Financial Innovation

RESEARCH

Open Access

Detecting DeFi securities violations from token smart contract code

Arianna Trozze[1,2*], Bennett Kleinberg[2,3] and Toby Davies[2,4]

*Correspondence:
arianna.trozze@ucl.ac.uk

[1] Department of Computer Science, University College London, Gower Street WC1E 6EA, London, UK
[2] Department of Security and Crime Science, University College London, 35 Tavistock Square, WC1H 9EZ London, UK
[3] Department of Methodology & Statistics, Tilburg University, Warandelaan 2, 5037, AB, Tilburg, Netherlands
[4] School of Law, The Liberty Building, University of Leeds, LS2 9JT Leeds, UK

Abstract

Decentralized Finance (DeFi) is a system of financial products and services built and delivered through smart contracts on various blockchains. In recent years, DeFi has gained popularity and market capitalization. However, it has also been connected to crime, particularly various types of securities violations. The lack of Know Your Customer requirements in DeFi poses challenges for governments trying to mitigate potential offenses. This study aims to determine whether this problem is suited to a machine learning approach, namely, whether we can identify DeFi projects potentially engaging in securities violations based on their tokens' smart contract code. We adapted prior works on detecting specific types of securities violations across Ethereum by building classifiers based on features extracted from DeFi projects' tokens' smart contract code (specifically, opcode-based features). Our final model was a random forest model that achieved an 80% F-1 score against a baseline of 50%. Notably, we further explored the code-based features that are the most important to our model's performance in more detail by analyzing tokens' Solidity code and conducting cosine similarity analyses. We found that one element of the code that our opcode-based features can capture is the implementation of the SafeMath library, although this does not account for the entirety of our features. Another contribution of our study is a new dataset, comprising (a) a verified ground truth dataset for tokens involved in securities violations and (b) a set of legitimate tokens from a reputable DeFi aggregator. This paper further discusses the potential use of a model like ours by prosecutors in enforcement efforts and connects it to a wider legal context.

Keywords: DeFi, Decentralized finance, Ethereum, Fraud, Cryptocurrency, Machine learning, Securities law

Introduction

Decentralized Finance (DeFi) refers to a suite of financial products and services delivered in a decentralized and permissionless manner through smart contracts[1] on a blockchain[2]. Ethereum is a leading example of such a blockchain. The promoters of DeFi have proclaimed it to be the future of finance (Gapusan 2021), an assertion supported by an increase in its market capitalization of more than 8000% between May 2020 and May 2021 (Wintermeyer 2021). Unfortunately, criminal activity in the DeFi ecosystem has grown along with its value. As of August 2021, 54% of cryptocurrency fraud was DeFi-related, compared to only 3% in the previous year (CipherTrace 2021). Furthermore, a vast number of new DeFi projects are created daily and anyone is permitted to create them. Collectively, these present challenges for law enforcement. The volume of projects, coupled with the magnitude of criminal offenses, makes the development of an automated fraud-detection method to guide investigative efforts particularly critical.

Securities violations are a category of crime that affects the cryptocurrency space (Eversheds Sutherland Ltd 2018; Musiala et al. 2020; Podgor 2019). Securities violations refer to offenses related to the registration of securities and misrepresentation related to the purchase or sale of securities, including pyramid schemes and foreign exchange scams (FBI 2021). Preliminary empirical research on decentralized exchanges (one of DeFi's core product offerings) points to the prevalence of specific types of securities violations on these platforms (such as exit scams[3], advance fee fraud[4], and market manipulation) (Xia et al. 2021). Others have chronicled securities violations, such as Ponzi schemes involving decentralized applications (dApps) (Hu et al. 2021).[5] This limited empirical work suggests possible approaches for identifying general scam tokens and certain types of securities violations, such as Ponzi schemes, in the wider cryptocurrency universe. Although we acknowledge the realm of work using opcode-based features to identify malicious activity (see, for example, Santos et al. (2013)), research has not yet explored the automated detection of securities violations (a) defined generally (rather than specific types of securities violations such as scam tokens or Ponzi schemes), (b) across a broader subspace of the DeFi ecosystem (i.e., ERC-20 tokens on all DeFi platforms, instead of a single decentralized exchange), and (c) alongside detailed analyses of the ERC-20 tokens' smart contract code. An automated approach is preferable because of the sheer volume of DeFi projects that exist and are being created.

Against this background, we seek to answer the following research questions: (1) Is a machine learning approach appropriate for identifying DeFi projects likely to violate U.S.

[1] Smart contracts are programs stored on a blockchain that automatically perform specified actions when certain conditions are met (Bartoletti et al. 2020a).

[2] A blockchain is a secure, decentralized database comprised of entries called blocks, which are cryptographically connected to one another through a hash of the previous block, thereby ensuring its security and resistance to fraud. In the case of cryptocurrencies, blockchains serve as a decentralized, distributed public ledger that records all transactions (Binance Academy 2021; Narayanan et al. 2016). In this sense, blockchains underpin the "decentralized" nature of "decentralized finance," as they allow users to transact with one another in a trustless manner without the need for an intermediary financial institution.

[3] Exit scams, also referred to as "rug pulls", involve developers of a project stealing all funds invested into their project (Kamps et al. 2022).

[4] Advance fee fraud refers to a scammer convincing a victim to transfer an amount of money in exchange for returning the original amount plus a premium. The fraudster simply takes the original funds (Trozze et al. 2022).

[5] DApps are the user interfaces of DeFi-based products and services.

securities laws?[6] (2) What are the reasons, at the feature level, for which such a model is or is not successful for this classification problem? This study presents and critically evaluates the first method for the automated detection of various types of securities violations in the DeFi ecosystem based on their token's smart contract code, providing a tool that may identify starting points for further investigation. The contributions of this study are as follows.

- We build a classifier to detect DeFi projects committing various types of securities violations. Our work is the first to expand existing machine learning-based classification models to encompass multiple types of securities violations.
- We use and make available a new dataset of violations verified by court actions.
- Our work is the first to prioritize the explainability of classification decisions in terms of opcode-based features.

Finally, our results contribute to the theory and practice of financial markets. Forecasting, detecting, and deterring financial fraud is critical for maintaining overall financial stability (Shams et al. 2021). In particular, "frauds harm the integrity of financial markets and disrupt the mechanism of efficient allocation of financial resources" (Shams et al. 2021). This is particularly pertinent in the cryptocurrency space (Shams et al. 2021), especially as these markets have become more entwined with traditional markets (Wang et al. 2022).

Decentralized finance (DeFi)

DeFi refers to a collection of financial products and services made possible by smart contracts built on various blockchains, most commonly the Ethereum blockchain. DeFi offers traditional financial products and services such as loans, derivatives, and currency exchange in a decentralized manner through smart contracts. DeFi is an open-source, permissionless system that is not operated by a central authority. Rather than transacting with one another through an intermediary, such as a centralized exchange, user interactions occur through dApps created by smart contracts on a blockchain (Schär 2021). This section describes our DeFi system model and briefly outlines its main components. Because it is the subject of our research, we focus our explanation on the Ethereum-based DeFi space, although DeFi exists on various blockchains.

Ethereum-based DeFi system model

Before explaining DeFi in more detail, we define our system model. The Ethereum-based DeFi system model can be conceptualized as a five-layer system consisting of network, blockchain consensus, smart contract, DeFi protocol, and auxiliary services layers (Zhou et al. 2023). The network layer is concerned with communicating data across and within the various layers. It involves several elements, including network communication protocols and the Ethereum network. In particular, it includes communication among Ethereum peers/nodes. The consensus layer refers to the consensus mechanism of the

[6] Developers write smart contracts in a high-level programming language called Solidity (Cai et al. 2018). Smart contracts are responsible for DeFi's application infrastructure and creating cryptocurrency tokens.

Ethereum blockchain (Zhou et al. 2023). At the time of our research, this was still Proof-of-Work (Wood 2021). The consensus layer also encompasses nodes' actions that rely on the consensus mechanism such as "data propagation," "data verification," executing transactions, and mining blocks. Although the first two layers are implied in our research, this study primarily focuses on the smart contract layer. This includes the smart contract code that creates the ERC-20 tokens from which we derived our dataset, and which creates the dApps that use these tokens. The smart contract layer also includes transactions executed by smart contracts, the Ethereum Virtual Machine (EVM) state, and state transition upon the execution of DeFi transactions. The DeFi protocol layer refers to decentralized applications with which users interact, whereas the auxiliary service layer involves services that facilitate DeFi's functioning, such as wallets and off-chain oracles. We describe these elements in more detail below and provide a visual representation in Fig. 1.

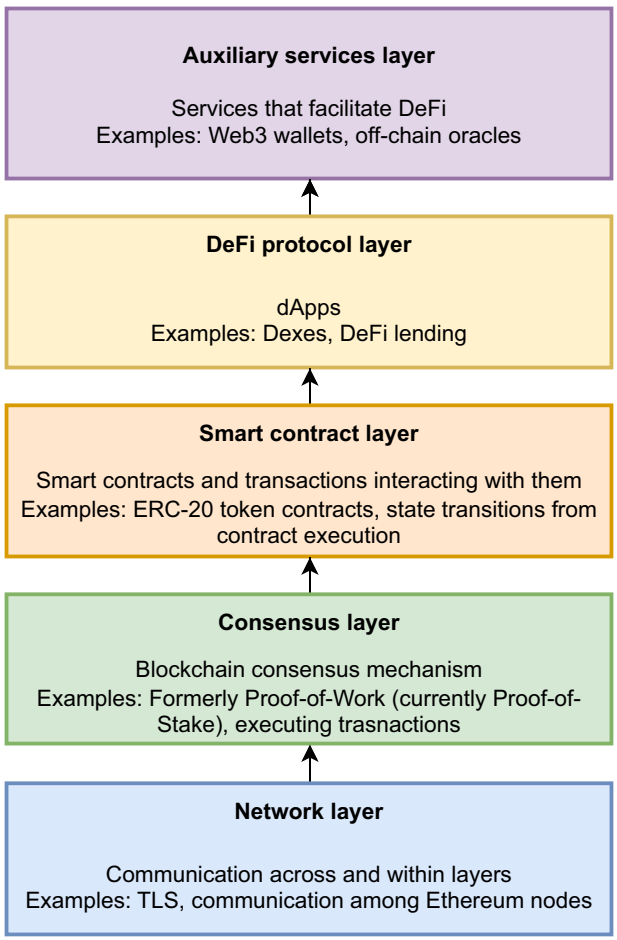

Fig. 1 Ethereum-based DeFi system model. Five layers of the Ethereum-based DeFi system (from bottom to top): network layer, blockchain consensus layer, smart contract layer, DeFi protocol layer, and auxiliary services layer

Ethereum

Ethereum functions as a distributed virtual machine and is the platform on which most of the DeFi ecosystem currently operates. This study focuses on explaining the features of Ethereum that are most relevant to the functioning of DeFi.[7]

In addition to holding balances, Ethereum accounts can store smart contract code and other information. A smart contract is a computer program that automatically performs certain actions when specific conditions—such as payments—are met (Narayanan et al. 2016). Smart contract code is immutable and publicly available on the blockchain. Smart contracts allow parties who do not trust one another to enter into contracts. Rather than trusting each other or a third party to execute the contract, smart contracts ensure that the terms will be executed as coded into the contract (Bartoletti et al. 2020a).

Before smart contracts are executed, they must be compiled for deployment and understanding by the Ethereum Virtual Machine (EVM). Once compiled, Ethereum smart contracts are represented as an array of bytes, often referred to as "bytecode". The EVM is a stack-based environment[8] with a 256-bit stack size. It reads bytecode as operational codes (opcodes), which are sets of instructions (from a set of 144 possible instructions). Opcodes include actions such as retrieving the address of an individual interacting with the contract, various mathematical operations, and storing information (Crytic 2021; Wood 2021).

Tokens

Many DeFi projects also have associated tokens created by smart contracts, which either entitle holders to something within the dApp (analogous to a video game's in-game currency) or serve as "governance tokens." For example, UNI is the governance token for the Uniswap decentralized exchange (Uniswap 2021). Another example is the SCRT token, which, in addition to being a governance token, is required to pay transaction fees on the Secret network (Secret Network 2021). Holders of governance tokens can vote on the future of projects and their voting power is proportional to the number of governance tokens they hold. Most non-NFT DeFi tokens on Ethereum follow the ERC-20 (Ethereum Request for Comment) standard, which facilitates interoperability among projects. The ERC-20 standard allows token capabilities, such as transferring among accounts, maintaining balances, and supplying tokens (BitcoinWiki 2021). In this study, we focus on ERC-20 tokens.

dApps

Developers create dApps that serve as the interfaces to execute these smart contracts. DeFi's current core product offerings, including decentralized exchanges (dexes), lending products, prediction markets, insurance, and other financial products and services, are delivered through dApps (Hertig 2020). Table 1 lists the primary products that constitute the DeFi ecosystem.

Figure 2 illustrates the process of dApp creation and execution through smart contracts, using Uniswap (a popular dex) as an example. As shown in Figure 2, DeFi users must have

[7] For further details on Ethereum, see the Ethereum Yellow Paper (Wood 2021).

[8] In stack-based programming, "all functions receive arguments from a numerical stack and return their result by pushing it on the stack." These specific functions come from a set of pre-defined functions (Perkis 1994).

Table 1 Overview of DeFi products

Decentralized exchanges	These services allow users to exchange cryptocurrencies using liquidity provided by other users, generally through Automated Market Makers, which algorithmically set prices (Xu et al. 2021). Participants can provide liquidity to liquidity pools for certain pairs of cryptocurrencies and receive a Liquidity Provider token for doing so. They can "stake" this token (i.e., lock it into the system and agree not to withdraw it for a certain period) and earn interest on it, usually paid in the decentralized exchange's governance token (referred to as "yield farming"). The return on investment for these yield farms may range from the hundreds to even thousands of percentage points. Participants can stake governance tokens in "pools" and earn further rewards. This incentivizes users to provide liquidity to keep the exchanges running.[a]
DeFi lending	Loans are issued through smart contracts rather than intermediaries and use cryptocurrencies as collateral (Bartoletti et al. 2020b). Loans are often issued in stablecoins-cryptoassets whose value is pegged to government-issued fiat currencies-and interest rates tend to be set algorithmically (Jagati 2021). Users can earn interest for providing liquidity for loans and earn fees from loans. One primary DeFi lending innovation is "flash loans", which are issued and repaid in a single transaction, and therefore do not require collateral (Kamps et al. 2022).
Prediction markets	These allow users to bet on real-world outcomes-such as sporting events or elections—through smart contracts (Binance Academy 2021). Prediction markets rely on blockchain oracles, which are external sources of information that determine the outcome of the prediction market (Kamps et al. 2022). Based on this information about the outcome, the smart contract releases the appropriate funds to the winners (Binance Academy 2021).
DeFi insurance	DeFi insurance community members serve as underwriters and share in premiums paid to the protocol. Holders of the project's governance token vote on claims payouts. DeFi insurance remains a nascent industry, but some companies are attempting to oversee claims directly through smart contracts. Thus far, DeFi insurance tends to insure only other DeFi protocols (Coinbase 2021).
Other financial products	A range of other financial products, including those not usually available to retail investors, can be implemented within DeFi. These include derivatives trading, margin trading, and other securities (DeFi Prime 2021).

[a] For a more detailed discussion of decentralized exchanges, see (Xia et al. 2021).

a Web3 software wallet to hold DeFi tokens and interact with the dApps. These wallets can be considered akin to mobile banking applications and exhibit similar features (sending transactions, showing balances, etc.). However, unlike banking applications, users retain custody of their funds and can send transactions and execute other functions directly rather than through an intermediary institution (Ethereum 2021). Using cryptographic digital signatures, users approve connections to their Web3 wallets, "sign in" to dApps, and approve interactions with the smart contracts on these platforms through their wallet.

(See figure on next page.)

Fig. 2 DApp creation and functioning. The first box contains an excerpt from the Solidity code for the exchange function of the popular dex Uniswap. The second box shows the same code compiled into EVM-readable bytecode. The third box shows the transaction that deployed this code to the Ethereum blockchain, thereby creating the dApp. The branch of Fig. 2 labelled "front-end" shows Uniswap's user interface for a sample exchange operation of 645.49035 USDT to ETH. Below the exchange interface is a screenshot of the Web3 wallet MetaMask. To execute a transaction like the exchange depicted above, the user must connect their Web3 wallet to the relevant dApp via the wallet's browser extension. From there, they can approve the transaction. After the transaction is executed, the user's MetaMask wallet automatically reflects the new balances of these cryptocurrencies. On the back-end, the aforementioned exchange takes the form of bytecode (depicted on the "back-end" branch of the figure), which is executed by the EVM. The final box shows the hash of the executed transaction exchanging USDT for ETH. Full details of this exchange can be found at https://etherscan.io/tx/0x7febc16c960a177077ddf0562c9ba21ac9bd5585bacf969d88a6b678e756081a. The full Solidity source code for the Uniswap V2 smart contract can be found at https://etherscan.io/address/0x7a250d5630b4cf539739df2c5dacb4c659f2488d#code

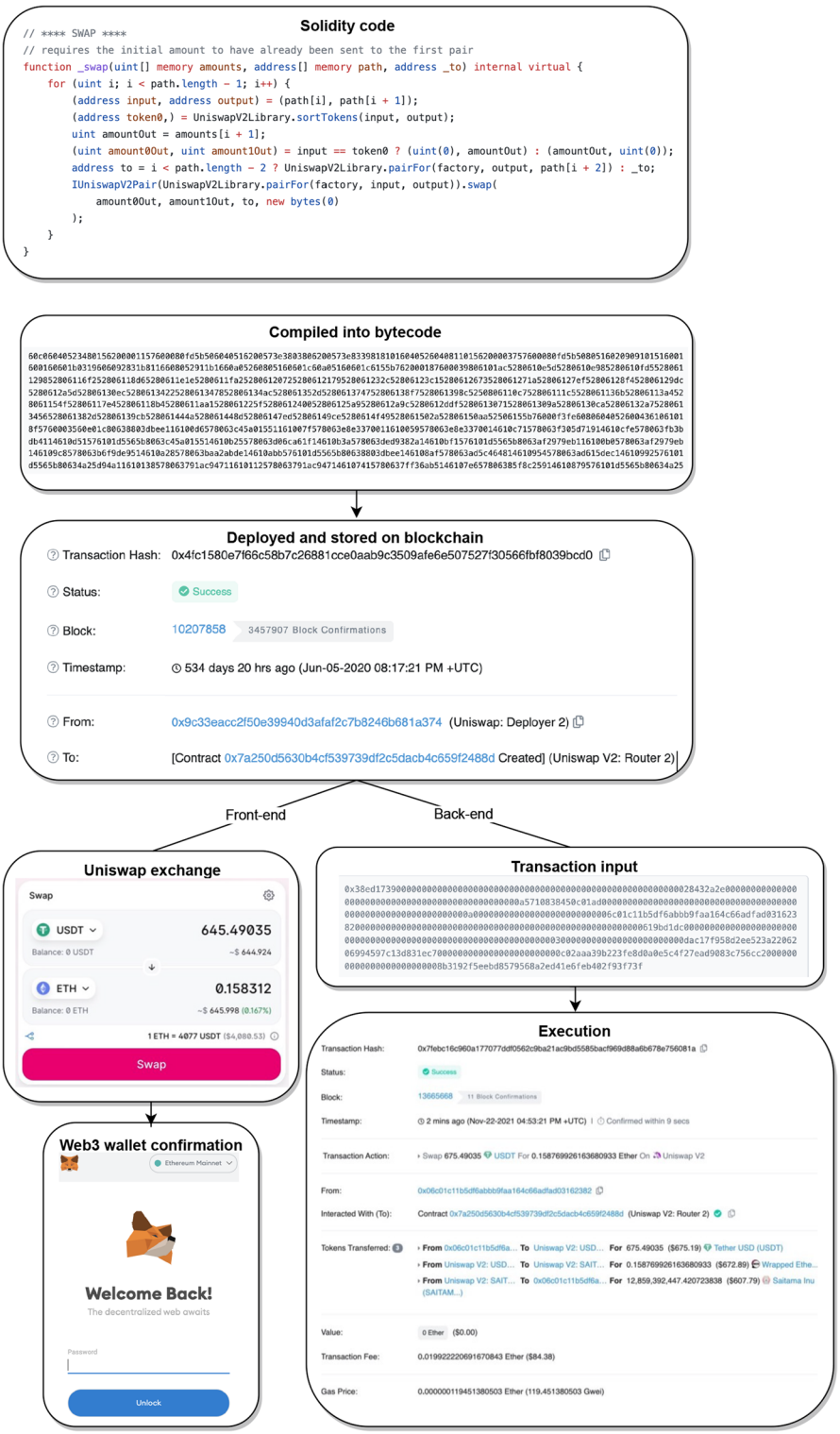

Fig. 2 (See legend on previous page.)

U.S. securities laws

An understanding of U.S. securities law is necessary before defining our DeFi threat model. DeFi has raised alarm in regulatory circles owing to concerns over the potential conflict of DeFi tokens with the existing U.S. securities laws (Blockchain Association 2019). U.S. securities are primarily governed at the federal level by the Securities Act and Exchange Act, although the Sarbanes-Oxley Act, Trust Indenture Act, Investment Advisers Act, and Investment Company Act are also relevant. The Securities and Exchange Commission (SEC) and Financial Industry Regulatory Authority (FINRA) enforce these laws (Practical Law Corporate & Securities 2021).

The Securities Act pertains to the offering and sale of securities. One of the key provisions charged in cryptocurrency cases is Section 5, which requires the registration of the offer and sale of securities and stipulates specific provisions thereof (Practical Law Corporate & Securities 2021).[9] Other sections detail the required registration information[10], and exemptions[11] (Practical Law Corporate & Securities 2021). Various SEC enforcement actions have successfully argued that ERC-20 tokens are securities (see, for example, *Securities and Exchange Commission v. LBRY* (2022)), arguing that they constitute investment contracts (U.S. Securities and Exchange Commission 2019). For further details on the application of the Howey Test (the SEC's criteria for determining whether a digital asset constitutes a security), see (U.S. Securities and Exchange Commission 2019)).

The Exchange Act specifies the reporting requirements of public companies and regulates securities trading through securities exchanges. It also oversees securities fraud. Under Sections 10(b) and 10b-5, fraud and manipulation in relation to buying or selling securities are illegal. One cannot make false or misleading statements (including omissions) in relation to the sale or purchase of securities, including those exempt from registration under the Securities Act. Sections 12 and 15 of the Exchange Act discuss the registration of securities, securities exchanges, brokers, dealers, and analysts (Practical Law Corporate & Securities 2021). Section 12 regulates the registration of initial public offerings, which is relevant for cryptocurrency initial coin offerings (ICOs).[12] Finally, Section 13 relates to companies' reporting obligations under the Exchange Act (Practical Law Corporate & Securities 2021).

In practice, in addition to various registration and reporting violations, U.S. securities laws tend to cover the following fraudulent conduct: high-yield investment programs, Ponzi schemes, pyramid schemes, advance fee fraud, foreign exchange scams, and broker embezzlement (FBI 2021).[13] Financial frauds have been shown to impact financial stability, market integrity, and resource allocation and, in the case of cryptocurrency frauds, to have an impact on markets in traditional finance (Shams et al. 2021; Xin et al. 2018).

[9] For cryptocurrency case law involving the Securities Act, see (Securities and Exchange Commission 2021, 2019, 2018).
[10] Sections 7 and 10.
[11] Section 3, Section 4, Regulation S, Rule 144A, Regulation D, Rule 144, Rule 701, Section 28.
[12] For cryptocurrency case law involving the Exchange Act, see (Securities and Exchange Commission 2021, 2019, 2018).
[13] For definitions of these offenses, see (Trozze et al. 2022).

Threat model

We define our threat model in line with the U.S. securities laws described above. Considering this, an "incident" is any activity that is in violation of these laws, such as failing to register a token as a security or an ICO, or committing securities fraud. These actions may be the result of intentionally malicious behaviors or ignorance of the law. Both cases "result in an unexpected financial loss" for users (Zhou et al. 2023). While we do not have an estimate of all losses incurred as a result of DeFi securities violations, the Finiko Ponzi scheme, for example, took $1.1 billion from victims in 2021 and rug pulls stole $2.8 billion worth of funds from victims in 2021 (Chainalysis 2022). Vulnerabilities that could lead to such incidents exist at the smart contract, protocol, and auxiliary layers of the DeFi system. The smart contract layer includes both the creation of the ERC-20 token itself (in the case of registration violations) and any malicious elements coded into smart contracts such as Ponzi schemes, advance fee fraud, or certain types of exit scams. At the protocol layer, market manipulation is the primary attack vector (Zhou et al. 2023). Finally, at the auxiliary layer, both "operational vulnerability" (such as price oracle manipulation) and "information asymmetry" (such as smart contract honeypots) are observed (Zhou et al. 2023). Information asymmetry primarily occurs in securities fraud. Users are often unable to (or do not take the time to) analyze DeFi protocol smart contracts (and the related security risks) before allowing them to utilize their assets. Users' "understanding of a contract operation" is more likely to come from project marketing materials than from the contract source code itself (Zhou et al. 2023).

This threat model involves several assumptions. The first assumption is that, based on the classification by the SEC of the tokens used to construct our dataset, the DeFi tokens in question are securities under U.S. law. As previously discussed, precedent has been established in this regard. Notably, this also means that many otherwise legitimate projects may operate contrary to U.S. securities laws because they are not appropriately registered. The second assumption is that, in the case of fraud coded into smart contract code, the developers of the DeFi tokens violating securities laws behave maliciously, rather than their violations being the result of errors. Therefore, patching or fixing smart contract code (as discussed in Rodler et al. (2021) and Ferreira Torres et al. (2022)) is not a suitable method for addressing this threat. Registration violations may result from malicious intent or naïve behavior.

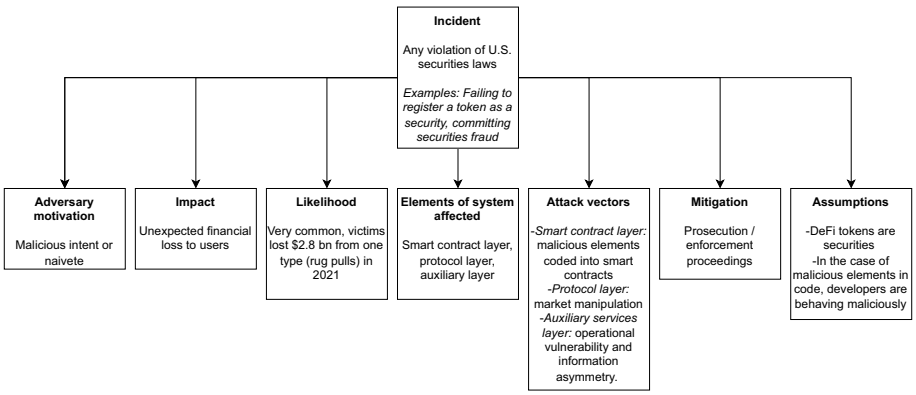

Fig. 3 Threat model. Threat model for violations of U.S. securities laws including attacker motivation, incident impact, elements of the system affected, attack vectors, mitigation, and assumptions

The final assumption is that prevention measures have failed in these instances and, therefore, the primary course of justice is detection and prosecution. While prevention is certainly preferable, it is not possible to prevent all crimes. Therefore, detection and prosecution remain important ways to remedy the threats discussed above. Figure 3 shows a visual representation of the threat model.

Related work on detecting fraud on ethereum

Previous studies have used machine learning to detect specific types of securities violations and fraud on Ethereum. The most common type of securities violation examined in the literature is the smart contract Ponzi scheme (Chen et al. 2021a; Cai et al. 2018; Chen et al. 2018; Fan et al. 2021; Hu and Xu 2021; Hu et al. 2021; Jung et al. 2019; Liu et al. 2022; Wang et al. 2021; Zhang et al. 2021). The various machine learning algorithms employed in these studies to identify such Ponzi schemes—and their relative performance—can be found in Table 2. We acknowledge the existence of various sequential machine learning studies in other contexts (many of which feature more sophisticated classification models). We also note the use of machine learning in cryptocurrency trading research (see, for example, Fang et al. (2022) for a comprehensive review and Sebastião and Godinho (2021) as an empirical example thereof). However, this review is limited to studies that apply sequential machine learning techniques to detect fraud on Ethereum, as the classification problems they seek to solve are most similar to ours.

Previous studies on smart contract Ponzi schemes have examined code-based features (Chen et al. 2021a; Hu and Xu 2021; Fan et al. 2021), transaction-based features (Hu et al. 2021), or both (Wang et al. 2021; Liu et al. 2022; Jung et al. 2019; Zhang et al. 2021; Chen et al. 2018, 2019). Code-based features include the frequency with which each opcode appears in a smart contract and the length of the smart contract bytecode (Jung et al. 2019). Transaction- and account-based features refer to those such as the number of unique addresses interacting with a smart contract and the volume of funds transferred into and out of a smart contract (Jung et al. 2019). One study (Chen et al. 2021a) identified four specific Ponzi scheme typologies based on bytecode sequences.

In addition to smart contract Ponzi schemes, other studies that have examined smart contracts use machine learning to detect general fraud and scams (Chen et al. 2021b; Ibrahim et al. 2021; Lašas et al. 2020; Li et al. 2021; Xia et al. 2021; Fan et al. 2022); advance fee fraud (Wilder 2020); smart contract honeypots (Hu and Xu 2021; Chen et al. 2020); ICO scams (Karimov and Wójcik 2021; Wu et al. 2020); and "abnormal contracts" causing financial losses (Aljofey et al. 2022). Notably, one study (Aljofey et al. 2022) also used features based on contract source code as well as those based on opcodes and transactions.

Most existing studies in this field have examined Ethereum smart contracts in general, but some specifically refer to dApps and DeFi in their work (Fan et al. 2021; Hu et al. 2021; Wang et al. 2021; Li et al. 2021; Xia et al. 2021), although they do so to varying degrees, and occasionally conflate DeFi with Ethereum more broadly. Notably, one study (Hu et al. 2021) used machine learning to classify different types of dApp smart contracts into various categories, including gaming, gambling, and finance.

Table 2 Related work detecting Ethereum Ponzi schemes

Study	Method	Features	Performance
Chen et al. (2021a)	Semantically-aware classifier that includes "a heuristic-guided symbolic execution technique"	Code-based	Precision: 100% Recall: 100% F1: 100%
Fan et al. (2021)	"Anti-leakage" model based on ordered boosting	Code-based	Precision: 95% Recall: 96% F1: 96%
Hu and Xu (2021)	Deep learning model	Code-based	Precision: 96.3% Recall: 97.8% F1: 97.1%
Hu et al. (2021)	Long-term short-term memory neural network	Transaction-based	Precision: Between 88.2% and 96.9% for different types of contracts Recall: Between 81.6% and 97.7% for different types of contracts F1: Between 85% and 96.7% for different types of contracts
Wang et al. (2021)	Long-term short-term memory neural network	Code- and transaction-based	Precision: 97% Recall: 96% F1: 96%
Liu et al. (2022)	Heterogeneous Graph Transformer Networks	Code- and transaction-based	F1: Between 78% and 82% for fraudulent smart contracts and 87% and 89% for normal smart contracts for different classification tasks
Zhang et al. (2021)	LightGBM	Code- and transaction-based	Precision: 96.7% Recall: 96.7% F1: 96.7%
Chen et al. (2018)	XGBoost	Code- and transaction-based	Precision: 94% Recall: 81% F1: 86%
Jung et al. (2019)	Decision trees, random forest, stochastic gradient descent	Code- and transaction-based	Precision: Between 90% and 98% for different models Recall: Between 80% and 96% for different models F1: Between 84% and 96% for different models
Chen et al. (2019)	Random forest	Code- and transaction-based	Precision: Between 64% and 95% for different features Recall: Between 20% and 73% for different features F1: Between 30% and 82% for different features

Gaps and issues

Although the results of the studies described in Table 2 suggest that approaches of this nature can perform well at this task, several points of caution have also been raised. The literature concerning smart contract Ponzi scheme detection points to issues of overfitting owing to the imbalance of classifications in many datasets (Fan et al. 2021). Studies have addressed this issue using over- and under-sampling techniques (Chen et al. 2021a; Fan et al. 2021; Wang et al. 2021; Zhang et al. 2021). Other scholars (Chen et al. 2021a) have criticized the interpretability of results based on opcode features; that is, why the presence of certain opcodes points to criminality. Li et al. (2022) emphasize the importance of interpretability of machine learning models applied in financial contexts. However, these studies have given little consideration to whether machine learning

techniques are necessary for this task, or superior to potentially simpler approaches. Although the applied methods undoubtedly show high performance, it is possible that similar metrics could be achieved without recourse to these types of techniques.

Another issue in previous studies is the repeated use of two particular datasets (Chen et al. 2018; Bartoletti et al. 2019). Of the studies cited above, four used the Bartoletti et al. (2019) dataset (Chen et al. 2021a; Hu and Xu 2021; Liu et al. 2022; Jung et al. 2019), two used the (Chen et al. 2018) dataset (Wang et al. 2021; Zhang et al. 2021), and one study combined both datasets and added additional data (Fan et al. 2021). Although using the same datasets may be helpful for comparing performance, it may be less useful in practice for combating fraud, with any shortcomings of these datasets having a polluting effect on the literature. In fact, when manually inspecting the dataset of Bartoletti et al. (2019), one study (Chen et al. 2021a) identified issues involving duplication and bias. Finally, although it is used to classify general fraud rather than Ponzi schemes, one article used proprietary company data (Li et al. 2021), which hinders the reproducibility and evaluation of the results.

The majority of existing related studies have used smart contracts in general, as opposed to ERC-20 token smart contracts. The only other study that specifically examined DeFi token smart contracts using machine learning is Xia et al. (2021). However, their work focused on scam tokens in general (rather than securities violations) and on a single dApp (the dex Uniswap).

Finally, literature reviews in this field highlight the importance of considering cybersecurity concerns and risk mitigation in cryptocurrency and blockchain research (Xu et al. 2019; Fang et al. 2022).

Aims of this paper

This study seeks to fill these gaps by (a) evaluating whether a machine learning approach is appropriate for identifying DeFi projects likely to engage in securities violations, (b) examining securities violations more comprehensively (rather than just scam tokens or Ponzi schemes), (c) investigating these violations across Ethereum-based DeFi rather than specific subspaces, such as decentralized exchanges, and (d) examining the code-based features identified by our model to better explain its performance. In addition, we develop an entirely new dataset of violating and legitimate tokens.

Method

This study derives methods from prior research on the detection of Ethereum smart contract Ponzi schemes, adapting an approach that performs well in that context (Jung et al. 2019) and applying it to DeFi projects engaging in securities violations. We built various classification models based on the features extracted from DeFi tokens' smart contract code to classify the tokens into two categories: securities violations and legitimate tokens. Figure 4 illustrates the proposed method.

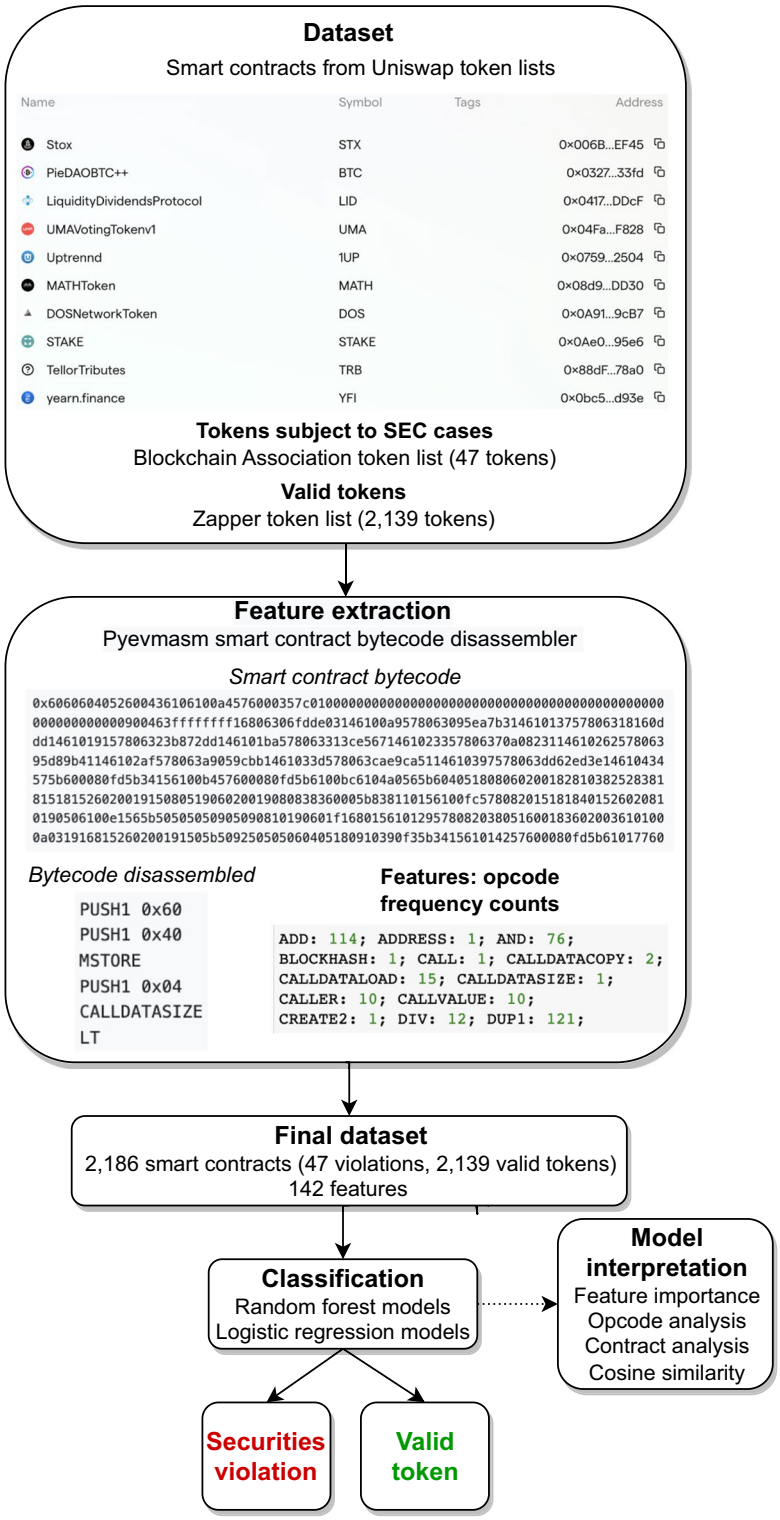

Fig. 4 Methods for detecting securities violations from DeFi token smart contract code

Data collection

To answer our question of whether it can be determined that a project may be engaging in securities violations from its token's smart contract code using machine learning, we first required ground truth sets of both securities violations and legitimate tokens. One source of this information is the token lists compiled by DeFi projects or companies around particular themes. One function of token lists is to help combat token impersonation and scams. Reputable token lists provide users with some assurance that the tokens that appear are not fraudulent. Uniswap posts lists contributed by projects in the community and users generally follow lists from projects that they trust (Uniswap 2020). The lists contain information such as project websites (important for avoiding phishing attempts), symbols, and smart contract addresses.

Securities violations

The Blockchain Association (BA), a lawyer-led blockchain lobbying organization, created a list of ERC-20 tokens subject to U.S. SEC enforcement actions.[14] At the time of our study, this list contained 47 tokens and these served as our ground truth for identifying projects engaged in securities violations. Many of these actions involve Initial Coin Offerings (ICOs), primarily in the context of companies or individuals failing to register their token as a security when required and/or making fraudulent misrepresentations in connection with the said token (e.g., Coinseed Token, Tierion, ShipChain SHIP, SALT, UnikoinGold, Boon Tech, and others) (U.S. Securities and Exchange Commission 2022). Other violations include Ponzi schemes (e.g., RGL) and market manipulation (e.g., Veritaseum) (*Securities and Exchange Commission v. Natural Diamonds Investment Co., et al.* (2019a), *Securities and Exchange Commission v. Reginald Middleton, et al.* (2019b)). The defendants in these cases are distinct, thereby supporting the independence of the tokens in our violation dataset. We acknowledge the limitations of using such a small set; however, to date, this set comprises all SEC actions involving DeFi tokens. Therefore, it provides a more credible ground truth dataset than searching for individual investment scams and securities violations on blockchain forums (as other datasets do, including (Bartoletti et al. 2019)), because it is more systematic and does not involve any subjective judgment of wrongdoing. Notably, the nature of the list itself highlights the need for a systematic detection method. Most of the actions were either derived from the U.S. government whistleblower program or were well-publicized scams, suggesting that enforcement is currently reliant on these sources (U.S. Securities and Exchange Commission 2021b).

Legitimate projects

The nature of the DeFi industry means that a substantial proportion of tokens may be of questionable validity, even if they have not been formally identified as violations. This poses a challenge when building a dataset of legitimate tokens. If we were to randomly sample all projects, it is likely that problematic tokens would be included, compromising our analysis. Therefore, we adopted an alternative approach that included only tokens for which we had some evidence of credibility. To do this, we used the token list maintained by the DeFi platform Zapper.[15] Zapper is a DeFi project aggregator that allows users to

[14] https://tokenlists.org/token-list?url=https://raw.githubusercontent.com/The-Blockchain-Association/sec-notice-list/master/ba-sec-list.json.

[15] https://tokenlists.org/token-list?url=https://zapper.fi/api/token-list

monitor their liquidity provision, staking, yield farming, and assets across different DeFi protocols. As of November 2021, Zapper had over one million users, $11 billion worth of transaction volume, and raised $15 million in venture funding (Zapper 2021). While they do not claim to provide financial advice to users, they make an effort to internally vet the projects they list, opting for those with audited contracts and reputable teams (zes 2020). To be clear, inclusion in this list does not provide any indication of the "quality" of a token—it is not analogous to a list of "blue chip" stocks—but simply an indication of authenticity. The Zapper list contains 2,146 ERC-20 tokens, which we used as the ground truth for the legitimate tokens. This may not be as representative of DeFi tokens in general as a random sample, but it is the best available source of tokens that have some marker of credibility.

Final dataset

We extracted the smart contract addresses for the ERC-20 tokens on both lists and combined them into a single dataset, with a binary indicator added to flag violations. This provided an initial dataset of 2,193 smart contract addresses. Seven tokens were present in both lists. These likely represent tokens that are otherwise legitimate but violated U.S. securities laws by failing to register as securities. Since the Blockchain Association list is verified by court actions, we removed these from our "legitimate" token set. This also shows that our dataset captures projects that occupy the "middle ground" with respect to legitimacy, rather than only at extremes of offending and non-offending. Thus, the final dataset consisted of 2186 tokens (47 of which were the subject to an SEC case in the U.S., where the individuals or company that created or marketed the token broke securities laws). Our final dataset (including the features described below) can be found here: https://osf.io/xcdz6/?view_only=5a61a06ae9154493b67b24fa4979eddb.

Features

Next, we used the Web3 Python package (web3.py. ethereum 2023) to extract the bytecode for each token in our dataset. We used an Infura node that allows users to interface with the Ethereum blockchain through nodes that the company runs, using their API.[16] Our classification features come from the token smart contract bytecode we collected. We opted to use only code-based features (rather than transaction-based features, for example), following other recent studies that achieved high levels of performance (including 100% precision and recall in Chen et al. (2021a)) in classifying Ethereum-based smart contract Ponzi schemes (Chen et al. 2021a; Fan et al. 2021; Hu et al. 2021). Furthermore, using only code-based features allows for classification as soon as smart contracts are deployed (Jung et al. 2019), rather than waiting to examine the characteristics of associated transactions, and permits the analysis of smart contracts with few transactions (Chen et al. 2021a).

For this initial analysis, our aim was to maintain the classifier's simplicity and computational inexpensiveness. It is also the first classifier for Ethereum-based DeFi securities violations more broadly; hence, our aim was to obtain a baseline for this novel classification problem in order to determine whether it is suited to machine learning, rather than improving the state-of-the-art for previously addressed problems (as (Fan

[16] https://www.infura.io/.

et al. 2021; Chen et al. 2021a) and others have done for smart contract Ponzi scheme classification).

The EVM bytecode can be computationally "disassembled" into its corresponding opcodes. This process is illustrated in Fig. 4. Following (Jung et al. 2019), we included a feature in our classifier for each opcode that appeared in our smart contracts, representing the frequency with which the opcode appeared in any given smart contract. We used the `Pyevmasm` Python package (Crytic 2020) to disassemble each contract's bytecode into its equivalent opcodes and then used a counter to determine the number of times each opcode appeared in the contract.

Feature exploration

Prior to building any classification models, we used Elastic Net regression (with $alpha = 0.001$ and the data mean-centered and normalized using Scikit-learn's `StandardScaler`)[17] to investigate the importance of these features. Table 3 lists the top 10 non-zero coefficients of the model.

Table 3 Feature exploration using Elastic Net regression

Feature	Coefficient
SWAP2	0.015
DUP2	0.014
SHL	0.013
PUSH30	0.012
PUSH21	0.012
PUSH9	0.011
LOG4	0.010
MLOAD	0.008
SLT	0.007
RETURN	0.007

Table 4 Opcode descriptions (Crytic 2021)

Opcode	Description
SWAP*n*	Exchange first and *n*th stack item
DUP*n*	Duplicate *n*th stack item
SHL	"Shift left"
PUSH*n*	Put *n*-byte item on the stack
LOG*n*	Append log record with *n* topics
MLOAD	Load a word previously saved to memory
SLT	"Signed less-than comparison"
RETURN	Stop code execution and return output data

[17] Elastic Net regression combines the ridge penalty (which reduces coefficients of correlated variables) and the lasso penalty (which chooses one of the correlated variables and eliminates the others). The alpha value sets this penalty, with $alpha = 0$ for full ridge regression and $alpha = 1$ for lasso (Hastie et al. 2023). We chose $alpha = 0.001$ and used Scikit-learn `StandardScaler` to pre-process our data to enable convergence of our model. The `StandardScaler` pre-processes the features in a dataset by "removing the mean and scaling to the unit variance" (scikit-learn developers 2023).

Overall, 55 features had non-zero coefficients in our Elastic Net regression model. However, none of these coefficients were particularly large. This is expected because each line of the Solidity code is ultimately translated into several opcodes, which means that multiple opcodes can capture the same behavior or action. Table 4 describes the opcodes listed in Table 3.

Classification

First, we used a random forest classifier to determine whether a project was potentially engaged in securities violations. We chose a random forest classifier for the following reasons.

1 Research involving data similar to ours achieved the best classification results with a random forest, compared with other classifiers ((Xia et al. 2021); precision: 96.45%, recall: 96.79%, F-1: 96.62%).
2 While initial work on smart contract Ponzi schemes (Jung et al. 2019) has been optimized in later studies (for example, Fan et al. (2021)), our goal is to achieve a baseline of performance for classifying Ethereum-based DeFi securities violations. Previous work (Jung et al. 2019) found the random forest algorithm performed the best on their dataset, when compared with other standard classification algorithms (J48 decision tree and stochastic gradient descent).
3 Given our primary goal to determine if machine learning methods are suitable for developing a classifier that is useful for law enforcement investigations, the use of a model with greater transparency and traceability is most informative. Prior research using machine learning techniques in financial applications (such as Li et al.'s (2022) research, which explores a novel approach to clustering using ten different financial datasets) also underscores the importance of feature interpretability in these contexts (Li et al. 2022).

Given the classification imbalance in our data, we used downsampling of the majority class to balance it with the minority class. Specifically, we randomly sampled 47 smart contracts from the majority class (i.e., from the $n = 2,139$ legitimate contracts) and ran a random forest classifier on the resulting balanced dataset (i.e., 47 violations versus 47 legitimate tokens). This procedure was repeated 100 times with different random samples and the average performance of these 100 iterations was reported.[18]

For each iteration, we used 70% of our data to train our model and 30% for our test set, following previous studies on classifying smart contract Ponzi schemes (Wang et al. 2021). We calculated the accuracy, weighted precision, recall, and F-1 score to evaluate our model (Prellberg and Kramer 2020). We calculated the means of these metrics across 100 iterations to obtain the final performance scores. We analyzed the average feature importance across 100 iterations of our model and then built several subsequent models based on this information.

[18] Undersampling, combined with properly executed cross-validation, performs well on highly imbalanced datasets (Blagus and Lusa 2015). Whereas other works related to ours (Fan et al. 2021) used the Synthetic Minority Over-Sampling Technique (SMOTE) to train imbalanced data, we chose the more conservative undersampling method. The SMOTE combines majority class undersampling and minority class oversampling, and synthesizes additional data for the minority class (Chawla et al. 2002).

For the aforementioned reasons, we focused on random forest classification. To comprehensively answer our first research question regarding the suitability of machine learning for this classification task, we needed to build multiple kinds of models, including a simpler approach. Therefore, we also built logistic regression models using our data. Similar to our random forest classifier, we used downsampling across 100 iterations, a 70–30% train-test split, and calculated the accuracy, weighted precision, recall, and F-1 score metrics. After analyzing feature importance, we constructed further models using different sets of features.[19]

Results

Classification

Random forest

Although we used the results of our Elastic Net-based feature exploration as input for some of our models, we performed further feature exploration with random forest models since none of the coefficients were notably large. We built our initial classification model using the frequency of all opcodes contained in our dataset (142 features), employing bootstrapped undersampling to evenly balance the classes in our dataset over 100 iterations. As is evident from the evaluation metrics shown in the top row of Table 5, we achieved satisfactory performance with this model compared with our baseline (50%). We then calculated the relative importance of the features included in the model. The results are listed in Table 6.

Next, we built models using only the 10 features with the highest importance in our original model, the three most important features, and the single most important feature (CALLDATASIZE). The weighted precision, recall, F-1 score, and accuracy are presented in Table 5. We also built models with all the non-zero coefficients of our Elastic Net regression model (a total of 55 features), calculated the feature importance for this model, and then used this information to build models with the 10 features with the

Table 5 Random forest model performance with undersampling

	Accuracy	Precision	Recall	F-1 score
RF1: Full-feature model	0.759	0.757	0.759	0.757
RF2: Top 10 features of model RF1	**0.801**	**0.800**	**0.800**	**0.800**
RF3: Top 3 features of RF1	0.780	0.780	0.780	0.780
RF4: Top feature of RF1 (CALLDATASIZE)	0.777	0.774	0.777	0.774
RF5: Non-zero coefficients of EN regression model	0.731	0.731	0.731	0.731
RF6: Top 10 features of RF5	0.743	0.742	0.743	0.742
RF7: Top 3 features of RF5	0.741	0.741	0.741	0.741
RF8: Top feature of RF5 (LT)	0.747	0.747	0.747	0.747
RF9: Top 10 features of EN regression model	0.690	0.689	0.690	0.689

The final model is indicated in bold

[19] We did not build any more complex machine learning models (like neural networks) to answer our first research question because our dataset was much smaller than those traditionally used to train deep learning models. Neural networks are much less interpretable than simpler machine learning models (Choi et al. 2020) and would therefore be less suitable for our purposes (where the results could potentially be involved in legal proceedings) in any case.

Table 6 Feature importance for random forest models with undersampling

RF1		RF2		RF3		RF5		RF6		RF7		RF9	
Feature	**Imp.**	**Feature**	**Imp.**	**Feature**	**Imp.**	**Feature**	**Imp.**	**Feature**	**Imp.**	**Feature**	**Imp.**	**Feature**	**Imp.**
CALLDATASIZE	0.062	CALLDATASIZE	0.203	CALLDATASIZE	0.340	LT	0.077	LT	0.165	LT	0.357	SWAP2	0.206
LT	0.034	LT	0.125	CALLVALUE	0.334	CALLVALUE	0.075	CALLVALUE	0.158	CALLVALUE	0.340	DUP2	0.192
CALLVALUE	0.032	CALLVALUE	0.123	LT	0.326	CALLER	0.046	CALLER	0.111	CALLER	0.303	MLOAD	0.180
SHR	0.029	SWAP3	0.118			DUP2	0.043	SWAP2	0.094			RETURN	0.152
EXP	0.026	EXP	0.110			SWAP2	0.043	DUP2	0.094			SHL	0.122
SWAP3	0.026	CALLER	0.089			MLOAD	0.039	MLOAD	0.088			PUSH21	0.061
NUMBER	0.023	SHR	0.072			DUP7	0.039	DUP7	0.084			SLT	0.031
PUSH5	0.021	NUMBER	0.063			DUP5	0.033	DUP5	0.074			LOG4	0.020
CALLER	0.018	PUSH5	0.055			GT	0.033	GT	0.071			PUSH30	0.019
ADDRESS	0.018	ADDRESS	0.041			DUP11	0.032	DUP11	0.061			PUSH9	0.018

Table 7 Additional opcode descriptions (Crytic 2021)

Opcode	Description
CALLDATASIZE	Retrieve size of "input data in current environment"
LT	"Less-than comparison"
CALLVALUE	Amount deposited by current transaction/instruction
EXP	"Exponential operation"
CALLER	Get address of caller
SHR	"Logical shift right"
NUMBER	Retrieve block number
ADDRESS	Retrieve address of account executing transaction
GT	"Greater-than comparison"

highest importance in this 55-feature model, the top three features, and the top feature (LT). Finally, we built a model using the top 10 non-zero coefficients in our Elastic Net regression model as our features. We assessed the feature importance for all subsequent models, as reported in Table 6. Table 7 lists the opcodes whose frequency in the smart contracts was determined to be of high importance to the models that were not previously described.

Using the F-1 score as our primary metric, we achieved the best performance with RF2, which was our 10-feature model built using the top 10 features from our full-feature model (RF1). This model performed relatively well (F-1 score of 80%) compared with our baseline of 50%. Three features—the frequencies of CALLDATASIZE, LT, and CALLVALUE—were the most important across all random forest models except RF9.

Logistic regression

To answer our research question of whether machine learning is appropriate for identifying DeFi projects likely to be engaging in violations of U.S. securities laws, we built a logistic regression model to see if a simpler model could correctly classify our data. We used the same bootstrapped undersampling as we did when constructing our random forest models, subsequently calculated the feature importance, and built further models accordingly. We report the accuracy, and weighted precision, recall, and F-1 scores for these models in Table 8 and the feature importance for the top 10 features in Table 9. Table 10 lists the opcodes among the features reported in Table 9 that have not been previously described.

Table 8 Logistic regression model performance with undersampling

	Accuracy	Precision	Recall	F-1 score
LR1: Full-feature model (using standard scaler)	0.739	0.738	0.739	0.738
LR2: Top 10 features of LR1	0.638	0.634	0.638	0.634
LR3: Top 8 features of LR1	0.639	0.634	0.639	0.634
LR4: All non-zero coefficients of EN regression (using standard scaler)	0.725	0.724	0.725	0.724
LR5: Top 10 features of LR4	0.606	0.601	0.606	0.601
LR6: Top 9 features of LR4	0.602	0.596	0.602	0.596
LR7: Top 10 features of EN regression model	0.648	0.646	0.648	0.646

Table 9 Feature importance for logistic regression models with undersampling

LR1		LR2		LR3		LR4		LR5		LR6		LR7	
Feature	**Imp.**	**Feature**	**Imp.**	**Feature**	**Imp.**	**Feature**	**Imp.**	**Feature**	**Imp.**	**Feature**	**Imp.**	**Feature**	**Imp.**
SWAP15	0.396	PUSH30	0.637	CALLVALUE	0.043	SWAP7	0.515	BALANCE	0.260	PUSH30	0.701	PUSH30	0.671
RETURN	0.340	SWAP15	0.514	CODESIZE	0.484	CALLVALUE	0.514	CALLER	0.023	SWAP15	0.389	LOG4	0.215
PUSH30	0.299	CODESIZE	0.443	EXP	0.020	PUSH9	0.430	CALLVALUE	0.058	BALANCE	0.215	PUSH21	0.168
CALLVALUE	0.287	PUSH31	0.146	LOG1	0.070	RETURN	0.427	LOG1	0.145	LOG1	0.165	RETURN	0.140
LOG1	0.253	LOG1	0.102	PUSH30	0.592	SWAP15	0.382	PUSH30	0.706	PUSH31	0.088	SLT	0.118
PUSH9	0.248	PUSH9	0.091	PUSH31	0.184	PUSH31	0.377	PUSH31	0.124	SWAP7	0.064	PUSH9	0.083
PUSH31	0.233	CALLVALUE	0.050	PUSH9	0.025	PUSH30	0.328	PUSH9	0.038	CALLVALUE	0.061	DUP2	0.022
EXP	0.230	EXP	0.022	SWAP15	0.590	LOG1	0.271	RETURN	-0.022	CALLER	0.012	MLOAD	0.007
SWAP7	0.197	RETURN	-0.030			CALLER	0.266	SWAP15	0.450	PUSH9	0.003	SHL	-0.031

Table 10 Additional opcode descriptions (Crytic 2021)

Opcode	Description
CODESIZE	"Size of code running in current environment"
BALANCE	Retrieve account balance

Overall, the logistic regression models performed worse than the random forest models. Using the weighted F-1 score as our primary metric, our best-performing logistic regression models were those with the most features, namely, our 142-feature (LR1, with an F-1 score of 73.8%) and our 55-feature models (LR4, with an F-1 score of 72.4%). The other logistic regression models performed closer to our baseline (50%).

There was little overlap in the most important features of our logistic regression and random forest models (except in those built with the top 10 features of our Elastic Net regression model). EXP was one of the most important features in certain random forest and logistic regression models (RF1, RF2, LR1, LR2, LR3). CALLVALUE was among the top 10 most important features for all our models, aside from those built using the top 10 non-zero coefficients of our Elastic Net regression model. CALLER, which was among the top 10 most important features for five of our random forest models, also had high levels of feature importance in logistic regression models LR4, LR5, LR6.

Using the weighted F-1 score as our primary metric, we achieved the best performance (an F-1 score of 80%) with RF2, which is a 10-feature random forest model. Therefore, this was the final model.

Opcodes

To better understand the performance of our final model, we compared the frequencies with which the 10 opcodes from our final model occurred in each of our classes (violations and legitimate tokens). A t-test was conducted to assess whether the

Table 11 Mean comparisons of opcode frequencies and t-test results with Cohen's d effect size

	Securities violations		Legitimate tokens					
Opcode	Mean	St. dev	Mean	St. dev	*t*-value	*p*-value	Effect size (Cohen's d)	CI(95%)
CALLDATASIZE	2.745	4.245	11.387	8.192	−7.210	<0.001	−1.063	[−1.354, −0.772]
LT	9.404	9.124	18.676	14.799	−4.277	<0.001	−0.631	[−0.920, −0.341]
CALLVALUE	17.957	10.449	10.497	13.658	3.720	<0.001	0.549	[0.259, 0.838]
SWAP3	24.723	17.501	37.273	25.214	−3.394	<0.001	−0.500	[−0.790, −0.211]
EXP	36	34.075	17.751	25.187	4.871	<0.001	0.718	[0.428, 1.008]
CALLER	16.851	9.648	11.768	11.244	3.074	0.002	0.453	[0.164, 0.743]
SHR	0.085	0.282	0.818	1.361	−3.688	<0.001	−0.544	[−0.833, −0.254]
NUMBER	0.063	0.323	1.712	2.196	−5.14	<0.001	−0.758	[−1.048, −0.468]
PUSH5	0.362	1.206	2.198	2.908	−4.321	<0.001	−0.637	[−0.927, −0.348]
ADDRESS	1.255	2.982	2.529	3.255	−2.658	0.008	−0.392	[−0.681, −0.103]

average frequencies were significantly different. The findings from these comparisons are reported in Table 11, and they support the analysis of feature importance in our final model. The mean frequencies of each feature in our final model (reported in Table 11) were significantly different between the securities violations and legitimate token sets, with a much larger effect size for the most important feature (CALLDATASIZE) than for the other features.

Analyzing solidity code

To better understand the top three features of our final model, we randomly selected five contracts from our set of securities violations and five contracts from our set of legitimate tokens and analyzed their Solidity code. The contracts, frequencies with which our model's top three features occurred in their code, the version of Solidity in which their code was written can be found in Table 12.

We used Etherscan,[20] the Ethereum blockchain explorer, to obtain the Solidity code for each of these tokens. Next, we used Remix[21], an Ethereum Integrated Development Environment that allows users to write, compile, deploy, and debug Ethereum-based smart contracts, including in virtual environments, to analyze the code. We compiled and deployed each smart contract using the Remix virtual machine.

We used Remix's "debugger" tool to analyze the transactions deploying each section of the compiled contracts. The debugger tool allows users to examine the opcodes for each transaction chronologically and highlights the corresponding line of the Solidity code for each opcode (each line of the Solidity code is compiled as several opcodes). It also provides information on the functions with which the transaction interacts, local Solidity variables, Solidity state variables, and other information (Remix 2022a, b). However, given that our goal was to better understand the features of our final classification model, our analysis focused on the opcode tool.

Table 12 Features and Solidity version for contracts analyzed

	Frequency of opcodes			Solidity version
	CALLDATASIZE	CALLVALUE	LT	
Violating tokens				
Gladius	5	17	11	0.4.15
Tierion Network Token	1	15	3	0.4.13
Dropil	1	16	6	0.4.18
OpportyToken	1	12	3	0.4.15
Boon Tech	3	25	18	0.4.19
Legitimate tokens				
Sparkle Loyalty	1	19	7	0.4.25
Prometeus	8	12	6	0.4.23
ARC Governance Token	11	1	18	0.5.0
Social Finance	9	22	5	0.4.23
OST	1	26	6	0.4.17

[20] https://etherscan.io/.
[21] https://remix.ethereum.org/.

We examined all elements of the smart contracts involved in their deployment transactions. Each time one of our target opcodes appeared, we noted the specific aspect of the token smart contract and the corresponding line of the Solidity code.

Though it is difficult to ascertain patterns that may be picked up by our classifier from a visual examination of our code, four of our five violating contracts (Dropil, Tierion Network Token, OpportyToken, and Boon Tech) had the same line of code that was resolved to the CALLVALUE opcode in the SafeMath portion of the smart contract: `library SafeMath {`. In our legitimate token smart contracts, the CALLVALUE opcode was present in only one of the five token contracts (the OST contract) in the SafeMath part of the contract. SafeMath is part of the OpenZeppelin smart contract development library, which allows developers to import standard, vetted, and audited Solidity code, such as for ERC-20 tokens (OpenZeppelin 2023). The SafeMath library, in particular, provides overflow checks for arithmetic operations in Solidity; arithmetic operations in Solidity "wrap" on overflow, which can lead to bugs and which attackers could exploit (OpenZeppelin 2023). SafeMath solves this issue by reverting transactions that result in operational overflow (OpenZeppelin 2023).[22]

Additionally, when subsequently specifying the implementation of SafeMath for various arithmetic operations, the violating tokens use the "constant" function modifier, as opposed to the "pure" modifier used in the legitimate token smart contracts.[23] These modifiers dictate whether a given function affects the global state of Ethereum. The use of "constant" indicates that no data from the function is saved or modified, while "pure" adds the attribute that the function also does not read blockchain data (Nabi 2022). While both attributes specify that the function will not write to the Ethereum state, in the case of "pure," the function also does not read state variables (Modi 2018). The "pure" attribute, being stricter about state modification, provides stronger assurance that arithmetic operations resulting in overflow will not (incorrectly) modify the contract's state. Legitimate token contracts would intuitively provide this additional security and specificity.

We previously noted that each line of Solidity code in a smart contract resolves to multiple opcodes. Our smart contract analysis highlights this point. In previous iterations of our model, JUMPDEST was among the most important features. Various lines of the Solidity code, such as `Contract BasicToken is ERC20Basic }`, involve both this opcode and CALLVALUE. However, we did not notice any distinctions in these lines of code between the violating and legitimate token classes.

CALLVALUE and LT were present numerous times in the aspects of the smart contracts we analyzed with the Remix debugger tool. However, based on our disassembly of these tokens' bytecode, the full frequencies were not present in the portions of the smart contracts we analyzed. The CALLDATASIZE opcode was absent. However, we

[22] Notably, the SafeMath library was rendered superfluous by Solidity releases 0.8.0 and above (0.8.0 was released in December 2020 (Solidity Team 2020; Solidity Dev Studio 2020). We initially hypothesized that violating tokens could utilize older versions of Solidity rather than legitimate ones because of the lengthy nature of the U.S. justice process. However, further inspection of the Solidity code for each token in our sample revealed that they all relied on Solidity versions between 0.4.13 and 0.5.0 (as shown in Table 12), and all but Gladius include it in their code.

[23] In later versions of Solidity, the "constant" modifier was changed to "view" (The Solidity Authors 2023).

were unable to execute other transactions in the Remix virtual environment to analyze the entirety of the smart contracts (although we explored a significant portion thereof). Furthermore, given our aim of using code-based features to develop a classifier that can be used immediately upon the deployment of a token contract, these are the aspects that are most relevant for our purposes.

Comparing smart contracts

Given our conclusions regarding the implementation of a particular library in the code as a distinguishing factor between violating and legitimate token contracts, we sought to delve further into potential code reuse as a reason for the performance of our classifier. Prior studies have found that 96% of Ethereum smart contracts contain duplicative elements (although it is unclear if this is the case in the Ethereum-based DeFi ecosystem specifically) (He et al. 2019). In this sense, if legitimate projects borrow code from other legitimate projects and projects violating securities laws borrow from other violating projects, smart contracts within each class will have a high degree of internal consistency.

Cosine similarity for solidity code

We used cosine similarities to analyze code reuse among the Solidity code of the smart contracts that we analyzed individually. To do this, we tokenized[24] the Solidity code using FastText (Meta Research 2023) and then calculated the cosine similarities between the vectors for each possible combination of the 10 contracts. The cosine similarity measures the level of similarity between two vectors and is bound to the range of −1 to 1. A cosine similarity of −1 means that the two vectors are perfectly opposite, 1 means they are identical, and 0 means that the two vectors are orthogonal to one another (Han et al. 2012). If code reuse were indeed a possible explanation, we would expect the difference between the cosine similarities within each class and those between violating and legitimate contracts to be more pronounced for violating smart contracts than for legitimate smart contracts. Subsequently, we compared the means of the cosine similarities for each token class. Our results are reported in Table 13.

Table 13 Cosine similarity of smart contract Solidity code

	Cosine similarity		Comparison with inter-class similarity			
Class	**Mean**	**St. dev**	**t-value**	***p*-value**	**Cohen's d**	**CI(95%)**
Securities violations	0.934	0.053	−0.906	0.3711	−0.337	[−1.067, 0.392]
Legitimate	0.962	0.028	0.652	0.519	0.252	[−0.506, 1.010]
			Comparison of violations and legitimate similarities			
Inter-class	0.951	0.048	1.361	0.191	0.625	[−0.275, 1.523]

[24] "Token" is used here in the sense of natural language processing, to refer to a portion of text (i.e., a word). It differs from the use of "token" in the rest of this article.

The Solidity code was not significantly different among our classes (at least as per the cosine similarity). We noted generally high levels of similarity among smart contracts. Because of the existence of token standards, this is as expected. Based on these results, it is unlikely that, in the case of these 10 contracts, code reuse explains our classifier's performance.

Cosine similarity of feature-based vectors

Next, we compared the opcode frequencies for the smart contracts to one another using cosine similarity. This comparison was intended to assess whether the opcodes generated from the tokens' Solidity code suggested code reuse could impact our classifier's performance. In this experiment, we used the cosine similarity of the vectors of the features rather than the Solidity code itself, in contrast to our first cosine similarity experiment, which revealed significant similarities among ERC-20 token smart contracts owing to code and token standards. We hypothesized that using opcode frequencies would better capture the nuances in the code. We converted the frequencies of the opcodes in each smart contract into vectors and compared them. We calculated the cosine similarity for each possible combination of (a) violating smart contracts, (b) legitimate smart contracts, and (c) violating and legitimate smart contracts ("inter-class"). For each set, we obtained the average of the calculated cosine similarities, the results of which are presented in Table 14. We also compared the cosine similarities for the set of violating contracts and the set of legitimate tokens with the interclass cosine similarities using t-tests. These results are also reported in Table 14.

Our results show that legitimate smart contract opcodes are slightly more similar than violating contract opcodes (0.054 and 0.040, respectively). The violating contracts' opcodes are less similar to each other than to legitimate contracts' (at least per cosine similarity, which is 0.052 for the inter-class cosine similarity and 0.040 for the violating class). This suggests that there may be slightly more code reuse among legitimate contracts than violating ones. However, the cosine similarity for both groups was rather small, as were the effect sizes, suggesting that overall code reuse is unlikely to be the primary reason for our classifier's performance (although this does not preclude the possibility of certain elements of the code, such as the use of the SafeMath library, being at least partially responsible).

Table 14 Cosine similarity of smart contract opcode frequencies

	Cosine similarity		Comparison with inter-class similarity			
Class	**Mean**	**St. dev**	**t-value**	***p*-value**	**Cohen's d**	**CI(95%)**
Securities violations	0.040	0.024	− 8.561	<0.001	− 0.262	[− 0.322, − 0.202]
Legitimate	0.054	0.062	7.719	<0.001	0.025	[0.019, 0.031]
			Comparison of violations and legitimate similarities			
Inter-class	0.052	0.048	7.381	<0.001	0.225	[0.165, 0.284]

Discussion

This study sought to determine whether it is useful to build a machine learning classifier to detect DeFi projects engaging in various types of securities violations from their tokens' smart contract code. Governments are currently struggling to manage fraud in the DeFi ecosystem, particularly because these platforms do not require "Know Your Client" (KYC) information; hence, a classifier could serve as a triage measure. In addition, we created a new dataset with a verified ground truth. Our research is also novel in its use of the ERC-20 token smart contract code to attempt to detect fraud across the Ethereum-based DeFi ecosystem and its deeper analysis of the features of our final model. Finally, our research contributes to the existing body of work on financial markets as well as its practical applications in that predicting and detecting fraud is crucial for financial stability, market integrity, and market efficiency, particularly in the cryptocurrency space (Shams et al. 2021).

Ultimately, we found that DeFi securities violations were detectable. We developed a suitable starting point for this classification problem that performed significantly better than the baseline (80% F-1 score against a baseline of 50%). Our performance was not as high as that of other models for similar classification problems; however, as described below, this may be due to overfitting and the datasets used in other studies. Previously developed baseline models for related classification problems exhibited performance that was more consistent with ours.

Regarding the second research question, further analysis at the feature level indicated that the success of our model may, in part, be related to the state-based attributes of the functions in the SafeMath library.

Comparisons with prior research

Because our study is the first to attempt to classify DeFi securities violations more broadly, we were unable to compare our model's performance with that of previous studies. We do note that other studies that successfully built high-performance classifiers for a related classification problem used more complex methods, which improved on several previous studies that addressed the same classification problem (Chen et al. 2021a). In addition, many previous studies, like ours, used data with imbalanced classes, but they did not always account for this in building their models. Fan et al. (2021) criticized previous studies, including (Jung et al. 2019), on this basis. This may have caused the high-performance metrics reported by other studies to be slightly misleading. For example, only 3.6% of smart contracts in the (Chen et al. 2018) dataset are Ponzi schemes.

In contrast to previous studies, our study considered whether machine learning classification is necessary for this task or whether a simpler model may suffice to solve the same problem. Previous studies have found that models built using logistic regression, for example, are much less effective than more complex models (Xia et al. 2021). We also found that this is true.

Chen et al. (2021a) noted the overall lack of interpretability of the results of classifiers with code-based features. The opcodes themselves have no obvious interpretation with respect to illegal activity, but, equally, there are no opcodes that

offer a straightforward interpretation-i.e., there is no "STEAL" opcode or similar. The opcodes whose frequencies constituted the features of our final model were CALLDATASIZE, LT, CALLVALUE, SWAP3, EXP, CALLER, SHR, NUMBER, PUSH5, and ADDRESS. The only method to draw definitive conclusions from opcode-based features is to dissect the Solidity code from which they were compiled. Although it would have been impractical to dissect all 2,186 smart contracts in our dataset, we gleaned some insights about our three most important features from a selected subset. Specifically, we observed that developers of violating tokens may implement the SafeMath library differently in their code. In particular, they appear to use the "constant" modifier when describing how arithmetic operation overflows should be handled, which offers weaker assurances about the lack of state modification by these functions. It is, therefore, intuitive that the use of the "pure" function would be associated with the legitimate tokens in this case. However, we note that our analysis of transactions deploying the compiled Solidity code for certain contracts does not capture all opcodes whose frequencies were among the top 10 most important features in our model. Future research could also utilize other frameworks for analyzing token transfer behavior from token bytecode, such as TokenAware (He et al. 2023). This would be particularly useful in instances in which contract Solidity code is not publicly available.

The use of code-based features

As in previous studies (Jung et al. 2019; Chen et al. 2021a), we emphasized the usefulness of our classifier immediately upon the deployment of the smart contract to the Ethereum blockchain, regardless of how many wallets interact with it. This is one of the key advantages of using only code-based features for classification, rather than transaction- or account-based features. This makes such a model useful not only as a retroactive tool for investigators, but also for preventing future fraud. This also enables investigators to monitor projects that may engage in future securities violations. Because investigations and prosecutions take a long time (up to several years for complex cases), it is important for prosecutors to be able to gather evidence as early as possible. However, we acknowledge that code-based analysis is merely one of many techniques and future research could explore alternatives.

Potential applications of our model

Our results highlight the importance of exploring the use of computational triage systems in the enforcement process. This is particularly important given that U.S. enforcement agencies seem to rely heavily on submissions to their whistleblower programs (Commodity Futures Trading Commission 2019), so a computational model could reduce reliance on whistleblowers and avoid the government needing to pay out a portion of the funds successfully recovered to whistleblowers (which can be millions of dollars (U.S. Securities and Exchange Commission 2021a)).

A classifier of the type we examined here would be more useful as a triage measure rather than as a source of evidence because of issues surrounding the admissibility of machine learning-generated evidence in U.S. courts and the risk of misclassification. There may be questions about its admissibility under the Fifth Amendment, Sixth

Amendment, and Federal Rules of Evidence; however, legal scholars do not ultimately consider these as impediments to its admission (Nutter 2018).[25] However, even if this is admissible, questions remain about its weight in court. In particular, explaining such evidence to a judge and jury (especially the "black box" calculations involved in developing machine learning models) may lead to it being discounted. There is variation in the levels of trust among jurors in machines in general, and jurors must further trust expert testimony, which explains machine learning tools (Nutter 2018). This is exacerbated by the need for prosecutors to explain complex concepts related to cryptocurrencies in these cases. A machine learning-based tool would likely lead investigators to more compelling transaction-based or qualitative evidence (e.g., marketing material), which can be more easily understood by a judge and jury, and has been effective in prosecuting cryptocurrency-based financial offenses in the past (see *United States v. Constanzo* (2018), *United States v. Murgio* (2016)).

Considering the ambiguity around the Hinman standard[26] in determining whether a project is sufficiently decentralized to avoid being classified as a security, a machine learning model could also serve as an additional tool in developers' arsenal for determination thereof. Finally, a machine learning model may be useful for people interested in participating in the DeFi ecosystem to research the validity of new projects to help protect themselves from fraud.

Limitations and future research

Our study has several limitations. The first is the potential to overfit the model, particularly in the case of imbalanced data (Fan et al. 2021). Because we do not have a separate dataset with known labels on which to test our model, overfitting could remain an issue despite our mitigation efforts. Ultimately, we chose a verified ground truth dataset that was significantly smaller than our set of legitimate tokens. We do acknowledge that our dataset may make this classification problem simpler than it is in reality. We chose a list of generally reputable projects for our legitimate token set and those subject to government enforcement action for securities violations. Given the experimental nature of DeFi (and the high risk appetite of its participants), and the initial inclusion of a few violating tokens in the legitimate set, this set is likely to capture projects that exist in the middle, rather than at only extremes of offending and non-offending, which may be harder to classify. However, it would still be useful for future research to develop more datasets of DeFi securities violations to further test and refine our models using more advanced sequential machine learning techniques. As the number of verified violations increases, future research could also explore DeFi securities violations using more granular classes.

[25] Though a complete discussion of the admissibility of machine learning evidence is outside of the scope of this paper, we provide a brief introduction here. The Fifth Amendment relates to an individual's right to due process (this could arise in the context of the "black box" of machine learning calculations) and the Sixth Amendment includes the Confrontation Clause. This "black box" not only refers to inexplicable machine learning algorithms but also lay people's lack of understanding about how these algorithms work. The Confrontation Clause requires experts to testify in person and submit to cross-examination. However, this is unlikely to be an issue, as the testimony of a machine learning expert should be satisfactory. The Federal Rules of Evidence around relevance, prejudice, and authenticity may be pertinent as well. Lawyers must further prove the accuracy of the evidence (Grossman 2021). Some argue that under Rule 702 and *Daubert v. Merrell Dow Pharmaceuticals*, machine learning evidence is admissible as expert testimony. Through *Daubert*, the court developed four considerations for evaluating expert testimony (Nutter 2018).

[26] The "Hinman standard" refers to William Hinman's 2018 speech which considered the level of decentralization of a project critical to determining whether it should be classed as a security (Blockchain Association 2019).

These classes could show different patterns for various types of securities violations and further aid in prevention and detection efforts. It would also be useful for future research to compare the use of code-based features with models using account-based features, or a combination thereof.

Given the seeming importance of the implementation of the SafeMath library, in part, to our classifier's performance, it may be the case that this classifier is less effective in classifying tokens created with Solidity versions 0.8.0 or later. However, this code does not account for all the most important features. Future research should examine this issue using an expanded dataset.

There are some limitations around the use of a classifier like this in practice. Our future research will specifically explore how to use such a classifier to investigate and build a viable legal case. This will involve performing manual, in-depth analyses on flagged tokens when applying our classifier to other datasets and interactions with their smart contracts (similar to Xia et al.'s (2021) work on Uniswap scam tokens).

Chen et al. (2021a) raised the issue of bad actors using adversarial obfuscation methods to trick classifiers like the one we propose, and Li et al. (2022) acknowledged this risk in the broader context of anti-fraud measures. We did not explicitly account for this possibility in building or model, nor did we test our model against known obfuscation techniques. This could be a useful avenue for future studies.

Further analysis of violating contracts, for example, using methods for analyzing token operational behavior from bytecode (such as TokenAware, which has successfully been applied to discrete instances of fraud (He et al. 2023)) would be fruitful. Analysis of smart contract code using the methods we proposed in this study, on a much larger scale, could also prove useful. Manually analyzing the token smart contracts using Remix added important insights to this study in terms of interpreting the results of our machine learning models. There is value in extending this further, including in other studies on machine learning-based Ethereum fraud detection. We encourage other scholars to pursue this line of research using our publicly available dataset.

Finally, the jurisdictional focus of our research was limited in scope due to the nature of our dataset. Legislation related to cryptocurrencies is changing frequently. In particular, the European Union has recently passed the Regulation on Markets in Cryptoassets (MiCA)(Scicluna and Debono 2023)[27]. A global approach incorporating legislation from multiple jurisdictions is a potential future goal. However, this would bring challenges because the legality of particular applications—equivalent, in technical terms, to the labels of training data—may vary across jurisdictions and change over time. Overcoming this challenge may require the adaptation of approaches from other domains.

One further prospect is that the output of a classification system such as this may be useful in detecting flaws or risks in novel DeFi functions. While the model would be trained on violations that had previously been detected, it is possible that cases may be identified as risky even if they do not correspond to known flaws, but rather because they have underlying similarities to existing cases. In such situations, the manual inspection of cases identified as potentially fraudulent may offer insights into new forms of offending. This type of work contributes to preventing DeFi fraud and the associated

[27] Currently, the only implemented EU regulations that apply to cryptocurrencies relate to money laundering. The EU's securities and investment regulations do not currently apply (Kolinska 2022).

losses to individuals. Research on the prevention of such offenses is a crucial complement to the detection work presented in this study.

Conclusions

Our final model achieves good performance (80% F-1 score against a baseline of 50%) in classifying DeFi-based securities violations based on ten features from the projects' tokens' smart contract code: the frequencies with which the CALLDATASIZE, LT, CALLVALUE, SWAP3, EXP, CALLER, SHR, NUMBER, PUSH5, and ADDRESS opcodes occurred in the contract. We achieved a higher performance with this random forest model than with logistic regression models, leading us to conclude that this classification problem is well suited to machine learning. Our study is novel because it provides a deeper analysis of the opcode-based features responsible for the performance of our classifier. Although this does not account for all the features, the implementation of the SafeMath library in token smart contracts appears to play a role. Despite the apparent influence of this aspect of the token smart contract code on our classifier, the cosine similarity analyses did not suggest that the overall code reuse was the primary reason for its performance. Overall, a computational model like ours would be highly useful for investigators as a triage tool but could be circumvented by nefarious developers in the future. Therefore, it is important to augment the model as further DeFi projects engaging in securities violations are revealed. This study constitutes the first classifier of securities violations in the emerging and fast-growing Ethereum-based DeFi ecosystem and is a useful first step in tackling the documented problem of DeFi fraud. Our work also contributes a novel dataset of DeFi securities violations with a verified ground truth and connects the use of such a classifier with a wider legal context, including how law enforcement can use it from the detection to prosecution stages of a case.

Abbreviations

DeFi	Decentralized finance
dApps	Decentralized applications
Opcode	Operational code
EVM	Ethereum virtual machine
Dexes	Decentralized exchanges
KYC	Know your client
SEC	Securities and exchange commission
FINRA	Financial industry regulatory authority
ICO	Initial coin offering
ERC-20	Ethereum request for comment
BA	Blockchain association
SMOTE	Synthetic minority over-sampling technique

Acknowledgements

The authors also thank Antonis Papasavva and Antoine Vendeville for their contributions to our code.

Author contributions

AT: conceptualization, data collection, analysis, interpretation, and drafting the final manuscript. BK and TD: conceptualization, study design, and feedback on the manuscript. All authors have reviewed the final manuscript.

Funding

This work was funded by the UK EPSRC grant EP/S022503/1 that supports the Centre for Doctoral Training in Cybersecurity at UCL.

Availability of data materials

The datasets generated and analysed during the current study are available in the OSF repository, Detecting DeFi Securities Violations https://osf.io/xcdz6/?view_only=5a61a06ae9154493b67b24fa4979eddb.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 27 April 2022 Accepted: 5 December 2023
Published online: 20 February 2024

References

Aljofey A, Rasool A, Jiang Q, Qu Q (2022) A feature-based robust method for abnormal contracts detection in ethereum blockchain. Electronics 11(18):2937. https://doi.org/10.3390/electronics11182937
Bartoletti M, Carta S, Cimoli T, Saia S (2020a) Dissecting ponzi schemes on ethereum: identification, analysis, and impact. Future Gener Comput Syst 102:259–277. https://doi.org/10.1016/j.future.2019.08.014
Bartoletti M, Chiang JH-y, Lluch-Lafuente A (2020b) SoK: lending pools in decentralized finance. arxiv:2012.13230. Accessed 22 Mar 2022
Bartoletti M, Carta S, Cimoli T, Saia S (2019) Dissecting Ponzi schemes on Ethereum
Binance Academy: (2021) Blockchain. https://academy.binance.com/en/glossary/blockchain Accessed 25 Nov
Binance Academy (2021) Blockchain use cases: prediction markets 29 April. https://academy.binance.com/en/articles/blockchain-use-cases-prediction-markets Accessed 25 Oct 2021
BitcoinWiki (2021) ERC20 Token Standard—Ethereum Smart Contracts—BitcoinWiki 3 Feb. https://en.bitcoinwiki.org/wiki/ERC20 Accessed 18 Nov 2021
Blagus R, Lusa L (2015) Joint use of over- and under-sampling techniques and cross-validation for the development and assessment of prediction models. BMC Bioinformat 16(1):363. https://doi.org/10.1186/s12859-015-0784-9
Blockchain Association (2019) Understanding the SEC's Guidance on Digital Tokens: the hinman token standard. https://blockchainassoc.medium.com/understanding-the-secs-guidance-on-digital-tokens-the-hinman-token-standard-dd51c6105e2a Accessed 10 Jan 2019
Cai W, Wang Z, Ernst JB, Hong Z, Feng C, Leung VCM (2018) Decentralized applications: the blockchain-empowered software system. IEEE Access 6:53019–53033. https://doi.org/10.1109/ACCESS.2018.2870644
Chainalysis (2022) The 2022 Crypto crime report. Technical report February
Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intel Res 16:321–357. https://doi.org/10.1613/jair.953
Chen W, Li X, Sui Y, He N, Wang H, Wu L, Luo X (2021a) Sadponzi: Detecting and characterizing ponzi schemes in ethereum smart contracts. Proc ACM Meas Anal Comput Syst 5(2):26–12630. https://doi.org/10.1145/3460093
Chen W, Zheng Z, Ngai EC-H, Zheng P, Zhou Y (2019) Exploiting blockchain data to detect smart ponzi schemes on ethereum. IEEE Access 7:37575–37586
Chen W, Zheng Z, Cui J, Ngai E, Zheng P, Zhou Y (2018) Detecting ponzi schemes on ethereum: towards healthier blockchain technology. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web—WWW '18, 1409–1418. https://doi.org/10.1145/3178876.3186046
Chen L, Fan Y, Ye Y (2021b) Adversarial reprogramming of pretrained neural networks for fraud detection. CIKM '21, 2935–2939. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3459637.3482053
Chen W, Guo X, Chen Z, Zheng Z, Lu Y, Li Y (2020) Honeypot contract risk warning on ethereum smart contracts, 1–8. IEEE
Choi RY, Coyner AS, Kalpathy-Cramer J, Chiang MF, Campbell JP (2020) Introduction to machine learning, neural networks, and deep learning. Transl Vis Sci Technol 9(2):14. https://doi.org/10.1167/tvst.9.2.14
CipherTrace: (2021) Cryptocurrency Crime and Anti-Money Laundering Report, August (2021). https://ciphertrace.com/cryptocurrency-crime-and-anti-money-laundering-report-august-2021/
Coinbase (2021) Around the Block #14: DeFi insurance 13 May. https://blog.coinbase.com/around-the-block-14-defi-insurance-ebf8e278da13 Accessed 25 Oct 2021
Commodity futures trading commission (2019) CFTC Whistleblower Alert: Be on the Lookout for Virtual Currency Fraud May. https://www.whistleblower.gov/whistleblower-alerts/Virtual_Currency_WBO_Alert.htm Accessed 22 Nov 2021
Crytic: (2021) Ethereum VM (EVM) Opcodes and instruction reference. https://github.com/crytic/evm-opcodes Accessed 17 Nov 2021
Crytic (2020) Pyevmasm. Crytic
DeFi Prime (2021) DeFi and Open Finance. https://defiprime.com/ Accessed 25 Oct 2021
Ethereum: (2021) Ethereum Wallets 23 October. https://ethereum.org/en/wallets/ Accessed 25 Oct 2021
Eversheds Sutherland Ltd. (2018) Navigating the issues securities enforcement global update. Report https://us.eversheds-sutherland.com/mobile/portalresource/lookup/poid/Z1tOl9NPluKPtDNlqLMRV56Pab6TfzcRXncKbDtRr9tObDdEpW3CmS3!/fileUpload.name=/Securities-Enforcement-Global-Update_Fall-2018.pdf
FBI (2021) Securities fraud awareness & prevention tips. https://www.fbi.gov/stats-services/publications/securities-fraud Accessed 18 Nov
Fan S, Fu S, Xu H, Cheng X (2021) Al-spsd: anti-leakage smart ponzi schemes detection in blockchain. Inf Process Manag 58(4):102587. https://doi.org/10.1016/j.ipm.2021.102587
Fan S, Fu S, Luo Y, Xu H, Zhang X, Xu M (2022) Smart Contract Scams Detection with Topological Data Analysis on Account Interaction. In: Proceedings of the 31st ACM international conference on information & knowledge management. CIKM '22, pp. 468–477. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3511808.3557454. Accessed 2022-11-21

Fang F, Ventre C, Basios M, Kanthan L, Martinez-Rego D, Wu F, Li L (2022) Cryptocurrency trading: a comprehensive survey. Financ Innov 8(1):13. https://doi.org/10.1186/s40854-021-00321-6
Ferreira Torres C, Jonker H, State R (2022) Elysium: context-aware bytecode-level patching to automatically heal vulnerable smart contracts. In: 25th international symposium on research in attacks, intrusions and defenses, pp. 115–128. ACM, Limassol Cyprus. https://doi.org/10.1145/3545948.3545975. https://dl.acm.org/doi/10.1145/3545948.3545975 Accessed 2022-11-21
Gapusan, J (2021) DeFi: Who will build the future of finance? (2021). https://www.forbes.com/sites/jeffgapusan/2021/11/02/defi-who-will-build-the-future-of-finance/ Accessed 18 Nov 2021
Grossman PG (2021) Maura: artificial intelligence as evidence. In: Maryland State Bar Association Young Lawyer's Section 25 August
Han J, Kamber M, Pei J (2012) Getting to know your data. In: Han J, Kamber M, Pei J (eds) Data mining, 3 edn. The Morgan Kaufmann series in data management systems, pp. 39–82. Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-381479-1.00002-2. https://www.sciencedirect.com/science/article/pii/B9780123814791000022 Accessed 19 April 2022
Hastie T, Qian J, Tay K (2023) An introduction to glmnet. https://glmnet.stanford.edu/articles/glmnet.html#introduction Accessed 10 May 2023
He Z, Song S, Bai Y, Luo X, Chen T, Zhang W, He P, Li H, Lin X, Zhang X (2023) Tokenaware: accurate and efficient bookkeeping recognition for token smart contracts. ACM Trans Softw Eng Methodol. https://doi.org/10.1145/3560263
He N. Wu L, Wang H, Guo Y, Jiang X (2019) Characterizing code clones in the ethereum smart contract ecosystem. arXiv:1905.00272 [cs]
Hertig, A (2020) What Is DeFi? 18 September. https://www.coindesk.com/learn/what-is-defi/ Accessed 25 Oct 2021
Hu T, Liu X, Chen T, Zhang X, Huang X, Niu W, Lu J, Zhou K, Liu Y (2021) Transaction-based classification and detection approach for ethereum smart contract. Inf Process Manag 58(2):102462. https://doi.org/10.1016/j.ipm.2020.102462
Hu H, Xu Y (2021) Scsguard: deep scam detection for ethereum smart contracts. arXiv:2105.10426 [cs]
Ibrahim RF, Mohammad Elian A, Ababneh M (2021) Illicit account detection in the ethereum blockchain using machine learning. In: 2021 International Conference on Information Technology (ICIT), pp 488–493. https://doi.org/10.1109/ICIT52682.2021.9491653
Jagati S (2021) DeFi lending and borrowing, explained 18 January. https://cointelegraph.com/explained/defi-lending-and-borrowing-explained Accessed 25 Oct 2021
Jung E, Le Tilly M, Gehani A, Ge Y (2019) Data mining-based ethereum fraud detection. IEEE, 266–273
Kamps J, Trozze A, Kleinberg B (2022) forthcoming. In: Wood, S., Hanoch, Y. (eds.) Cryptocurrency Fraud. A fresh look at fraud: theoretical and applied approaches. Routledge, forthcoming
Karimov B, Wójcik P (2021) Identification of scams in initial coin offerings with machine learning. Front Artif Intel 4:718450. https://doi.org/10.3389/frai.2021.718450
Kolinska D (2022) Cryptocurrencies in the EU: new rules to boost benefits and curb threats https://www.europarl.europa.eu/news/en/press-room/20220309IPR25162/cryptocurrencies-in-the-eu-new-rules-to-boost-benefits-and-curb-threats Accessed 22 Aug 2022
Lašas K, Kasputytė G, Užupytė R, Krilavičius T (2020) Fraudulent behaviour identification in ethereum blockchain
Li T, Kou G, Peng Y, Yu PS (2022) An integrated cluster detection, optimization, and interpretation approach for financial data. IEEE Trans Cybern 52(12):13848–13861. https://doi.org/10.1109/TCYB.2021.3109066
Li J, Baldimtsi F, Brandao JP, Kugler M, Hulays R, Showers E, Ali Z, Chang J (2021) Measuring illicit activity in defi: The case of ethereum. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp 197–203. https://doi.org/10.1007/978-3-662-63958-0_18
Liu L, Tsai W-T, Bhuiyan MZA, Peng H, Liu M (2022) Blockchain-enabled fraud discovery through abnormal smart contract detection on ethereum. Futur Gener Comput Syst 128:158–166. https://doi.org/10.1016/j.future.2021.08.023
Meta Research (2023) fastText. original-date: 2016-07-16T13:38:42Z https://github.com/facebookresearch/fastText Accessed 28 April 2023
Modi R (2018) Solidity Programming Essentials. Packt, https://subscription.packtpub.com/book/application- development/9781788831383/7/ch07lvl1sec81/the-view-constant-and-pure-functions Accessed 09 May 2023
Musiala RAJ, Goody TM, Reynolds V, Tenery L, McGrath M, Rowland C, Sekhri S (2020) Cryptocurrencies: Forensic techniques to meet the challenge of new fraud and corruption risks | FVS Eye on Fraud. Report, AICPA Winter https://future.aicpa.org/resources/download/cryptocurrencies-forensic-techniques-to-face-new-fraud-and-corruption-risks
Nabi T (2022) Pure vs view in solidity. https://hashnode.com/post/pure-vs-view-in-solidity-cl04tbzlh07kaudnv1ial1gio Accessed 09 May 2023
Narayanan A, Bonneau J, Felten E, Miller A, Goldfeder S (2016) Bitcoin and cryptocurrency technologies
Nutter PW (2018) Machine learning evidence: admissibility and weight comments. Univ Pa J Const Law 21(3):919–958
OpenZeppelin (2023) Contracts. https://docs.openzeppelin.com/contracts/2.x/ Accessed 09 May 2023
OpenZeppelin (2023) Math. https://docs.openzeppelin.com/contracts/2.x/api/math Accessed 15 May 2023
Perkis T (1994) Stack-based genetic programming. In: Proceedings of the First IEEE conference on evolutionary computation. IEEE world congress on computational intelligence, pp. 148–1531. https://doi.org/10.1109/ICEC.1994.350025
Podgor ES (2019) Cryptocurrencies and securities fraud: In need of legal guidance. Available at SSRN 3413384
Practical Law Corporate & Securities (2021) US securities laws: overview. Practice Note 3-383-6798, Thomson Reuters
Prellberg J, Kramer O (2020) Acute lymphoblastic leukemia classification from microscopic images using convolutional neural networks. arXiv:1906.09020 [cs]
Remix (2022a) Debugger. https://remix-ide.readthedocs.io/en/latest/debugger.html Accessed 15 May 2023
Remix (2022b) Debugging transactions. https://remix-ide.readthedocs.io/en/latest/tutorial_debug.html Accessed 15 May 2023

Rodler M, Li W, Karame GO, Davi L (2021) EVMPatch: Timely and automated patching of ethereum smart contracts. In: 30th usenix security symposium (USENIX Security 21), pp. 1289–1306. USENIX Association. https://www.usenix.org/conference/usenixsecurity21/presentation/rodler
Santos I, Brezo F, Ugarte-Pedrero X, Bringas PG (2013) Opcode sequences as representation of executables for data-mining-based unknown malware detection. Inf Sci Int J 231:64–82. https://doi.org/10.1016/j.ins.2011.08.020
Schär F (2021) Decentralized finance: on blockchain- and smart contract-based financial markets. https://doi.org/10.20955/r.103.153-74. Accessed 22 Mar 2022
Scicluna MC, Debono J (2023) MiCA: landmark crypto regulation approved by EU Parliament. https://www.lexology.com/library/detail.aspx?g=152d8020-bc6e-47b9-b236-5b1f8c6b2b88 Accessed 15 May 2023
Scikit-learn developers (2023) sklearn.preprocessing.StandardScaler. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html Accessed 10 May 2023
Sebastião H, Godinho P (2021) Forecasting and trading cryptocurrencies with machine learning under changing market conditions. Financ Innov 7(1):3. https://doi.org/10.1186/s40854-020-00217-x
Secret network (2021) About Secret (SCRT). https://scrt.network/ Accessed 22 Nov 2021
Securities and exchange commission v. AriseBank, Jared Rice Sr., and Stanley Ford (2020). N.D. Tex. 23 January. No. 3-18-cv-0186-M
Securities and exchange commission v. PlexCorps, Dominic LaCroix, and Sabrina Paradis-Royer (2019). E.D.N.Y. 2 October. No. 17-cv-7007 (CBA) (RML)
Securities and exchange commission v. REcoin Group Foundation, LLC, DRC World INC. A/k/a Diamond Reserve Club, and Maksim Zaslavskiy (2018). E.D.N.Y. 14 May. No. 17-cv-05725
Securities and exchange commission v. LBRY (2022). D.N.H. November 7. No. 21-CV-260-PB
Securities and exchange commission v. Natural Diamonds Investment Co., Eagle Financial Diamond Group Inc A/k/a Diamante Atelier, Argyle Coin, LLC, Jose Angel Aman, Harold Seigel, and Jonathan H. Seigel (2019a). S.D. Fla. 11 December. No. 19-cv-80633
Securities and exchange commission v. Reginald Middleton, et al. (2019b). E.D.N.Y. 1 November. No. 19-cv-4625
Shams SMR, Sobhan A, Vrontis D (2021) Detection of financial fraud risk: implications for financial stability. J Oper Risk
Solidity Team (2020) Solidity 0.8.0 release announcement. https://blog.soliditylang.org/2020/12/16/solidity-v0.8.0-release-announcement/ Accessed 2023 May 09
Solidity Dev Studio: (2020) Exploring the new Solidity 0.8 Release. https://soliditydeveloper.com/solidity-0.8 Accessed 09 May 2023
The Solidity Authors (2023) Contracts. https://docs.soliditylang.org/en/v0.8.19/contracts.html#view-functions Accessed 09 May 2023
Trozze A, Kamps J, Akartuna EA, Hetzel F, Kleinberg B, Davies T, Johnson S (2022) Cryptocurrencies and future financial crime. Crime Science
U.S. Securities and Exchange Commission (2019) Framework for investment contract analysis of digital assets. https://www.sec.gov/corpfin/framework-investment-contract-analysis-digital-assets Accessed 2023 Feb 20
U.S. Securities and Exchange Commission (2021a) SEC Awards $22 Million to Two Whistleblowers. https://www.sec.gov/news/press-release/2021-81 Accessed 22 Nov 2021
U.S. Securities and Exchange Commission: (2021b) Annual Report to Congress Whistleblower Program. https://www.sec.gov/files/2021_OW_AR_508.pdf Accessed 2022 Apr 04
U.S. Securities and Exchange Commission (2022) Cyber enforcement actions 19 January. https://www.sec.gov/spotlight/cybersecurity-enforcement-actions Accessed 10 Feb 2022
Uniswap: (2021) Uniswap Governance. https://gov.uniswap.org/ Accessed 22 November 2021
Uniswap: (2020) Introducing Token Lists 26 August. https://uniswap.org/blog/token-lists Accessed 18 Nov 2021
United States v. Costanzo (2018) D. Arizona 10 August. No. 2:17-cr-00585-GMS
United States v. Murgio (2016) S.D.N.Y. 19 September. No. 15-cr-769 (AJN)
Wang L, Cheng H, Zheng Z, Yang A, Zhu X (2021) Ponzi scheme detection via oversampling-based long short-term memory for smart contracts. Knowl-Based Syst 228:107312. https://doi.org/10.1016/j.knosys.2021.107312
Wang L, Sarker PK, Bouri E (2022) Short- and long-term interactions between bitcoin and economic variables: evidence from the US. Comput Econ. https://doi.org/10.1007/s10614-022-10247-5
web3.py. ethereum (2023) original-date: 2016-04-14T15:59:35Z. https://github.com/ethereum/web3.py/blob/acd5b24474dd5b13548dffa33e1d2872c3dccad9/docs/index.rst Accessed 28 April 2023
Wilder RP (2020) Heidi: Tracing cryptocurrency scams: Clustering replicated advance-fee and phishing websites. arXiv preprint arXiv:2005.14440
Wintermeyer L (2021) After Growing 88x In A Year, Where Does DeFi Go From Here? (2 November 2021). https://www.forbes.com/sites/lawrencewintermeyer/2021/05/20/after-growing-88x-in-a-year-where-does-defi-go-from-here/ Accessed 18 Nov
Wood, G (2021) Ethereum: a secure decentralised generalised transaction ledger. Ethereum 2 Nov. https://ethereum.github.io/yellowpaper/paper.pdf
Wu J, Lin D, Zheng Z, Yuan Q (2020) T-edge: Temporal weighted multidigraph embedding for ethereum transaction network analysis. Front Phys 8:204. https://doi.org/10.3389/fphy.2020.00204
Xia P, wang H, Gao B, Su W, Yu Z, Luo X, Zhang C, Xiao X, Xu G (2021) Demystifying scam tokens on uniswap decentralized exchange. arXiv:2109.00229 [cs]
Xin Q, Zhou J, Hu F (2018) The economic consequences of financial fraud: evidence from the product market in China. China J Account Stud 6(1):1–23. https://doi.org/10.1080/21697213.2018.1480005
Xu M, Chen X, Kou G (2019) A systematic review of blockchain. Financ Innov 5(1):27. https://doi.org/10.1186/s40854-019-0147-z
Xu J, Paruch K, Cousaert S, Feng Y (2021) SoK: Decentralized exchanges (DEX) with automated market maker (AMM) protocols. arxiv:2103.12732. Accessed 22 Mar 2022
Zapper: (2021) Your Homepage to DeFi. https://zapper.fi/ Accessed 18 Nov 2021

Zes (2020) Is it safe to Zap into all liquidity pools on Zapper?. https://zapper.crunch.help/zapper-fi-faq/is-it-safe-to-zap-into-all-liquidity-pools-on-zapper Accessed 18 Nov 2021
Zhang Y, Yu W, Li Z, Raza S, Cao H (2021) Detecting ethereum ponzi schemes based on improved lightgbm algorithm. IEEE Trans Comput Soc Syst. https://doi.org/10.1109/TCSS.2021.3088145
Zhou L, Xiong X, Ernstberger J, Chaliasos S, Wang Z, Wang Y, Qin K, Wattenhofer R, Song D, Gervais A (2023) SoK: decentralized finance (DeFi) attacks. arxiv:2208.13035

Publisher's Note