyond the capabilities of MEV-inspect, we gain the insight that our features are descriptive of various kinds of EOAs.

5.3 Feature Investigation

To find the most important features of the top classifier in the binary and multiclass setting and measure the impact of the most important features, we carry out further analysis based on explainable AI using the binary bot dataset and the multiclass MEV dataset.

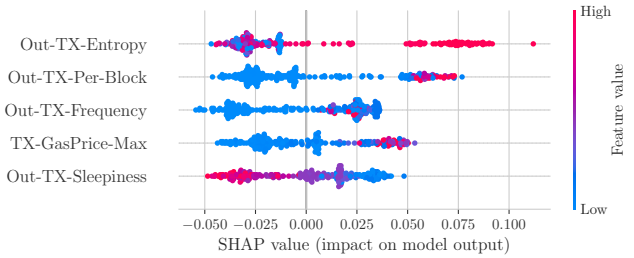

Figure 3: For each of the 5 most influential features of the top classifier a row displays 100 dots symbolizing randomly sampled addresses from the binary bot dataset. The x-axis shows each address's SHAP value, and the dot color indicates the feature value.

5.3.1 SHAP Values. SHAP (SHapley Additive exPlanations) values [22] are a method used in machine learning and data science to explain the output of a machine learning model's predictions for individual data points. They provide a way to attribute the contribution of each feature to the final prediction for a specific instance by keeping all other features the same and altering the specific feature investigated. In Figure 3, SHAP values are plotted for a sample of the binary bot dataset to further investigate the top classifier to display the most influential features. Following is a description of the features in the Figure.

(1) **Out-TX-Entropy**: The entropy of the distribution of times of outgoing transactions. Before calculating the entropy, the times are counted based on the hour of the day thereby yielding a discrete distribution. Higher values tend to shift the model's prediction towards the bot class.
(2) **Out-TX-Per-Block**: The average number of outgoing transactions per block. This feature signals high activity of an address and as high SHAP value addresses have high values of this feature, it seems to raise the suspicion of the model.
(3) **Out-TX-Frequency**: The average number of outgoing transactions per block in the timeframe the EOA was active. High SHAP value addresses contain both very high and very low feature value addresses. Low SHAP value addresses always have a low frequency.
(4) **TX-GasPrice-Max**: The maximum of the gas price of outgoing transactions. The higher the highest gas price paid, the more the model tends to classify an address as a bot.
(5) **Out-TX-Sleepiness**: The GapBasedSleepiness for outgoing transactions. The more extreme the value, the greater its impact on the model's prediction. A smooth transition in color in its representation indicates that it can, by itself, shift the model's propensity towards a certain class. Compared to the other features, Out-TX-Sleepiness has low feature values

for high SHAP value EOAs. This means that a low number in Out-TX-Sleepiness shifts the model's prediction to "Bot".

Figure 4 displays means of absolute SHAP values of the top classifier for all four classes in the multiclass MEV dataset. As SHAP values are defined for each class separately, they are displayed in a more aggregated view. The five most influential features are all statistics of the "gas limit" parameter of a transaction which defines the maximum amount of gas that may be used. Especially liquidation bots are heavily impacted by these metrics indicating they are willing to pay higher gas fees for their transactions. Gas limit metrics are least effective for identifying sandwich bots. For them, the standard deviation in Ether transferred is most significant.

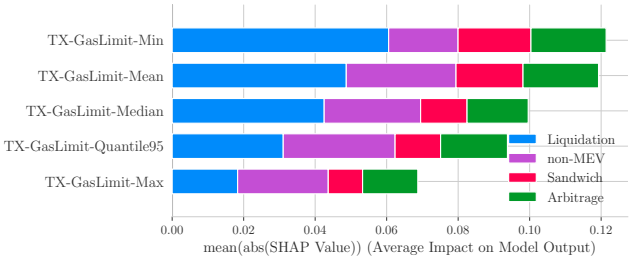

Figure 4: Stacked bar plot of the mean absolute SHAP values of all addresses in the multiclass MEV dataset with respect to the top classifier.

6 DISCUSSION AND CONCLUSIONS

Our study sheds light on the role of financial bots in the Ethereum ecosystem. First, we identified and categorized the types of bots that are active on Ethereum, and we published a ground-truth dataset of 133 human and 137 bot addresses. Second, by deploying both supervised and unsupervised machine learning models, we devised an approach to detect bots automatically. Our best-performing clustering and classification models respectively reach above 82.6% purity and 83% accuracy. Third, we used explainable AI to investigate which features are most informative to our models, finding that the most influential ones relate to time, frequency, gas price, and gas limit of outgoing transactions.

We note that clustering using a Gaussian Mixture Model proves to be effective in grouping certain subtypes of bots, and humans with high purity. This means that after an initial clustering step, clusters can be mapped to certain categories to single out categories of interest. Random Forest and Gradient Boosting work well both on a binary Bot/Human task and a four-class task classifying three types of MEV and non-MEV. Analyzing the features used in the model yielding the best results, we found that bots show different activity/inactivity patterns compared to humans, as captured by the *GapBasedSleepiness* feature, first proposed in this work.

Further, our findings have some relevant implications. First, our taxonomy establishes a state-of-the-art description of the existing financial bots, and therefore provides support to discern bots that can provide critical infrastructure to the DLT ecosystem from those that can be exploited for predatory trading and market manipulation.

Second, even though we acknowledge that the sample we consider is small, our study provides preliminary insights into the extent of the bot deployment in the Ethereum ecosystem. Indeed, we found that more than 50% of the labeled addresses are bots. Our work can provide additional insights into the magnitude of this phenomenon. By devising an approach to detect bots automatically, we are also able to predict with a good level of accuracy how many bots execute transactions in future incoming blocks. Similarly, in principle, our approach can be scaled to include previous transactions, allowing to provide an estimation of the magnitude of this phenomenon on the entire transaction history of Ethereum. Third, explainable AI tools such as SHAP values can contribute to informing human users and policymakers on potential risks and characteristics of users that are likely to be bots.

Besides this, we acknowledge that our work faces some limitations and opens directions for further research. One conceptual issue is that it is hard to define exactly what a bot is, and what are the requirements to precisely define a bot based on its activity. The major limit to our study, however, is the small size of our labeled dataset. Our current ground-truth dataset is composed of just 270 EOAs that have been labeled by three independent annotators. A straightforward improvement to this study would be to extend the number of test blocks and annotators to augment the labeled dataset and increase the confidence level of the assigned labels. An effective strategy might be the adoption of semi-supervised learning methods [47] to combine our small labeled data with readily available unlabeled data. As a restriction to a small set of blocks leads to the overrepresentation of highly active addresses, expanding the number of blocks annotated would also lead to a more representative distribution. In future work, additional data may allow for a standard data science workflow such as CRISP-DM [44], and enable optimization of model choice and parameters. Furthermore, we currently analyze only 100,000 blocks. However, our approach is easily scalable to larger time windows and the major purpose of this paper is to provide the methodological approach to identify bots. Another limitation is that this study focuses on bots that engage in financial activity. However, bots are likely utilized also in other domains, for instance, to automate voting processes in Decentralized Autonomous Organizations.

Future studies could extend our taxonomy also to non-financial bots. Further, we currently conduct multiclass classification on a restricted class of bots, i.e., MEV bots. Future work could also extend this analysis to a larger number of bot categories. Finally, future studies could investigate the phenomenon on other blockchains, and verify whether the bot categories differ across DLTs.

7 ACKNOWLEDGEMENTS

This work is partially funded by the Austrian security research program KIRAS of the Federal Ministry of Finance (BMF) under the project DeFiTrace (grant agreement 905300), the FFG BRIDGE project AMALFI (grant agreement 898883), and the COMET Centre ABC (Austrian Blockchain Center) managed by the FFG (grant agreement 909237).

REFERENCES

[1] 0age. 2019. On Efficient Ethereum Addresses. https://medium.com/coinmonks/on-efficient-ethereum-addresses-3fef0596e263

[2] Raphael Auer, Jon Frost, and Jose María Vidal Pastor. 2022. Miners as intermediaries: extractable value and market manipulation in crypto and DeFi. (2022).

[3] Raphael Auer, Bernhard Haslhofer, Stefan Kitzler, Pietro Saggese, and Friedhelm Victor. 2023. The technology of decentralized finance (DeFi). *Digital Finance* (Aug. 2023). https://doi.org/10.1007/s42521-023-00088-8

[4] Hyochang Baek, Junhyoung Oh, Chang Yeon Kim, and Kyungho Lee. 2019. A Model for Detecting Cryptocurrency Transactions with Discernible Purpose. In *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*. 713–717. https://doi.org/10.1109/ICUFN.2019.8806126 ISSN: 2165-8536.

[5] Mikolaj Barczentewicz. 2023. MEV on Ethereum: A Policy Analysis. https://doi.org/10.2139/ssrn.4332703

[6] Frank Benford. 1938. The Law of Anomalous Numbers. *Proceedings of the American Philosophical Society* 78, 4 (1938), 551–572. https://www.jstor.org/stable/984802 Publisher: American Philosophical Society.

[7] Gianluca Bonifazi, Enrico Corradini, Domenico Ursino, and Luca Virgili. 2022. Defining user spectra to classify Ethereum users based on their behavior. *Journal of Big Data* 9, 1 (April 2022), 37. https://doi.org/10.1186/s40537-022-00586-3

[8] Álvaro Cartea, Fayçal Drissi, and Marcello Monga. 2023. Execution and Statistical Arbitrage with Signals in Multiple Automated Market Makers. https://doi.org/10.2139/ssrn.4388104

[9] Federico Cernera, Massimo La Morgia, Alessandro Mei, Alberto Maria Mongardini, and Francesco Sassi. 2023. Ready, Aim, Snipe! Analysis of Sniper Bots and their Impact on the DeFi Ecosystem. In *Companion Proceedings of the ACM Web Conference 2023*. ACM, Austin TX USA, 1093–1102. https://doi.org/10.1145/3543873.3587612

[10] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[11] Weili Chen, Zibin Zheng, Edith C.-H. Ngai, Peilin Zheng, and Yuren Zhou. 2019. Exploiting Blockchain Data to Detect Smart Ponzi Schemes on Ethereum. *IEEE Access* 7 (2019), 37575–37586. https://doi.org/10.1109/ACCESS.2019.2905769 Conference Name: IEEE Access.

[12] Lin William Cong, Xi Li, Ke Tang, and Yang Yang. 2021. Crypto Wash Trading. https://doi.org/10.2139/ssrn.3530220 Available at: https://ssrn.com/abstract=4529817.

[13] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 2020. Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability. In *2020 IEEE Symposium on Security and Privacy (SP)*. 910–927. https://doi.org/10.1109/SP40000.2020.00040 ISSN: 2375-1207.

[14] Inderjit S. Dhillon, James Fan, and Yuqiang Guan. 2001. Efficient Clustering of Very Large Document Collections. In *Data Mining for Scientific and Engineering Applications*, Robert L. Grossman, Chandrika Kamath, Philip Kegelmeyer, Vipin Kumar, and Raju R. Namburu (Eds.). Vol. 2. Springer US, Boston, MA, 357–381. https://doi.org/10.1007/978-1-4615-1733-7_20 Series Title: Massive Computing.

[15] Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. 2020. SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain. In *Financial Cryptography and Data Security (Lecture Notes in Computer Science)*, Andrea Bracciali, Jeremy Clark, Federico Pintore, Peter B. Rønne, and Massimiliano Sala (Eds.). Springer International Publishing, Cham, 170–189. https://doi.org/10.1007/978-3-030-43725-1_13

[16] Flashbots. 2023. Flashbots Transparency Dashboard. https://transparency.flashbots.net/

[17] Letterio Galletta and Fabio Pinelli. 2023. Sharpening Ponzi Schemes Detection on Ethereum with Machine Learning. http://arxiv.org/abs/2301.04872 arXiv:2301.04872 [cs].

[18] Bernhard Haslhofer, Rainer Stütz, Matteo Romiti, and Ross King. 2021. GraphSense: A General-Purpose Cryptoasset Analytics Platform. http://arxiv.org/abs/2102.13613 arXiv:2102.13613 [cs].

[19] Georgios Konstantopoulos. 2022. Symbolic MEV Extraction. https://www.youtube.com/watch?v=VkSR9jz_C-0

[20] Alfred Lehar and Christine A. Parlour. 2021. Decentralized Exchanges. https://doi.org/10.2139/ssrn.3905316

[21] Guozhu Dong Liu, Huan (Ed.). 2018. *Feature Engineering for Machine Learning and Data Analytics*. CRC Press, Boca Raton. https://doi.org/10.1201/9781315181080

[22] Scott Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. https://doi.org/10.48550/arXiv.1705.07874

[23] Dhendra Marutho, Sunarna Hendra Handaka, Ekaprana Wijaya, and Muljono. 2018. The Determination of Cluster Number at k-Mean Using Elbow Method and Purity Evaluation on Headline News. In *2018 International Seminar on Application for Technology of Information and Communication*. 533–538. https://doi.org/10.1109/ISEMANTIC.2018.8549751

[24] Leland McInnes, John Healy, and James Melville. 2020. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. https://doi.org/10.48550/arXiv.1802.03426 arXiv:1802.03426 [cs, stat].

[25] Vijay Mohan. 2022. Automated market makers and decentralized exchanges: a DeFi primer. *Financial Innovation* 8, 1 (Feb. 2022), 20. https://doi.org/10.1186/s40854-021-00314-5

[26] Julien Piet, Jaiden Fairoze, and Nicholas Weaver. 2022. Extracting Godl [sic] from the Salt Mines: Ethereum Miners Extracting Value. https://doi.org/10.48550/arXiv.2203.15930 arXiv:2203.15930 [cs].

[27] Kaihua Qin, Liyi Zhou, Pablo Gamito, Philipp Jovanovic, and Arthur Gervais. 2021. An Empirical Study of DeFi Liquidations: Incentives, Risks, and Instabilities. In *Proceedings of the 21st ACM Internet Measurement Conference*. 336–350. https://doi.org/10.1145/3487552.3487811 arXiv:2106.06389 [cs, q-fin].

[28] Kaihua Qin, Liyi Zhou, and Arthur Gervais. 2021. Quantifying Blockchain Extractable Value: How dark is the forest? https://doi.org/10.48550/arXiv.2101.05511 arXiv:2101.05511 [cs].

[29] Elnaz Rabieinejad, Abbas Yazdinejad, and Reza M. Parizi. 2021. A Deep Learning Model for Threat Hunting in Ethereum Blockchain. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 1185–1190. https://doi.org/10.1109/TrustCom53373.2021.00160 ISSN: 2324-9013.

[30] Dan Robinson and Georgios Konstantopoulos. 2020. Ethereum is a Dark Forest. https://www.paradigm.xyz/2020/08/ethereum-is-a-dark-forest

[31] Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* 20 (Nov. 1987), 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

[32] Pietro Saggese, Esther Segalla, Michael Sigmund, Burkhard Raunig, Felix Zangerl, and Bernhard Haslhofer. 2023. Assessing the Solvency of Virtual Asset Service Providers: Are Current Standards Sufficient? https://doi.org/10.2139/ssrn.4586682

[33] Sirine SAYADI, Sonia BEN REJEB, and Ziéd CHOUKAIR. 2019. Anomaly Detection Model Over Blockchain Electronic Transactions. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. 895–900. https://doi.org/10.1109/IWCMC.2019.8766765 ISSN: 2376-6506.

[34] Gideon Schwarz. 1978. Estimating the Dimension of a Model. *The Annals of Statistics* 6, 2 (1978), 461–464. https://www.jstor.org/stable/2958889 Publisher: Institute of Mathematical Statistics.

[35] C. E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27, 3 (July 1948), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x Conference Name: The Bell System Technical Journal.

[36] Corwin Smith. 2023. Maximal extractable value (MEV). https://ethereum.org Available at: https://ethereum.org/developers/docs/mev.

[37] Louis Tremblay Thibault, Tom Sarry, and Abdelhakim Senhaji Hafid. 2022. Blockchain Scaling Using Rollups: A Comprehensive Survey. *IEEE Access* 10 (2022), 93039–93054. https://doi.org/10.1109/ACCESS.2022.3200051 Conference Name: IEEE Access.

[38] C. F. Torres, R. Camino, and R. State. 2021. Frontrunner Jones and the Raiders of the Dark Forest: An Empirical Study of Frontrunning on the Ethereum Blockchain. *ArXiv* (Feb. 2021). https://www.semanticscholar.org/paper/Frontrunner-Jones-and-the-Raiders-of-the-Dark-An-of-Torres-Camino/189c624e936060f5c106c7247ac5e87a75becdb8

[39] Fabian Vogelsteller and Vitalik Buterin. [n. d.]. ERC-20: Token Standard. https://eips.ethereum.org/EIPS/eip-20

[40] Qin Wang, Rujia Li, Qi Wang, and Shiping Chen. 2021. Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges. https://doi.org/10.48550/arXiv.2105.07447 arXiv:2105.07447 [cs].

[41] Ye Wang, Yan Chen, Haotian Wu, Liyi Zhou, Shuiguang Deng, and Roger Wattenhofer. 2022. Cyclic Arbitrage in Decentralized Exchanges. In *Companion Proceedings of the Web Conference 2022 (WWW '22)*. Association for Computing Machinery, New York, NY, USA, 12–19. https://doi.org/10.1145/3487553.3524201

[42] Ye Wang, Patrick Zuest, Yaxing Yao, Zhicong Lu, and Roger Wattenhofer. 2022. Impact and User Perception of Sandwich Attacks in the DeFi Ecosystem. In *CHI Conference on Human Factors in Computing Systems*. ACM, New Orleans LA USA, 1–15. https://doi.org/10.1145/3491102.3517585

[43] Sam Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William Knottenbelt. 2023. SoK: Decentralized Finance (DeFi). In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies (AFT '22)*. Association for Computing Machinery, New York, NY, USA, 30–46. https://doi.org/10.1145/3558535.3559780

[44] R. Wirth and Jochen Hipp. 2000. CRISP-DM: Towards a standard process model for data mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining* (Jan. 2000).

[45] Haoyin Xu, Kaleab A. Kinfu, Will LeVine, Sambit Panda, Jayanta Dey, Michael Ainsworth, Yu-Chung Peng, Madi Kusmanov, Florian Engert, Christopher M. White, Joshua T. Vogelstein, and Carey E. Priebe. 2021. When are Deep Networks really better than Decision Forests at small sample sizes, and how? http://arxiv.org/abs/2108.13637 arXiv:2108.13637 [cs, q-bio, stat].

[46] Bill Zhang and Amy Chou. 2023. chi-research/symbolic-searcher. https://github.com/chi-research/symbolic-searcher original-date: 2022-09-10T16:50:58Z.
[47] Xiaojin (Jerry) Zhu. 2005. *Semi-Supervised Learning Literature Survey.* Technical Report. University of Wisconsin-Madison Department of Computer Sciences. https://minds.wisconsin.edu/handle/1793/60444 Accepted: 2012-03-15T17:19:12Z.
[48] Morit Zwang, Shahar Somin, Alex 'Sandy' Pentland, and Yaniv Altshuler. 2018. Detecting Bot Activity in the Ethereum Blockchain Network. https://doi.org/10.48550/arXiv.1810.01591 arXiv:1810.01591 [cs].

A METRICS

For a multiset of categorical variables S, we define the entropy as

$$\text{Entropy}(S) = -\sum_i p_i \log_e p_i,$$

where p_i is the proportion of occurrences of the i-th category among all variables in set S [35]. For a given cluster C, entropy is defined as the entropy of the multiset S = comprising the categorical variables within C.

The purity of a cluster is defined as:

$$\text{Purity}(C) = \frac{1}{|C|} \max_i(n_i),$$

where $|C|$ is the size of cluster C and n_i is the number of data points in C that belong to class i [14].

B CLUSTERING

Table 6 shows results where the number of clusters was found automatically by minimizing the Bayesian Information Criterion (BIC) [34] for the GMM and the elbow method for k-means because the BIC would decrease even to a high number of clusters of 100.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

Table 5: Representative examples and references for subcategories of bots.

Subcategory	References and anecdotal evidence
Front-running Insertion	Eskandari et. al. [15]
Sandwich	Qin et.al. [28]
Front-running Displacement	Eskandari et. al. [15]
Front-running Suppression	Eskandari et. al. [15]
Generalized front-running	Robinson and Konstantopoulos [30]
Atomic Arbitrage	Qin et.al. [28]
Statistical Arbitrage	Cartea et.al. [8]
Liquidation	Qin et.al. [28]
Gen. Searching	Konstantopoulos [19]
Sniping	Cernera et. al. [9]
Hot Wallet	Saggese et. al. [32]
Deposit Wallet	Saggese et. al. [32]
Funding & Aggregating	Saggese et. al. [32]
DEX - Custom	https://github.com/bobalice7/PCS-Prediction
Liqu. Compound	https://github.com/lwYeo/Crypto-LP-Compounder
NFT - Custom	https://github.com/What-The-Commit/nft-marketplaces-offer-bot
Parallel Minting	11 consecutive transactions to minting EOAs by address 0x2F3646Ef40734Ca4FE9C0201999824De14EdD823 in block https://etherscan.io/txs?block=14900202&p=3
Game Prog. Action	https://github.com/SansegoTek/DFKQuestRunner
Protocol Update	https://etherscan.io/address/0x69f36eA1ebf4Ec9E53e3aaBF11CAF62b034ff3eE
Rollup	https://etherscan.io/address/0xA173BDdF4953C1E8be2cA0695CFc07502Ff3B1e7
Routine Paym.	https://github.com/s-tikhomirov/pethreon
Airdrop Coll.	https://github.com/jaeaster/airdrop-collectooor https://etherscan.io/address/0xe99c516e18241a699255Bb5317A209fa8980aE7e https://etherscan.io/address/0x5fdA118E9DbFc64DAEce3FDe657eED9333d42F1c
Meta-wallet	https://etherscan.io/address/0x17BF7a2f3f4758909Ba29f59824211D8286356f2 https://etherscan.io/address/0x3ae32939ec8d457f4528881E1E612C6534513476 https://etherscan.io/address/0x727E98A662C20E2b47c0bDA3b1d810f1D805A200
Non. Attr.	https://etherscan.io/address/0xe36Bd1ebD771d5960fC4706d05D6cb03Ab8C3315

Table 6: Average entropy and purity of the combinations of preprocessing methods and clustering algorithms explored with a cluster size optimizing the BIC.

Algorithm	Imputation	Dimension. Reduction	Purity	Entropy	Clusters
GMM	mean	non-UMAP	0.774	0.719	18
		UMAP	0.770	0.726	29
	-1	non-UMAP	**0.789**	**0.606**	30
		UMAP	0.748	0.765	29
kmeans	mean	non-UMAP	0.585	0.970	5
		UMAP	0.652	0.923	5
	-1	non-UMAP	0.622	0.906	6
		UMAP	0.704	0.868	5

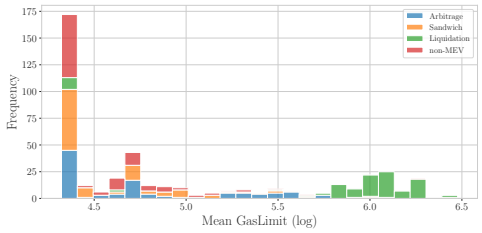

Figure 5: Log distribution of mean gasLimit of classes of the multiclass MEV dataset.

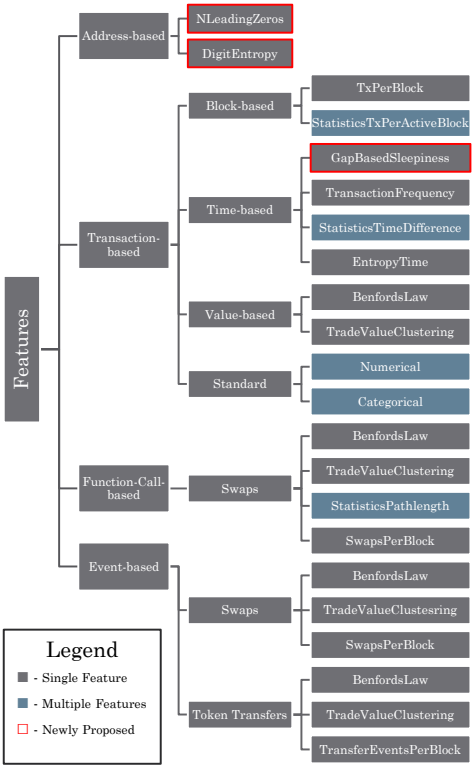

Figure 6: Overview of features.