DEPARTMENT OF COMPUTER SCIENCE

Exposing Rug Pulls with Artificial Intelligence in Cryptocurrency Markets

[redacted]

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Engineering in the Faculty of Engineering.

Wednesday 3rd May, 2023

Abstract

The popularity of cryptocurrencies is rising due to their transparent and decentralized nature, offering an alternative to traditional financial systems. However, unregulated cryptocurrency markets have led to a rise in fraudulent activities that harm investors and undermine the credibility of blockchain technology. To prevent cryptocurrency rug pulls and ensure the continued growth of decentralized finance, it is essential to develop effective methods for detecting potential scams and protecting investors.

While there are existing tools available such as TokenSniffer[22] and RugPullDetector[19] that can aid in the detection of fraudulent cryptocurrency tokens, they often rely on simple heuristics and are limited in their accuracy. To combat the increasing sophistication of scams, it is crucial to stay ahead of the curve and leverage advanced technologies such as artificial intelligence.

Our study is aimed at expanding the current cryptocurrency dataset to include 24 new features, which will help us create more accurate and effective models for detecting cryptocurrency rug pulls. Our primary goal is to evaluate the effectiveness of different machine learning algorithms in identifying non-legitimate tokens while showcasing how neural networks can be just as effective for this use case. To ensure our model can perform well in real-world scenarios, we conducted a sensitivity analysis to test its robustness against varying class distributions. We also categorized different subsets of features to identify areas where further research could be conducted and analyzed the importance of the features provided by the models.

Our work provides a valuable tool for investors and regulators, such as government agencies, to detect cryptocurrency rug pulls. Furthermore, the enhanced dataset resulting from this study is expected to bring significant benefits to future research in the cryptocurrency markets.

Contents

List of Figures

List of Tables

Ethics Statement

This project did not require ethical review, as determined by my supervisor, Matthew Edwards.

Supporting Technologies

- I used **requests** to make HTTP requests to APIs.
- I used **json** for storing the data obtained from the APIs.
- I used **pandas** and **numpy** for reading, manipulating, and analyzing data.
- I used **keras** for building, training, and evaluating neural networks.
- I used **sklearn** for preprocessing, training various machine learning algorithms, and evaluating them.
- I used **xgboost** for training the XGBoost model.
- I used **optuna** for optimizing the hyperparameters of the models presented.
- I used **matplotlib** for plotting various results in the analysis.
- I used **Jupyter Notebook** for machine learning, data analysis, and visualization.
- I used **imblearn.over_sampling.SMOTE** as an over-sampling technique for the minority class in the dataset.

Notation and Acronyms

AI	:	Artificial Intelligence
BVM	:	Bitcoin Virtual Machine
DeFi	:	Decentralized Finance
EOA	:	Externally owned address
ECSDSA	:	Elliptic Curve Digital Signature
ETH	:	ether
NFT	:	non-fungible token
ERC-20	:	Ethereum Request for Comments
dApp	:	Decentralized application
DEX	:	Decentralized Exchange
AMM	:	Automated Market Maker
SVM	:	Support Vector Machine
XGBoost	:	eXtreme Gradient Boosting
FNN	:	Feedforward Neural Network
API	:	Application Programming Interface
ReLU	:	Rectified Linear Unit
SMOTE	:	Syntethic Minority Over-sampling Technique
CG	:	CoinGecko
RNN	:	Recurrent Neural Network
HHI	:	Herfindahl–Hirschman index

Chapter 1

Introduction

1.1 Project Topic and Problem

Cryptocurrencies are a virtual form of currency that uses cryptography for security and functions independently of traditional financial institutions. They exist on a public digital ledger called the blockhain, which records transactions in a secure and transparent manner. The blockchain technology enables quick, safe, and low-fee transactions without the need for intermediaries, excluding the involvement of centralized systems such as banks. This decentralized and secure nature of cryptocurrencies has led individuals to believe that they have the potential to replace traditional financial institutions[18]. By eliminating the need for intermediaries, cryptocurrencies can reduce transaction fees, increase transparency, and improve security. These are some of the main advantages offered by this technology, which makes us believe that digital currencies will be universally adopted in the next few years[36].

Cryptocurrencies have gained significant popularity in recent years, with an increasing number of investors turning to these digital assets as an alternative to traditional forms of investment. However, the unregulated nature of cryptocurrency markets has made this area attractive for scammers. Various scams took place over the last few years, which caused significant financial losses to the investors. Moreover, this damaged the reputation of the cryptocurrency as a whole.

One particularly common type of cryptocurrency scam is called a rug pull. It is a form of scam where the people behind the project intentionally deceive investors by promising them significant returns on their investment, only to abruptly exit the project with the funds that had been invested. Rug pulls have become increasingly common in the cryptocurrency markets, with some estimates suggesting that they account for a significant portion of all cryptocurrency-related scams.

The complex and evolving nature of rug pulls has made them difficult to detect and prevent. Artificial intelligence techniques are increasing in popularity for fraud detection. These models can analyze a vast amount of data and identify patterns that might not be apparent to a human analyst. As cryptocurrencies continue to grow in popularity, it is likely that AI models will become increasingly important tools for identifying and preventing fraud on the blockchain. By leveraging these advanced technologies, it may be possible to reduce the prevalence of rug pulls and increase investor confidence in the cryptocurrency markets. Moreover, the detection models can contribute and educate people by explaining what specific features influenced the model's classification.

1.2 Significance and Importance

As mentioned above, scams are prevalent on the blockchain. Every scammer has the chance to take advantage of the open nature of this technology and create their own cryptocurrency.

Many investors are non-experts when it comes to understanding the complexities of blockchain technology, and they lose significant amounts of money by investing in scams.

It is essential to ensure trust in order to have stability and continued growth in the crypto markets, which will implicitly accelerate innovation and progress in the industry. Currently, there is a lack of effective methods for detecting rug pulls in the cryptocurrency markets. There are some free tools available, but none of them take advantage of advanced techniques such as machine learning algorithms or artificial neural networks, which are widely adopted in the fraud detection and prevention field.[15].

Developing a platform that incorporates a model that detects crypto rug pulls would be widely adopted by both investors and traders in the cryptocurrency market. The public would be better informed when making a decision to invest. This increases trust in the community, which leads to faster adoption of the blockchain. Centralized exchange platforms can also automate the process of listing new crypto assets on their platform by adding an extra layer of protection to their audit system, protecting their investors from losing their money. Moreover, regulatory agencies can make use of this model in order to identify and prosecute the creators behind these cryptocurrency scams.

1.3 Project Aim and Challenges

The overall goal of this project is to develop a reliable, effective, and efficient classifier to identify potential rug pulls in cryptocurrencies, empowering investors to make informed decisions.
For the scope of this project, my aims can be brought down to:

1. Enhance the existing cryptocurrency dataset from the literature by incorporating novel features, with the aim of boosting the performance of artificial intelligence models
2. Train and evaluate a range of machine learning models and perform sensitivity analysis to assess their robustness.
3. Propose a feedforward neural network architecture and analyze its effectiveness in identifying rug pulls in cryptocurrency markets
4. Analyze the importance of individual features and gain insights into the data

Chapter 2

Background

2.1 Blockchain Technology and Development

The blockchain is a shared, distributed, and immutable ledger that records transactions and tracks digital assets on a peer-to-peer network. Instead of being controlled by an organization or authority, the key feature of blockchain technology is that it can be decentralized and maintained by a network of computers. Each computer has a copy of the ledger, making it a highly resilient system. The list of records, which are also referred to as blocks, are securely linked together via cryptographic algorithms to prevent data tampering[29]. The Bitcoin blockchain was the first decentralized system to use blockchain technology, which demonstrated the possibility of an alternative financial system not controlled by a centralized institution.

In the beginning, the Bitcoin network was designed primarily as a decentralized digital currency system. However, developers soon realized that the Bitcoin Virtual Machine (BVM), which is the component of the Bitcoin network that allows for the execution of decentralized applications, had limited programmability. This meant that developers were unable to create complex decentralized applications on the Bitcoin network.

To address these limitations, a group of developers created Ethereum, a blockchain platform that is Turing-complete, meaning it can support any arbitrary computation or computer program[30]. Ethereum's platform is designed to be more flexible than the BVM, allowing developers to create and execute more advanced decentralized applications. Therefore, the Ethereum blockchain has become a popular platform for the development of decentralized applications and smart contracts. Its network has also spawned a whole ecosystem of decentralized finance (DeFi) applications, such as decentralized exchanges, lending platforms, and other financial tools that run on top of this ecosystem. There are multiple blockchains, such as Binance Smart Chain, Solana, and Polkadot, that are gaining popularity for DeFi applications due to their scalability and lower transaction fees. However, in this paper, we will focus on the most popular blockchain for decentralized applications[23], which is the Ethereum blockchain.

2.2 Accounts & Transactions

2.2.1 Accounts

Accounts can be defined as a unique identifier for an entity on the network. There are two types of addresses:

1. Externally owned addresses (EOAs, or wallet addresses) are a type of account on Ethereum that is controlled by a private key. The private key is generated randomly, and then the Elliptic Curve Digital Signature Algorithm (ECSDSA) public key encryption algorithm[6]

is used for deriving the public key. Finally, the Keccak-256 hash function[13] is applied to the public key to generate the address. It is 40 characters long and is represented in hexadecimal format, which includes the prefix "0x"[49]. An user can have one or multiple addresses and is responsible for keeping the private key secure, which will be used for interacting with the account. However, there are wallet providers that abstract the complexity of private key management, making the interactions with the blockchain more user-friendly.

2. Contract addresses are a type of account on Ethereum that is controlled by code. It uses the same hash function used for generating EOAs and is generated when an EOA deploys a new smart contract (see Section 2.3) on the blockchain. Unlike an EOA, a contract address does not have an associated private key, it is fully controlled by the logic of the code.

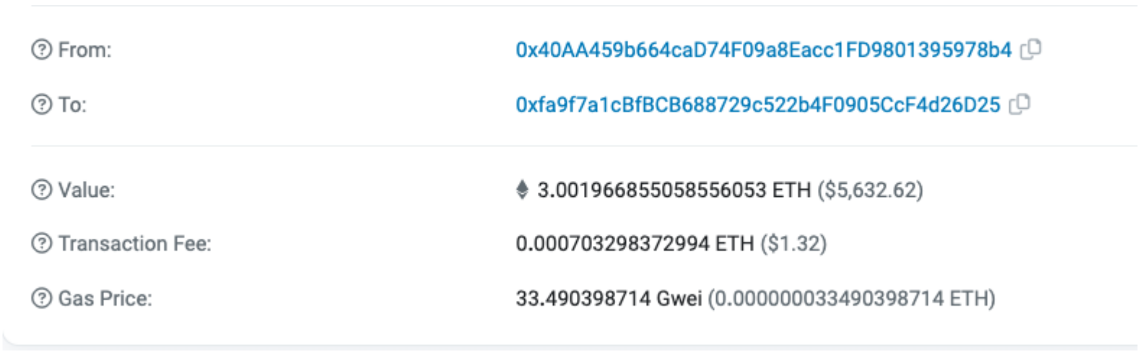
The difference between these two types of addresses is that an EOA can initiate any transaction on the chain, while the contract address can only initiate a transaction in response to receiving other transactions from one or more EOAs.

2.2.2 Transactions

An Ethereum transaction is similar to one in the traditional banking system. An address can send a transaction on the blockchain that represents an object containing data. Within the data, you specify who you are (the sender address), who the receiver is (the recipient address), what asset is being exchanged, and the quantity of the asset[33]. Each transaction has a transaction fee called the gas fee. The gas fee is determined by two factors, which are represented by the complexity of the transaction and how busy the network is at that specific moment. On the Ethereum network, it is paid in its native cryptocurrency, known as ether (ETH).

The simplest kind of transaction in terms of complexity is when a user wants to send an amount of ETH to another user, a transaction from one address to another.

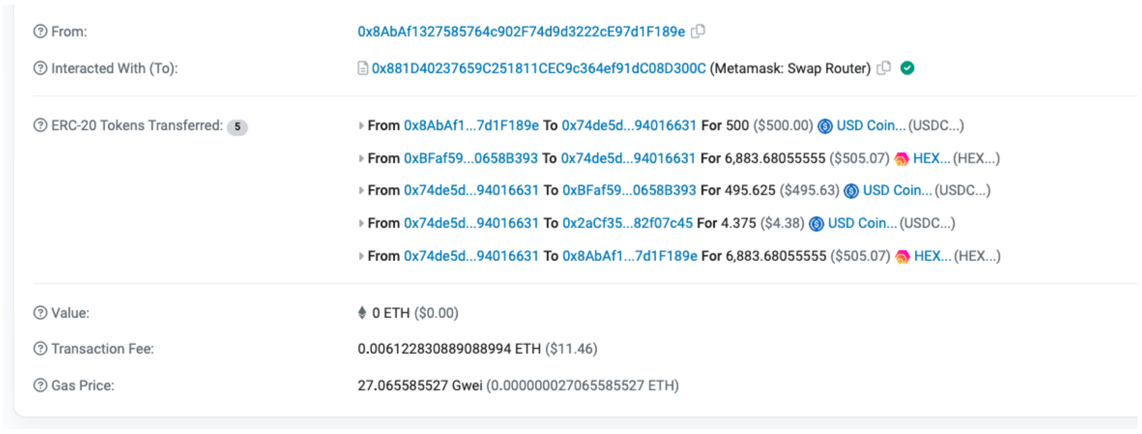
All the transactions are available and transparent on the Ethereum Blockchain Explorer: Etherscan[9]. In Figure 2.1, there is a simple transaction of sending ETH from one EOA to another. The *0x40aa459b664cad74f09a8eacc1fd9801395978b4* address completed a transaction that transferred the amount of 3 ETH to *0xfa9f7a1cbfbcb688729c522b4f0905ccf4d26d25*. It can also be observed that the transaction fee for this was 0.00007 ETH.



From:	0x40AA459b664caD74F09a8Eacc1FD9801395978b4
To:	0xfa9f7a1cBfBCB688729c522b4F0905CcF4d26D25
Value:	3.001966855058556053 ETH ($5,632.62)
Transaction Fee:	0.000703298372994 ETH ($1.32)
Gas Price:	33.490398714 Gwei (0.000000033490398714 ETH)

Figure 2.1: Transfer ETH transaction

Some of the steps for validating a transaction are that the initiator of the transaction has signed it with their private key, which proves the ownership of the address, and then the blockchain verifies it has enough balance to send the corresponding amount and to pay the transaction fee.



From:	0x8AbAf1327585764c902F74d9d3222cE97d1F189e
Interacted With (To):	0x881D40237659C251811CEC9c364ef91dC08D300C (Metamask: Swap Router)
ERC-20 Tokens Transferred: 5	From 0x8AbAf1...7d1F189e To 0x74de5d...94016631 For 500 ($500.00) USD Coin... (USDC...)
	From 0xBFaf59...0658B393 To 0x74de5d...94016631 For 6,883.68055555 ($505.07) HEX... (HEX...)
	From 0x74de5d...94016631 To 0xBFaf59...0658B393 For 495.625 ($495.63) USD Coin... (USDC...)
	From 0x74de5d...94016631 To 0x2aCf35...82f07c45 For 4.375 ($4.38) USD Coin... (USDC...)
	From 0x74de5d...94016631 To 0x8AbAf1...7d1F189e For 6,883.68055555 ($505.07) HEX... (HEX...)
Value:	0 ETH ($0.00)
Transaction Fee:	0.006122830889088994 ETH ($11.46)
Gas Price:	27.065585527 Gwei (0.000000027065585527 ETH)

Figure 2.2: Swap transaction

As mentioned above, one of the factors determining the cost of a transaction is its complexity. In Figure 2.2, there is an interaction with a contract address. In this case, the transaction is between an EOA and the Metamask Router (a smart contract, see section 2.3) which performs multiple swap transactions between different assets. The value of ETH is set to zero because the transactions use an asset called USD Coin. When the EOA interacts with the contract, there are five different transfers happening under the hood, initiated by the logic of the contract. The EOA sends 500 USD Coin and receives 6,883 HEX in exchange. As expected, the transaction fee is significantly higher than in the simpler transaction from Figure 2.1, because it requires multiple transactions to be processed.

2.3 Smart Contracts

Smart contracts are computer programs that operate on top of the blockchain platform. Their design is to enforce an agreement between parties who do not trust each other without the involvement of a trusted third-party[38]. If we compare them to traditional contracts, the smart contract can be explained as an agreement written in code. In a traditional contract scenario, if any clause is broken by one of the parties, there is usually a need for a third party to take action. The main advantage of having a smart contract is that, assuming that there are no vulnerabilities present in the code, it is impossible to break any clauses. They are immutable, like anything else on the blockchain, to prevent any tampering. An EOA is able to deploy a smart contract by sending a transaction of deployment to the blockchain. After this, the EOA address becomes the owner of the contract. After deployment, it is the owner's choice to make the source code available and open to anyone. If they choose not to verify it, the bytecode will be the only thing available to the public. A smart contract will be uniquely identified by its generated contract address. The code implementation typically involves a collection of functions that allow users (addresses) to interact with it. It is common practice for the contract owner to impose certain restrictions on function access. For instance, access to functions may be restricted to owners only or may be subject to a specific area (e.g., you can interact with this function only if you send an amount of ETH bigger than 0.1 through the transaction).

In a smart contract, you can have anything from simple functions that perform mathematical calculations to complex algorithms that manage entire decentralized applications. They refer to a set of guidelines established by the Ethereum network. Some of the most well-known Ethereum smart contract standards are ERC-20 (for fungible tokens), ERC-721 (for non-fungible tokens also known as NFTs), and ERC-1155 (for multi-token contracts).

In this paper, the focus will be on the *ERC-20 Token Standard*. As mentioned previously in Section 2.2.2, on the Ethereum blockchain, the native currency is ether. The ERC-20 standard facilitates the creation of assets on the chain, which can be traded exactly like the native cryptocurrency. The ERC-20 standard consists of a list of specific rules and interfaces that a token should follow. Some of the functionalities that the standard provides are[7]:

1. transfer tokens from one account to another
2. get the current token balance of an account
3. get the total supply of the token available on the network
4. approve whether an amount of token from an account can be spent by a third-party account

Each ERC-20 token can have its own name and ticker, which is a unique symbol of letters similar to what is used for the stock market. Take, for example, the famous token Dogecoin[5], whose ticker is DOGE. The standard does not place any restrictions on the selection of a name or ticker that has not been used before. In Section 2.5, we will discuss in more detail how the lack of restrictions could potentially open up opportunities for scammers to deceive investors.

Many ERC-20 tokens include additional functionalities beyond the standard base functions. Some of these can enhance the user experience, such as atomic transactions, which enable the exchange of tokens for a very small gas fee. However, most of them can potentially give the contract owner or unauthorized parties control over an investor's investment. It is important to be aware of these functionalities and understand how they are implemented, as they can have significant implications for the security and value of the token[41]. A couple of popular examples of these are the following:

1. Mint function: This function allows the contract owner to create additional tokens on demand, allowing for increased flexibility in managing the token supply. However, this could potentially lead to a dilution of the value of existing tokens, affecting all investors.
2. Burn function: This function allows the contract owner to permanently remove tokens from circulation. This would help increase the value of the remaining tokens. However, there is a risk that the contract owner would use it in a malicious way to control the market, increase the price of the token, and sell his own assets.
3. Pause trading: This function allows the contract owner to pause the trading of the token, preventing all transactions from being executed.
4. Freeze address: This function allows the contract owner to forbid any interactions with the contract at a specific address, preventing the investor from getting their investment back.

It is important to mention that cryptocurrencies and token assets, according to their definitions, should not be interchangeable terms. A cryptocurrency typically has its own blockchain network (e.g. ether, bitcoin), while ERC-20 tokens are often used to represent assets or utilities within decentralized applications and can have different use cases. However, both literature and media are used interchangeably when discussing scams, so we will follow a similar approach here.

2.4 Decentralized applications

Decentralized applications (dApps) are built on top of smart contracts. They can provide unique capabilities, such as the ability to create and manage digital tokens, enable secure voting and governance systems, or facilitate complex financial transactions. A good example of a complex dApp is a decentralized marketplace. If we compare this with a traditional marketplace like eBay, this platform acts like a middleman between buyers and sellers, facilitating transactions and taking a commission on each sale. In contrast, on a dApp marketplace, the need for a middleman is eliminated, allowing users to interact and transact directly with each other using cryptocurrency as a form of payment.

2.4.1 Decentralized Exchanges

Our study will primarily center on *Decentralized Exchanges* (DEXs)[27], which are considered to be the foundation of the DeFi ecosystem. The demand in the blockchain ecosystem is to have a decentralized exchange mechanism that does not involve any central authority, especially when it comes to exchanging tokens. Unlike centralized exchanges, a blockchain-based DEX does not store user funds and personal data on centralized servers. Instead, it facilitates the trading of digital assets by matching buyers and sellers through smart contracts[50].

On a DEX, the trades are publicly available on the blockchain. The first generation of DEXs featured an order book system[28], which functions as a list of buy and sell orders, recording the interests of both buyers and sellers. A smart contract is implemented as a matching engine, which uses this book to determine which orders can be executed, matching buyers and sellers based on their specified price and quantity criteria. This type of order book system is commonly used in stock markets to facilitate the trading of stocks. Research has shown that the same mechanism can be effectively applied to the trading of cryptocurrencies[42]. However, the first generation of DEXs with order book systems faced significant challenges in terms of adoption and generating high trading volume. The design imposed high costs on the mechanism of the market maker. A market maker was required to spend gas for each action, such as adding, modifying, or canceling an order from the book. This mechanism bloated the blockchain without necessarily resulting in a trade between two parties[48]. Another problem mentioned was the limited liquidity and the slow, clunky UX of the available platforms. Etherdelta[8] is an example of an order book DEX, founded by Zachary Coburn in 2016, which started as a market for ERC-20 tokens.

The next generation of DEXs are known as *Automated Market Makers* (AMMs). They do not rely on the traditional order book system from the previous generation to facilitate trades. They use a mathematical formula and a pool of funds to enable trades between different cryptocurrencies. Liquidity providers are the key to facilitating trades for AMMs. The liquidity providers must supply two distinct types of tokens, which in turn allow users to trade against the token pool. The value of the pool is determined by a mathematical formula based on the ratio of existing tokens in the pool. The most popular existing DEXs that use the AMM model are Uniswap[26], Sushiswap[21], and Curve[3] and each of them uses different variations for calculating the exchange rate for the tokens existing in a liquidity pool.

Uniswap

In this study, we will be looking at the most relevant decentralized exchange, *Uniswap*. It was launched in November 2018, and a study reported that there are more than 40,000 ERC-20 tokens tradable in the Uniswap Protocol, adding a total value of 7 billion USD[40]. It was created by Hayden Adams, and it supports all the ETH and ERC-20 liquidity pools, enabling swaps between any ERC-20 token. It is being continuously developed, and it has reached its third version of the protocol, called Uniswap V3. For the purpose of this paper, we will narrow our attention to version 2 of the protocol, which is currently the most used version and has a large number of existing liquidity pools available.

Uniswap V2 is an AMM that relies on two key functions to enable decentralized trading of ERC-20 tokens: liquidity pool and a mathematical formula. The liquidity pools are smart contracts that hold two different tokens and determine the price of a token based on the ratio of the two tokens available balances in the pool. The formula is a constant product, which can be expressed as $x * y = k$, where x and y are the balances of the two tokens in the pool, and k is a constant value that stays the same throughout all trades. For example: if a liquidity pool holds 10 ETH and 1000 EXAMPLE, $10 * 1000 = 10,000$, which means that the following expression of equality will always be true: $k = 10,000$. If a user comes and exchanges 111.11 EXAMPLE for ETH, he will receive 1 ETH. The new balance of the liquidity pool would be: 9 ETH, $\approx$ 1111.11 EXAMPLE and the value of k would stay constant.

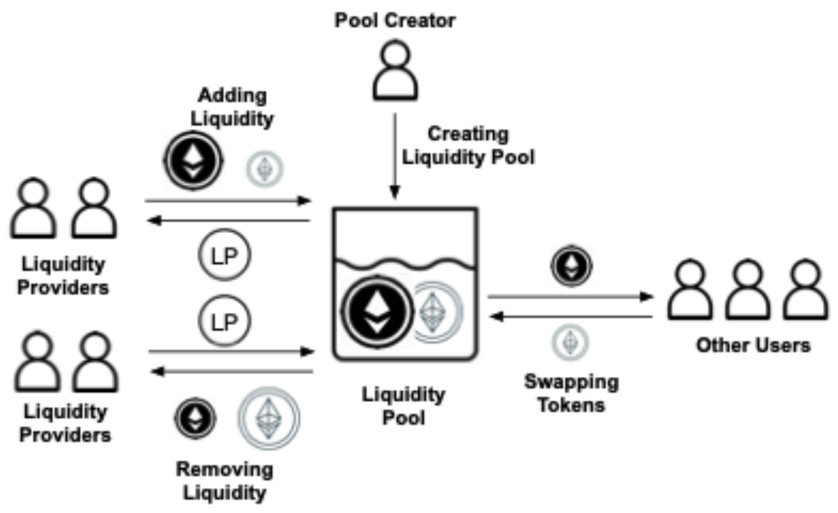


Figure 2.3: Interacting with Uniswap V2 protocol[50]

In Figure 2.3, there is a representation of how the Uniswap V2 protocol works:

1. Pool creator: The pool creator is usually the contract owner of the ERC-20 token, the individual who deployed the smart contract on the blockchain. They have to create a liquidity pool by adding two different cryptocurrencies. Take as an example that the pool creator has a token called EXAMPLE, which he has just created. He wants to add 10 ETH and 1000 EXAMPLE to the pool, but in the case of EXAMPLE, there is no market value for this token because it has not been listed yet. The amount of ETH added to the pool will determine the market value of the EXAMPLE token. They interact with the Uniswap V2 protocol, which automatically creates a liquidity pool that consists of a smart contract. After adding the liquidity, they receive back from Uniswap a liquidity pool token, which can be used as a key for getting the liquidity back whenever they desire.

2. Liquidity Providers: This can be the pool creator or any other user who wants and can contribute to the liquidity. They can take action only after the pool's creation. Going back to our example, the cryptocurrency EXAMPLE already has a market value after pool creation. Therefore, if someone desires to provide more liquidity, they have to deposit an equal value of both tokens (ETH and EXAMPLE). Liquidity providers can then charge a small fee on every trade that occurs within the pool. As mentioned above, they can use the liquidity pool token they received to recover their investment at any time.

3. Other users: They can interact with the liquidity pool just after the pool is created and liquidity is provided. For our example, they can swap ETH for EXAMPLE or EXAMPLE for ETH if there is sufficient liquidity to facilitate the transaction.

The Uniswap V2 protocol tracks the states of the ERC-20 token contract, and events are emitted by a Uniswap V2-related smart contract[25]:

1. PairCreated: This event is triggered every time a new pool is created.

2. Sync: This event is triggered every time the reserves of the balance of the assets in the pool change.

3. Mint, Burn & Transfer: These events are triggered when the state of a liquidity pool token changes. Mint is triggered when a liquidity pool is added, Burn is triggered when a

liquidity pool is removed; and Transfer is triggered when the liquidity pool token has been transferred to a new address.

Liquidity locking is a practice in DeFi that came up as a partial solution to rug pulls. When a newly added token hits the Uniswap market, it is not uncommon for the number of liquidity providers to be relatively low, often with the only liquidity provider being the project owner. This can make potential investors hesitant, fearing that the liquidity could be removed at any time. This would mean that the project owner would run away with all the investments, leaving them with worthless tokens. Liquidity locks serve as a safeguard against such concerns. *UniCrypt*[24] is a highly popular DeFi platform that provides a range of services, including token locking. It automatically creates a smart contract that holds the liquidity pool token for the specified locking period. This makes it impossible for the liquidity provider to remove the liquidity during the locking period, providing an extra layer of security for the tokens and establishing trust among the investors. An alternative to this can be burning the liquidity pool token, which means getting rid of the liquidity pool token, which offers access to remove the liquidity. However, it is worth mentioning that this option may not be widely favored because there could be situations where the token owner may need to access the liquidity funds in order to relaunch the project or make other necessary adjustments.

2.5 Types of cryptocurrencies scams

Cryptocurrencies have made a reputation for themselves as breeding grounds for scams due to a lack of regulations. The blockchain offers anonymity, making it a paradise for scammers and allowing them to operate without being caught. The high potential returns on investment in the crypto market have drawn the attention of many individuals without any prior experience with blockchain technology. Unfortunately, most of them became easy targets for fraudsters, falling victim to cryptocurrency scams. Moreover, influencers and celebrities from all around the world are promoting cryptocurrencies without taking the necessary steps to fully understand the legitimacy of a project. Their lack of responsibility for the content they promote misleads their followers into investing in scams.

There are many types of cryptocurrency scams that have been observed in the past five years. It is important to be aware of them and protect yourself. DYOR is an acronym that stands for "Do your own research". It is commonly used in the cryptocurrency investing community to remind individuals to conduct their own research before making an investment decision and to avoid falling into the trap of scams.

2.5.1 Phishing scams

Phishing is a form of social engineering where attackers trick people into exposing sensitive information to them. Scammers impersonate legitimate websites or emails and try to make users reveal their private keys associated with their addresses. Once they get this information, they can access users' funds and steal their cryptocurrency. One prevalent form of this type of scam involves the creation of a new token that mimics a legitimate one by adopting its name and ticker symbol. Scammers then promote this fake token online, with the intention of misleading and defrauding unsuspecting investors.

2.5.2 Ponzi schemes

A Ponzi scheme is an investment scam where earlier investors benefit from the capital brought in by newer investors rather than from the profit generated by the investment. This gives the appearance that the investment is legitimate and generates profit. However, the scheme will eventually collapse. Ponzi schemes on the blockchain are sometimes advertised as mining or staking, which tricks people not familiar with this technology. There is a study[34] that conducts further analysis on these schemes and reports 630,000 USD gained through Ponzi schemes from more than 2000 individuals on the Ethereum Blockchain. Another study[45], estimated that the situation is similar on other chains. On Bitcoin, from September 2013 to September 2014, Ponzi schemes gathered more than 7 million USD.

2.5.3 Pump and dump schemes

Pump and dump schemes are investment frauds where a group of investors artificially inflate the price of a specific cryptocurrency, then sell their holdings at the inflated price, causing the price to collapse. In the cryptocurrency communities, the people who take part in these groups are often referred to as "whales". There is a study[46] that indicates that these groups operate on cryptocurrencies with market capitalizations below $50 million. The whales are usually individuals active in the Twitter and Telegram crypto communities. They buy a large quantity of the total supply of a token (pump), then start aggressively promoting the asset on social media platforms. Once the price peaks, they proceed to sell their investment (dump), causing the price of the asset to drop significantly, leaving many investors with a worthless investment.

2.5.4 Rug pulls

Rug pull refers to any malicious maneuver performed by developers to abandon a project and exit with all the investors' money. They are considered unethical and sometimes illegal. Rug pull projects often appear legitimate, and investors do not suspect any hidden risks associated with the project. These types of rug pulls usually fall into two main categories:

1. Soft rug pull: a rug pull where the developers gradually withdraw from the project, reducing their communications with investors. This can influence the market value of the project and make supporters take back their investments. Developers do not suddenly exit with all funds; they usually just remain with the tax transactions taken from the investors' transactions with the asset. This is considered just unethical, meaning that the token creators do not directly violate any legal rules.

2. Hard rug pulls: a rug pull where the developers suddenly exit with all the funds associated with the project. This would leave investors with little or no recourse to recover their initial investments, meaning that the project has totally collapsed. These actions are both unethical and completely illegal. If the developers are proven to have engaged in intentional theft, they may face charges under various criminal laws.

Liquidity theft

This is the simplest form of a rug pull. The token creator removes all the liquidity, leaving the investors with worthless tokens. A famous example is the Squid Game Token, which was a cryptocurrency project launched in September 2021, named after the famous TV series "Squid Game". It claimed to allow users to play games and earn rewards in the form of its native token, SQUID. Unfortunately, cryptocurrency communities started to question the lack of transparency of the project, the fact that the developers were anonymous, and the questionable marketing techniques. Just a few weeks after the project launch, the developers performed a hard rug pull, taking all the liquidity from the project and walking away with cryptocurrency worth $2.1 million[35]. This was considered a "hard rug pull".

Token dumping

Token dumping rug pull require the token creator to hold a large share of the total supply and suddenly sell the entire quantity. This causes the investors to remain with tokens that are essentially worthless, making it impossible to even recover their initial investments. On July 2022, the developers of the meme coin TeddyDoge performed a soft rug pull by selling $4.5M worth of tokens[39]. The price of the Teddy Doge token feel by 99.7% in less than 24 hours. This type of scam can be categorized as ”soft rug pull”, because the blockchain's transparency allows users to view the distribution of tokens. This means that investors could have seen the potential risk involved in investing in this project.

Limiting sell orders

Another example of a hard rug pull is when the smart contract includes functionalities that allow the creators to add restrictions and prevent users from selling their tokens. It is not uncommon for ERC-20 token smart contracts to implement small fees that are applied when the token is bought or sold. These are used to generate revenue for the token's creator. However, some of the implementations allow the token's owner to modify the transaction fee to any percentage they desire. If they choose to abuse this, they increase the tax to 100%, making it impossible for investors to sell the token. Another functionality that can be exploited is the ability to halt trading. This means that the owner of the token can permanently freeze trading on the token, restricting the ability of investors to recover their investments. These types of contracts are also referred to as honeypots. There is a study[44] that identifies 690 honeypot smart contracts with an accumulated profit of more than $90,000 for the honeypot creators.

2.6 Existing solutions

There are some existing platforms that are free to use and have addressed the detection of rug pulls on decentralized exchanges. Token Sniffer[22] helps you visualize simple heuristics of the token. It performs analysis, such as verifying the token distribution among the holders, in order to prevent a dump rug pull. Based on these and other features, it provides the user with an audit score, their own measure of how well a token meets the criteria for safety. However, they specify that a token with a high score may still have hidden malicious code, so they do not guarantee that token would not eventually turn into a rug pull. HoneyBadger[11] and Rugpulldetector[19] are tools that scan the smart contract of a token and try to determine if the presence of any malicious function can affect the investors. These come as support for people who are not used to Solidity as a programming language or for those who do not know anything about programming in general. Another tool is HoneypotDetector[12]. It simulates a buy and sell transaction to determine if a token is a honeypot or not. It is a common practice for investors to make use of this tool to avoid being tricked and losing their money by interacting with a malicious ERC-20 token.

Moreover, there are two studies that focus on an overview of various rug pulls and introduce the way they are executed.

The first paper[50] comes up with a dataset comprising over 10,000 scam tokens listed on Uniswap. It provides a method for collecting transaction events (such as mint, swap, and burn events) related to Uniswap. It presents an effective and accurate method for detecting scam tokens when labeling the dataset. Moreover, they perform an analysis of all these scams, and they estimate that the scammers have gained a profit of at least $16 million from around 40,000 potential victims.

The second study[40], complements the previous study by expanding the dataset with 18,000 scam tokens. It provided a theoretical classification for understanding different types of rug pulls and a new methodology for identifying rug pulls based on the transactions of a token. Moreover, there are presented two machine learning models that are able to distinguish between malicious and non-malicious tokens, demonstrating the usefulness of these models for detecting fraudulent activity in a token at an early stage.

2.7 Artificial Intelligence Models

2.7.1 Logistic Regression

Logistic regression[14] is a statistical model that is commonly used in machine learning for classification. The model uses a logistic function, also called the sigmoid function, to transform the linear combination of input features into a probability value between 0 and 1. The sigmoid function's role is to map any input to a value between 0 and 1, which can be interpreted as a probability. The model estimates the best set of weights to be applied to the input features in order to maximize the accuracy of the predictions. These weights are estimated by minimizing the logisitic loss function, which measures the difference between the predicted probability and the actual binary output value. Once the model is trained, it can be used to predict the probability of the binary output variable taking a certain value given new input features. If the probability is above a certain threshold, the output is predicted to be positive, and if it is below the threshold, the output is predicted to be negative.

2.7.2 Support Vector Machine (SVM)

Support Vector Machine[20] is a type of machine learning algorithm that aims to identify the best hyperplane that separates data points from different classes. This hyperplane is chosen in a way that maximizes the distance, known as the margin, between the hyperplane and the nearest points belonging to each class. The goal is to achieve the greatest possible margin while still correctly classifying the data points.

2.7.3 Random Forest

Before giving a brief explanation of how the random forest classifier works, we will first introduce the concept of decision trees[4]. A decision tree is a tree-like model where each non-leaf node represents a decision on a feature of the data, and the leaf nodes represent the class labels in a classification problem. The algorithm works by greedily splitting the dataset into smaller subsets based on the features, such that each subset either has datapoints with similar values or datapoints with the same class label. The objective of each split is to maximize the information gained. Once the best split is found, the subset is further divided into two smaller ones, and this process is repeated recursively until a stopping criterion is met.

A Random Forest classifier[17](see Figure 2.4) is a machine learning algorithm that uses a collection of decision trees. It belongs to the bagging family of algorithms, which involves training each classifier on a randomly selected subset of the training data. The prediction is then made by taking the majority vote of all the individual trees. One advantage of using random forests over a single decision tree is that it can reduce the variance of each tree. Since they use a random number of features, the model is less prone to overfitting the data, making it less likely to make errors on new, unseen data.

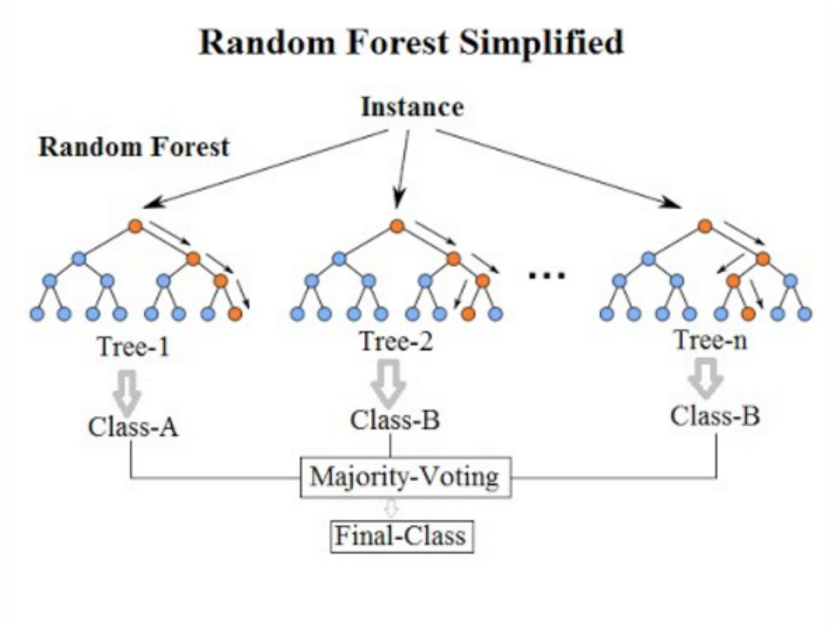


Figure 2.4: Random Forest Architecture[17]

2.7.4 eXtreme Gradient Boosting (XGBoost)

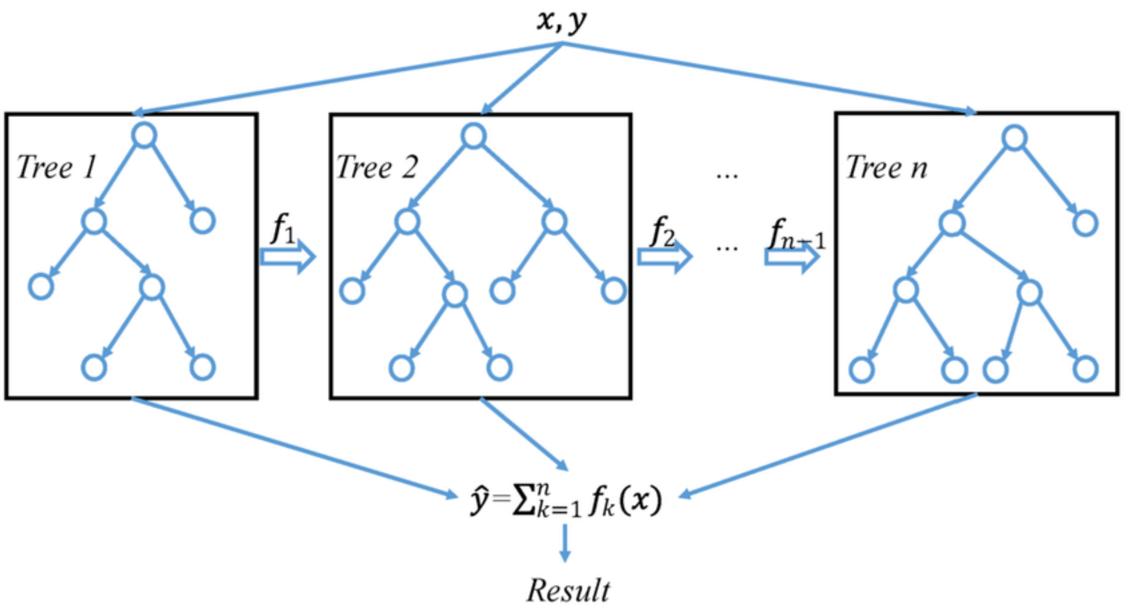


Figure 2.5: XGBoost Architecture[47]

XGBoost[32] is a machine learning algorithm that can be used for regression, classification, and ranking problems. It is a boosting algorithm that combines several weak models to create a strong one. The algorithm makes use of decision trees, where each new tree is trained to correct the errors of the previous tree. After each iteration, it evaluates the performance of the model and determines the best set of parameters for the next tree to be added to the ensemble. The predicted output of XGBoost is not obtained by adopting the majority vote heuristic used in Random Forest of combining the outputs of all the individual trees, but rather through a weighted sum of the outputs of the weak models in the ensemble (see Figure 2.5).

2.7.5 Feedforward Neural Network (FNN)

Feedforward Neural Network[37] is a type of artificial neural network that is designed to process input data through multiple layers of interconnected nodes, known as neurons. It is "feed-forward" because the data flows in one direction from input to output, without looping back through the nodes. The first layer of neurons, the input layer, receives the input data. The output of each neuron from the first layer is fed into the next layer, the hidden layer. The process can continue through any number of hidden layers until the final layer, the output layer, that produces the network's output. The diagram in Figure 2.6 illustrates a feedforward neural network architecture example with two hidden layers. The input layer of the neural network has 10 neurons, which receive 10 features as inputs. The first hidden layer has 8 neurons, and the second hidden layer has 4 neurons. Finally, the output layer consists of a single neuron. This neural network architecture is suitable for tasks that require predicting a single output value based on the given inputs, such as regression problems or binary classification problems.

It is essential to mention here that the activation function is applied to the output of each neuron in the hidden layers. It is used to introduce nonlinearity into the network, allowing it to model complex relationships between inputs and outputs. Without the activation function, the network would simply be a linear combination of its inputs and would not be able to represent more complex functions.

During the learning phase, the network is adjusting the strength (weights and biases) between neuronal connections, trying to minimize the difference between the actual output and the desired output. This is known as the backpropagation algorithm, which is a form of gradient descent algorithm that calculates the gradient of the error with respect to the weights and biases and then updates them to minimize the error.

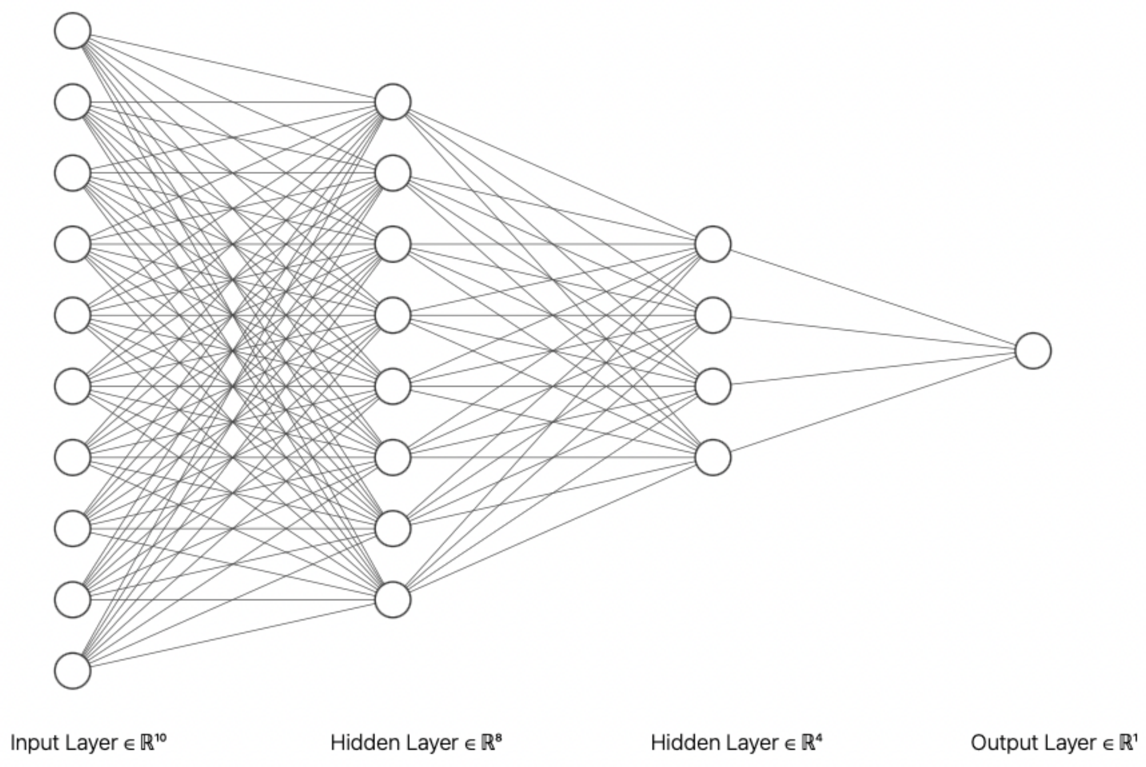


Figure 2.6: Feedforward Neural Network Architecture Example

Chapter 3

Method

In this chapter, the methodology and strategies employed in the development of a data science pipeline for detecting cryptocurrency rug pulls are presented. The pipeline involves multiple steps: data collection from diverse and representative sources, the processing and cleaning of the data, and the development of appropriate machine learning models for automated rug pull detection and their evaluation.

3.1 Dataset collection

The data collection efforts for this study build upon an existing dataset from a paper[40] in the literature. This initial dataset serves as the cornerstone for subsequent data collection and analysis, providing a solid foundation for the development of a more comprehensive and diverse dataset.

The dataset consists of 29,329 liquidity pools from various cryptocurrencies that are listed on the Uniswap V2 protocol. It includes a range of features that capture different aspects of token trading and liquidity pool activity. Furthermore, an additional file containing more liquidity pool features is also available, which can be used to expand the dataset even further.
All of these features can be grouped up into three different categories:

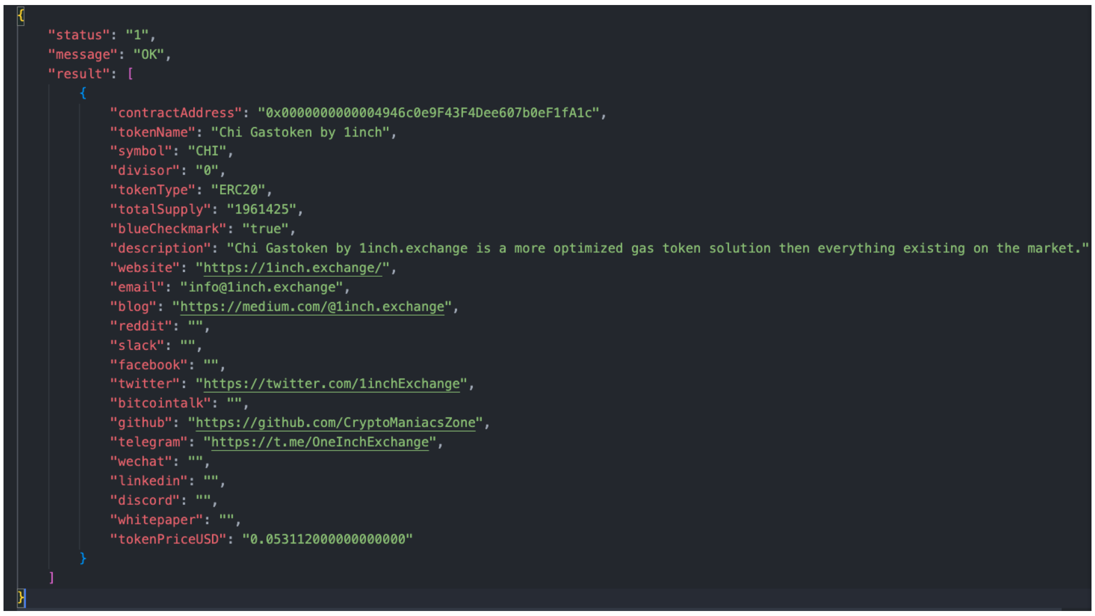
1. Pool Group
   - (a) n_syncs: The total number of liquidity pool Sync events emitted by Uniswap V2 (see section 2.4.1)
   - (b) WETH: the total value of ETH existing in the liquidity pool
   - (c) prices: the price of the token in ETH
   - (d) liquidity: the total number of tokens in the liquidity of the non-native cryptocurrency
   - (e) tx_curve: HHI[43] applied to the token distribution, which measures the distribution of tokens across holders
   - (f) difference_token_pool: The total number of transactions between token creation and pool creation
   - (g) burns: This feature is 1 if the liquidity has been burned (see section 2.4.1) and 0 otherwise.
   - (h) lock: This feature is 1 if the liquidity has been locked (see section 2.4.1) and 0 otherwise.
   - (i) yield - This feature is 1 if there is yield farming[31] involved and 0 otherwise. It is a form of lending crypto assets in order to generate passive income.

2. Liquidity Pool (LP) Token Group
   (a) Mint: The total number of liquidity pool Mint events emitted by Uniswap V2 (see section 2.4.1)
   (b) Burn: The total number of Burn events emitted by Uniswap V2 (see section 2.4.1)
   (c) Transfer: The total number of Transfer events emitted by Uniswap V2 (see section 2.4.1)
   (d) liq_curve: HHI[43] applied to the liquidity pool tokens, which measures the distribution of them among individuals
3. Token Transfers Group
   (a) num_transactions: The total number of transfers between EOAs with the ERC20 token
   (b) n_unique_addresses: The total number of unique addresses which interacted with the token
   (c) cluster_coeff - an heuristic for measuring the distribution of the token among the investors using transaction graphs

Unfortunately, Uniswap does not provide any information about the liquidity pool of a cryptocurrency, even if the scam has already occurred. The study includes a corresponding list of 27,588 labeled tokens using their own methodology of characterizing a token as a rug pull or as a legit token. The labels are 1 and 0, where 1 corresponds to a legitimate token and 0 corresponds to a rug pull token.

The next step for data collection was making use of different APIs to extend the number of existing features in the dataset presented above. As the API endpoints had restrictions on the number of calls allowed per minute, I had to incorporate specific delay periods between each call to prevent any rejections of the GET request. This was necessary to ensure smooth and uninterrupted execution of the scripts.

Etherscan API Pro



```
{
    "status": "1",
    "message": "OK",
    "result": [
        {
            "contractAddress": "0x0000000000004946c0e9F43F4Dee607b0eF1fA1c",
            "tokenName": "Chi Gastoken by 1inch",
            "symbol": "CHI",
            "divisor": "0",
            "tokenType": "ERC20",
            "totalSupply": "1961425",
            "blueCheckmark": "true",
            "description": "Chi Gastoken by 1inch.exchange is a more optimized gas token solution then everything existing on the market."
            "website": "https://1inch.exchange/",
            "email": "info@1inch.exchange",
            "blog": "https://medium.com/@1inch.exchange",
            "reddit": "",
            "slack": "",
            "facebook": "",
            "twitter": "https://twitter.com/1inchExchange",
            "bitcointalk": "",
            "github": "https://github.com/CryptoManiacsZone",
            "telegram": "https://t.me/OneInchExchange",
            "wechat": "",
            "linkedin": "",
            "discord": "",
            "whitepaper": "",
            "tokenPriceUSD": "0.053112000000000000"
        }
    ]
}
```

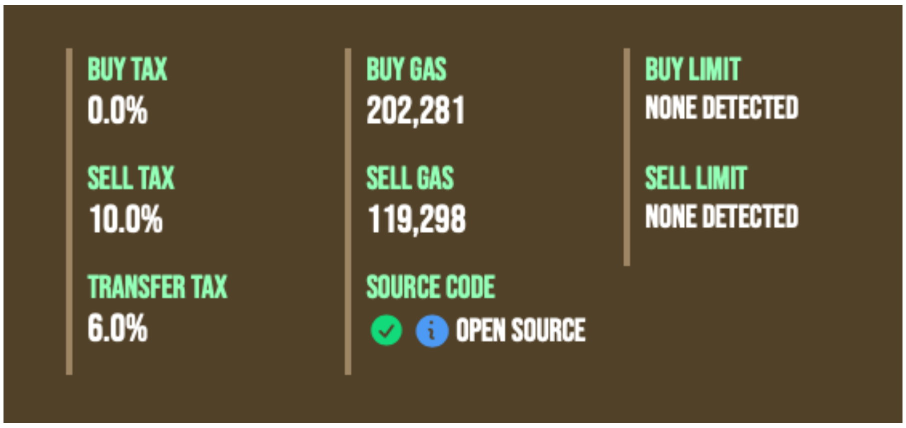
Figure 3.1: API response from Etherscan PRO API

Etherscan API Pro[10] is a set of web-based APIs that allow developers to programmatically access and retrieve data from the Ethereum blockchain. A useful set of features that can be included in the presented dataset is information related to the contract and the token communication channels.

I created two scripts that perform API calls for each token address from the previously mentioned dataset. The first endpoint used returns if the contract source code of the token is verified (see section 2.3). This feature, *contract_status*, will have a value of 1 if the token's smart contract is verified and 0 otherwise. Scammers usually try to hide the malicious behavior of their code; therefore, this feature can bring benefits in the learning stage.

The second script uses an endpoint that returns mostly social information about the cryptocurrency. In Figure 3.1, there is an API response generated by running this script. We are interested in extending the dataset with features that include online communication channels related to the token (website, email, Twitter, Telegram, etc.) and the total supply.

Honeypot API



BUY TAX
0.0%

SELL TAX
10.0%

TRANSFER TAX
6.0%

BUY GAS
202,281

SELL GAS
119,298

SOURCE CODE
✓ OPEN SOURCE

BUY LIMIT
NONE DETECTED

SELL LIMIT
NONE DETECTED

Figure 3.2: API response from Honeypot API

By the time of the study, 16 March 2023, Honeypot.is[12] did not include any API available to the public. Their platform was performing GET requests to an AWS hosted API on the client side. I used that endpoint and extracted valuable information about each existing token from the dataset. In Figure 3.2 there is an example of some of the information it provides which can be incorporated in the dataset. The following are the ones in which we are interested in collecting:

- Buy Tax: Custom tax on the token for buying
- Sell Tax: Custom tax on the token for selling
- Transfer Tax: Custom tax on the token for transferring
- Buy Limit: if there is a buy limitation per address
- Sell Limit: if there is a sell limitation per address
- Buy Gas: the amount of gas used for execution a buy transaction
- Sell Gas: the amount of gas used for execution a sell transaction
- Honeypot (omitted in the figure): it performs an analysis on Buy Gas and Sell Gas to determine if the token is a honeypot. It has value 1 if it is a honeypot and 0 otherwise.

CoinGecko API V3

Like the aforementioned APIs, I developed another script that calls an endpoint provided by the CoinGecko API[2] for each token. Unfortunately, due to the limited availability of information for the tokens used in this dataset, I was forced to exclude this set of features from the main analysis. However, I will cover this dataset in section 4.5 for further analysis.

3.2 Data preprocessing and Data cleaning

Cleaning the dataset

In the given dataset, there were numerous liquidity pools available for the same token address, which led to increased complexity. To address this, it was decided to retain only the most significant liquidity pool for each token address. The selection criteria were based on the pool's trading activity, with the pool having the highest number of interactions by traders being chosen. The next step involved adding the supplementary features (*lock, burns, yield*) provided by the study to the dataset.

Preprocessing the new features

For the features that include the online communication channels, which are: *description, website, email, blog, reddit, slack, facebook, twitter, bitcointalk, github, telegram, wechat, linkedin, discord, whitepaper*, the approach chosen was to transform this into categorical data, 1 if the token includes data in the corresponding field and 0 if it does not include it.

Building the enhanced dataset and scaling the data

The next step was to associate all the newly created features with the corresponding entries in the existing dataset.

Then, I applied the *StandardScaler* provided by *sklearn.preprocessing* to all the features of the dataset. The *StandardScaler* standardizes features by subtracting the mean from each feature and then scaling the feature to the unit variance. The formula used for transformation is: $z = (x - u)/s$, where z is the standardized feature value, x is the original feature value, u is the mean of the feature values, and s is the standard deviation of the feature values.

3.3 Feature selection

Highly correlated features can cause problems in a machine learning model by introducing multicollinearity, which can make it difficult for the model to estimate the contribution of each feature to the target variable. Therefore, for this, I have chosen to plot the correlation matrix for all the features. By analyzing the correlation matrix, I was able to identify pairs of features that are highly correlated with each other. Using domain knowledge, I chose to eliminate the ones that are correlated above a certain correlation coefficient.

Then the next step was to calculate the feature importance using a Random Forest machine learning algorithm, which allowed me to determine the contribution of each feature to the model's prediction. Based on the feature importance ranking, I identified a subset of features that had very low importance and removed them from the dataset. This helped in selecting the most informative features for the model and reducing the dimensionality of the data.

3.4 Model selection

During the model selection phase, the performance of various supervised machine learning algorithms was examined. The selection of models was based on algorithms that are typically used for identifying fraud patterns. From these evaluations, I identified three top-performing models that were best suited for detecting rug pulls in the cryptocurrency domain.

Logistic Regression

Logistic regression is a relatively simple model that can quickly identify patterns and relationships in the data. They are computationally efficient, making them well-suited for large datasets, which we are using in this study.

Support Vector Machine (SVM)

Support Vector Machine can be a suitable model for this scenario, as they can identify patterns in data that can be used to distinguish between legitimate and fraudulent tokens. This model works well with high-dimensional feature spaces, which is often the case with cryptocurrency data, as there are many variables to consider. It handles non-linearly separable data, meaning that even if the data is not linearly separable, it can find a boundary that best separates the two classes. This is important as fraudulent cryptocurrency tokens may not always follow a clear pattern and require a more complex model to identify them.

Random Forest

The random forest model provides a measure of feature importance, which can be useful in identifying the most relevant features for the classification task. This information can be used to gain insights into the underlying factors that contribute to cryptocurrency rug pulls and improve overall detection performance.

eXtreme Gradient Boosting (XGBoost)

XGBoost is a suitable machine learning model for the dataset used in this study due to its ability to handle large datasets with a high number of input features, which is important for analyzing complex cryptocurrency data. Moreover, XGBoost includes techniques such as weight balancing, early stopping, and regularization to handle imbalanced class distributions, which is often a challenge in fraud detection problems.

Feedforward Neural Network

In Figure 3.3 there is a proposed architecture for a Feedforward Neural Network with two hidden layers. In order to prevent overfitting, I will make use of the Dropout regularization technique, which randomly "turns off" some of the neurons in the layer with a certain probability. The dropout will be added after each hidden layer. After this, I will perform hyperparameter tuning for the number of neurons in the hidden layers and the dropout values (see section 3.5). As an activation function for the hidden layers, there will be ReLU (Rectified Linear Unit), as it has been shown to be effective in preventing the vanishing gradient problem and accelerating the convergence of the model during training. For the output layer, I am using the sigmoid activation function because it is well-suited for binary classification problems, such as identifying rug pulls versus legitimate cryptocurrency tokens.

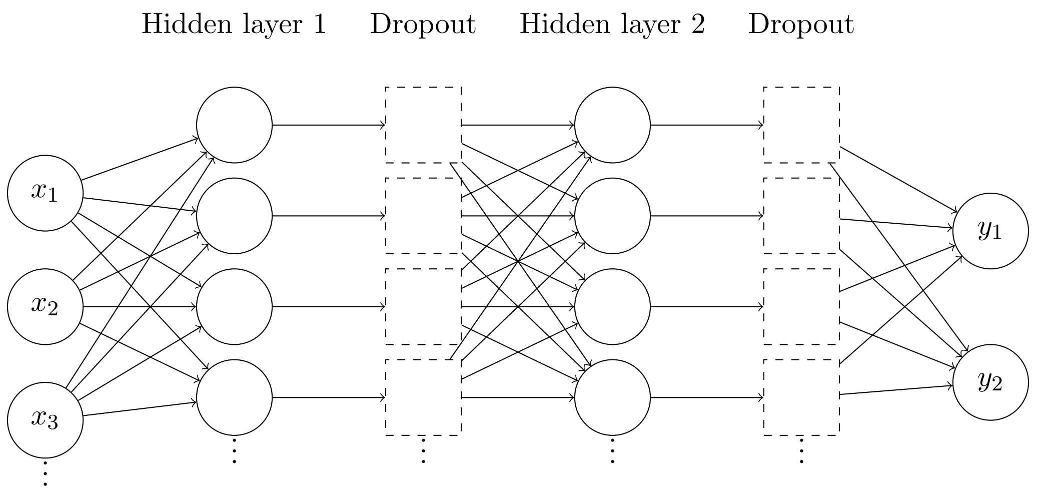


Figure 3.3: Feedforward Neural Network Architecture

3.5 Hyperparameter tuning

In this section there will be discussed the hyperparameter tuning optimizations for the top three most performant models from Section 3.4.

Optuna[16] is the framework used for hyperparameter optimization. It automates the process of tuning the hyperparameters of artificial intelligence models to improve their performance. The optimizer evaluates the performance of the model based on the result of an objective function for a fixed number of iterations or until a stopping criterion is met. I created custom objective functions for each model that try to maximize the performance of the models. An objective function samples for each corresponding hyperparameter of the model that is being tuned from a given interval of values. Then the function returns the performance of the model on the validation set.

For all three selected models, I have used a total of 100 trials, trying to maximize the score of the defined objective function. Below, there will be presented the hyperparameters tuned for the model and the corresponding intervals of integer or float values or the set of categorical values.

3.5.1 Random Forest

- n_estimators - The number of trees in the forest: [50, 500], integer
- max_depth - The maximum depth of a tree: [10, 20], integer
- min_samples_split - The minimum number of samples required to split an internal node: [2, 10], integer
- min_samples_leaf - The minimum number of samples required to be a leaf node: [1, 10], integer
- max_features - The number of features to be considered when looking for the best split: ['sqrt', 'log2'], categorical
- class_weight: [None, 'balanced'], categorical

3.5.2 XGB Boost

- max_depth - The maximum depth of a tree: [2, 10], integer
- learning_rate - The step size of each iteration while moving toward the minimum of the loss function: [0.001, 0.1], float
- subsample - The fraction of observations to be randomly sampled for each tree: [0.5, 1], float
- colsample_bytree - The fraction of columns to be randomly sampled for each tree: [0.5, 1], float
- gamma - The minimum loss reduction required to make a split: [0.001, 1], float
- reg_lambda - L2 regularization term on weights: [0.1, 10], float
- reg_alpha - L1 regularization term on weights: [0.1, 10], float
- min_child_weight - The minimum sum of weights of all observations required in a child: [1, 10], integer

3.5.3 Feed Forward Neural Network

- n_neurons_1 - Number of neurons in first hidden layer: [16, 512], integer
- n_neurons_2 - Number of neurons in second hidden layer: [16, 512], integer
- dropout_rate_1 - Dropout rate after the first hidden layer: [0.0, 0.5], float
- dropout_rate_2 - Dropout rate after the second hidden layer: [0.0, 0.5], float

3.6 Validation Methods

Before model selection, I used the train-test split technique, which involves splitting a dataset into two parts: the training set and the testing set. The training set was used for model selection and hyperparameter tuning for training the model. The testing set was used to evaluate the performance of these models on unseen data.

Based on the size of the dataset used, we carefully selected a split ratio of 80%-20%, which strikes a balance between the model's need for a sizable training set to learn complex patterns and the requirement for sufficient testing data to evaluate its performance on new data. This thoughtful allocation ensures that the model has access to a vast array of training examples while also providing ample opportunities to test its generalization capabilities, preventing overfitting the model on the collected dataset. In order to ensure a representative distribution of the classes, the split was performed in a stratified manner based on the class labels of the imbalanced dataset. This should ensure a reliable evaluation of the models.

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

Figure 3.4: Precision formula

As an evaluation metric, it is important to take into consideration that there is an imbalance in the dataset. In the total number of 26,839 tokens, there are 622 labeled with 1 as legitimate safe tokens, and the rest of them are labeled with 0 as scam tokens. In an imbalanced dataset,

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

Figure 3.5: Recall formula

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.3}$$

Figure 3.6: F1-score formula

where one class is represented by significantly fewer samples than the other class, accuracy is considered a misleading metric, as a model can achieve high accuracy by simply predicting the majority class. To avoid this issue, in addition to accuracy, I have decided to use F1-score, precision, and recall as evaluation metrics.

The F1-score (see formula 3.6) represents the harmonic mean of precision and recall. Precision (see formula 3.4) measures the proportion of true positive predictions among all positive predictions. Recall (see formula 3.5) measures the proportion of true positive predictions among all actual positive entries in the dataset.

To ensure that the model does not overfit on the training dataset, stratified K-fold cross-validation with k = 5 was performed. This technique was chosen because it ensures that the training and validation sets have the same proportion of each target class. It is particularly useful in this study as the dataset is imbalanced, with more rug pulls than legitimate tokens. This approach can help improve the generalization of the model by training it on a more diverse set of examples and reducing the risk of bias.

To evaluate the performance of the model in each fold, metrics such as accuracy, precision, recall, and F1-score were computed. The average accuracy score across all 5 folds and the average F1-score across all 5 folds were determined. Similarly, the average precision and recall across all folds were also computed.

This approach provides a more reliable estimate of the model's overall performance as it considers the variability in the evaluation metrics across different folds. Computing the average scores across all folds enables the identification of any consistent trends or patterns in the model's performance, as well as its ability to generalize to new and unseen data.

In order to perform sensitivity analysis, which in this study will focus on artifically adjusting the proportion of classes in the training data and evaluating how this affects the model's performance metrics, we have used the Syntethic Minority Over-sampling Technique (SMOTE). The data augmentation technique creates new samples of the minority class by interpolating between existing samples. We employed this technique to address class imbalances in the training data and evaluated the performance of the most performant model on different class balances.

Feedforward Neural Network Validation Methods

For the neural network, it is crucial to provide detailed information about the optimizer, loss function. This is particularly important compared to other machine learning models previously mentioned, as neural networks have a much larger number of parameters and are more complex to train.

1. Optimizer - The optimizer used for updating weights during training - Adam, which is known for its ability to converge quickly and its ability to handle noisy or sparse gradients. It was used within the following configuration:

   (a) Learning rate: 0.001

   (b) Beta 1: 0.9
   (c) Beta 2: 0.999
   (d) Epsilon 1×10^7
   (e) Decay: 0

2. Loss function: The loss function used is the binary cross-entropy function which goal is to classify input into one of two classes. This is exactly what is needed for our study case.

3. Early stopping: This regularization technique has been introduced in the proposed neural network architecture to make sure the model does not become too complex and start to fit the noise in the training data, rather than the underlying patterns. For training, the early stopping was set to happen after ten consecutive epochs in which the loss function stopped decreasing, meaning that the performance of the model did not improve among the last ten epochs.

Chapter 4

Results

4.1 Dataset Analysis

4.1.1 Dataset preprocessing and cleaning

The original dataset contained 29,329 liquidity pools of tokens. However, after conducting data preprocessing and cleaning, the resulting dataset consisted of 26,839 liquidity pools. This indicates that a total of 2,490 liquidity pools were removed from the original dataset during the data cleaning process.

Model		**Literature dataset**	**Enhanced dataset**
Random Forest	Accuracy	0.994	0.995
	F1 Score	0.87	0.89
XGBoost	Accuracy	0.994	0.996
	F1 Score	0.88	0.91
Neural Network	Accuracy	0.994	0.995
	F1 Score	0.86	0.88

Table 4.1: Impact of the Enhanced dataset on model performance

Table 4.1 presents a comparison of model performance when trained on the literature dataset versus the enhanced dataset, highlighting the impact of the enhanced dataset on the accuracy and F1 score metrics. The F1 score and accuracy are obtained by the models by predicting on unseen data, and while there is a small improvement in accuracy for some models, the F1 score shows a consistent increase of 2-3% across all three models, suggesting that the enhanced dataset has a positive impact on the performance of these models.

4.1.2 Feature selection

In Table 4.2 it can be observed the features with the lowest importance score, provided after training the dataset on a plain random forest model. The decision was made to drop all 13 of these features from the table because they have the least impact on the prediction of the target variable.

The next step was looking at the correlation matrix of all the features. It was hard to visualize because it was a square matrix with dimensions of 40 x 40, resulting in a total of 1764 cells to be analyzed. The approach taken was to filter out the correlated features with an absolute value of the correlation coefficient over 0.90, which pairs can be observed in Table 4.3. Taking into consideration domain knowledge, the following six features have been dropped: Burn, sell_gas, description, email, twitter, telegram. After eliminating the less important features and the highly correlated ones, the remaining number of features used for model training and evaluation is 21.

Feature name	Feature score
discord	0.000688
linkedin	0.000563
burn	0.000509
lock	0.000385
bitcointalk	0.000355
slack	0.000240
buy_tax	0.000148
wechat	0.000088
prices	0.000048
liquidity	0
max_tax_amount	0
tx_curve	0
totalSupply	0

Table 4.2: Feature with feature importance score below 0.001

Feature pair	Correlation coefficient
(Mint, Burn)	0.95
(honeypot, buy_gas)	-0.95
(buy_gas, sell_gas)	0.95
(description, website)	0.97
(description, email)	0.91
(description, twitter)	0.93
(website, twitter)	0.92
(twitter, telegram)	0.92

Table 4.3: Highly correlated features

4.2 Model Analysis

In this section, there will be presented the results from the model selection and hyperparameter tuning.

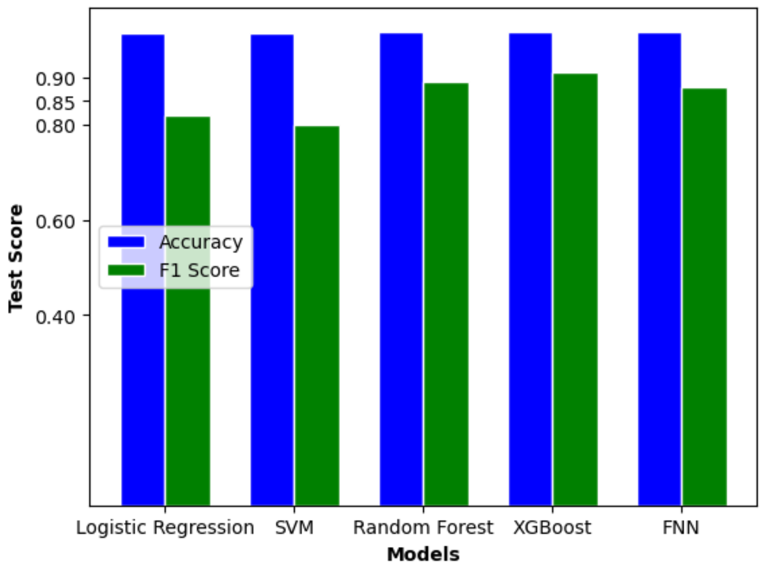


Figure 4.1: Model selection results

To kick things off, we started our model selection process by training our dataset on five diverse models: Logistic Regression, Support Vector Machine (SVM), Random Forest, XGBoost, and our proposed Feedforward Neural Network. In Figure 4.1, there are the performance results of each of the five models on the test set. While the accuracy was high due to the imbalanced nature of the dataset, the top three models based on their F1-score were the Random Forest, XGBoost, and the proposed FNN architecture model.

4.2.1 Random forest

Hyperparameter	**Value**
n_estimators	200
max_depth	20
min_samples_split	5
min_samples_leaf	1
max_features	'sqrt'
class_weight	None

Table 4.4: Random Forest model - Hyperparameters

Table 4.4 displays the optimal configuration settings for the hyperparameters, which were obtained through the use of the Optuna framework.

	Accuracy	Recall	Precision	F1	F1 minority
5-Fold Stratified CV Mean	0.995	0.88	0.91	0.90	0.90
Test Set	0.996	0.90	0.93	0.91	0.91

Table 4.5: Random forest tuned model score

In Table 4.5, you can see the evaluation score for the model. A high mean score obtained from the cross-validation technique indicates that the model is consistently performing well on different subsets of the training data, which suggests that it may generalize well to new, unseen data. By looking at the performance on the test set of the tuned model against that of the non-tuned model (see Table 4.1, it appears that there is a 2% improvement in the F1 score. As expected, the accuracy is close to 100%, because the model is trained on a heavily imbalanced dataset. There is a high score for both F1 and F1 minority class 91%, indicating that the model is performing well overall and also on minority class.

Figure 4.2 shows the confusion matrix for the trained Random Forest model. The results indicate that for the rug pull class, only a very small number of misinterpretations occurred, with just nine instances where rug pulls were mislabeled as legitimate tokens. In contrast, for the legitimate token class, approximately 9% of the results were mislabeled.

Based on the results obtained from evaluation of the model on unseen data, as well as the mean of cross-validation results, and analysis of the confusion matrix, it can be concluded that the model is performing well and is not overfitting.

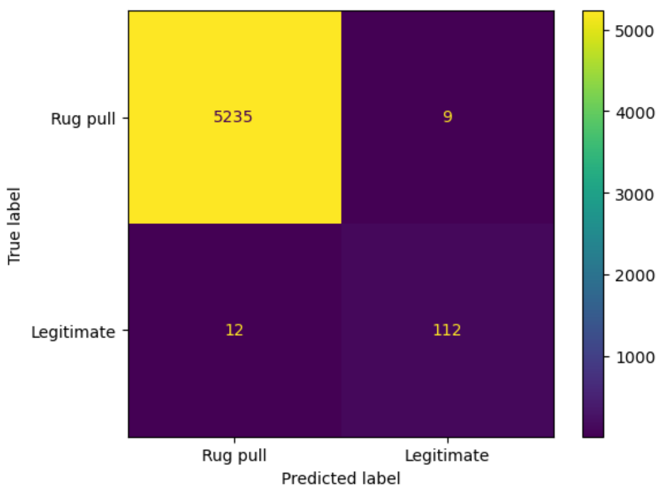


Figure 4.2: Random Forest - Confusion matrix

4.2.2 XGBoost

The Optuna framework was used to determine the best configuration settings for the hyperparameters of the XGBoost model, and these optimal settings are presented in Table 4.6.

Hyperparameter	**Value**
max_depth	6
learning_rate	0.09975769900178301
subsample	0.9871936147154661
colsample_bytree	0.9457019542955731
gamma	0.4278545378845186
reg_lambda	5.970927594444518
reg_alpha	0.16187754457940612
min_child_weight	2

Table 4.6: XGBoost model - Hyperparameters

	Accuracy	Recall	Precision	F1	F1 minority
5-Fold Stratified CV Mean	0.995	0.89	0.92	0.90	0.90
Test Set	0.996	0.90	0.94	0.93	0.93

Table 4.7: XGBoost tuned model score

By tuning the XGBoost model, in Table 4.7 it can be seen that the F1 score is 93%, showing a 2% improvement when comparing it to the plain model (see Table 4.1). Similar to the Random Forest model, the evaluation metrics (accuracy, recall, precision, F1 score, and F1 minority score) for the model being considered have high scores on both the cross-validation folds and the unseen data (test set). The confusion matrix shown in Figure 4.3 indicates that the model is performing similarly to the Random Forest model, with high scores for True Positives and True Negatives, and relatively low scores for False Positives and False Negatives. Additionally, it appears that the number of mislabeled legitimate tokens has decreased, which is a positive sign indicating that the model is becoming more accurate in predicting the class of legitimate tokens. Based on the results discussed, the model is unlikely to be overfitting.

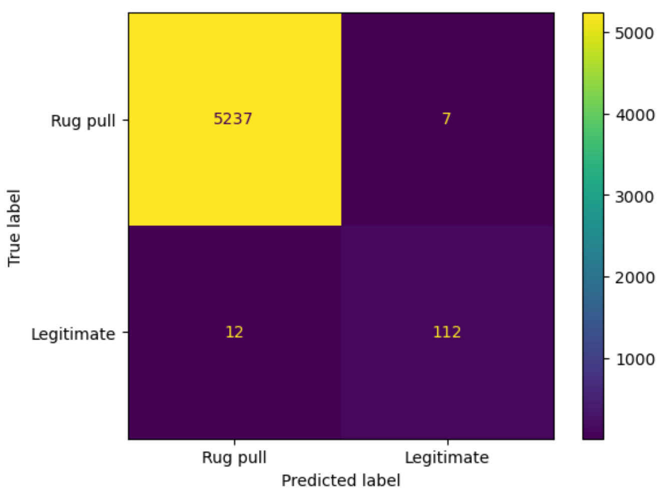


Figure 4.3: XGBoost - Confusion Matrix

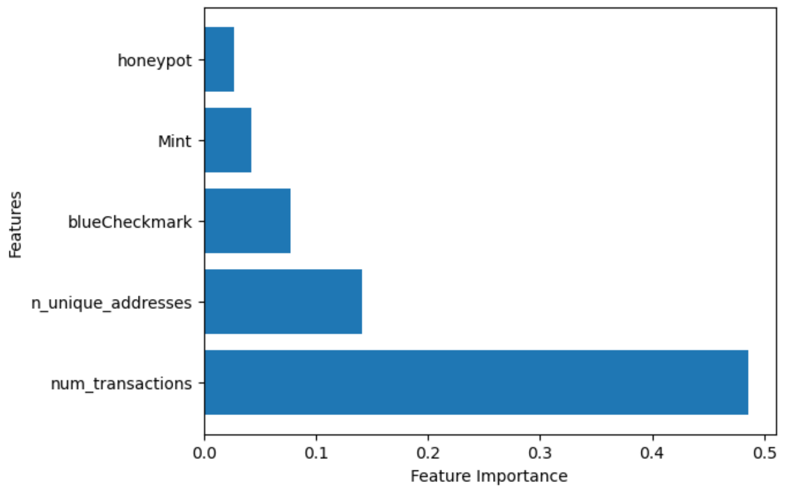


Figure 4.4: Top 5 Feature Importance

In Chapter 5, we will further analyze and discuss the five most important features presented in Figure 4.4 and their significance in the context of our study.

4.2.3 Feedforward Neural Network

Similar to the other models, the optimal configuration settings for the hyperparameters of the feedforward neural network model can be found in Table 4.8.

Table 4.9 shows that the F1 score on unseen data for the tuned neural network is 91%, which represents the biggest improvement (3%) among the three models when compared to their non-tuned versions. Similar to the Random Forest and XGBoost models, the tuned neural network demonstrates good performance with mean scores above 90% for both the F1 score and the F1 minority on the 5-fold cross-validation splits as well as on the unseen data (test set). Additionally, the confusion matrix (see Figure 4.5 provides further insights that the model is not overfitting and is performing well on both classes. These results suggest that it is a robust model that generalizes well and is effective in identifying both legitimate and malicious tokens.

Hyperparameter	**Value**
batch size	128
number of epochs	100
n_neurons_1	200
n_neurons_2	120
dropout_rate_1	0.1
dropout_rate_2	0.05

Table 4.8: FNN model - Hyperparameters

	Accuracy	Recall	Precision	F1	F1 minority
5-Fold Stratified CV Mean	0.995	0.86	0.95	0.90	0.90
Test Set	0.996	0.87	0.96	0.91	0.91

Table 4.9: FNN tuned model score

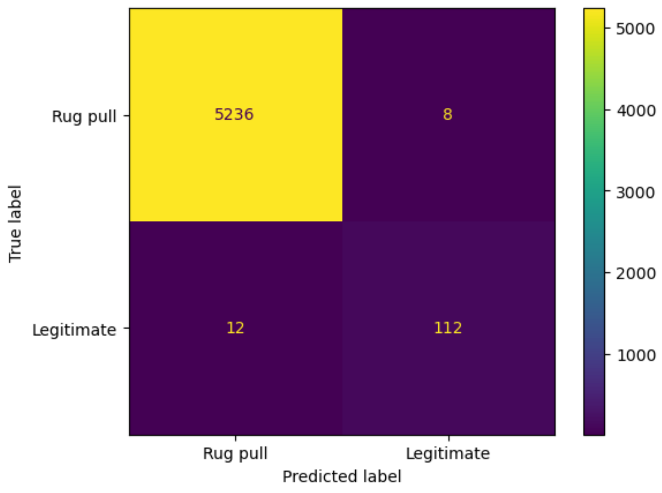


Figure 4.5: FNN - Confusion Matrix

4.3 Sensitivity Analysis

In this section, sensitivity analysis will be conducted on the tuned XGBoost model because it has the best overall performance. In the context of machine learning, sensitivity analysis is the process of examining how changes in input features affect the output of a model. In the case of our study, the focus is on how changes in class proportions would impact model performance. By conducting sensitivity analysis in this manner, we can evaluate how well the models would perform under different market conditions where the proportion of malicious tokens varies. This provides valuable insights into the robustness and reliability of the models as well as their potential effectiveness in real-world scenarios.

In the used dataset, there is a ratio of 1:42 of the classes, meaning that there is a legitimate token in every 42 scam tokens. It is important to mention that before generating any syntethic data, the dataset will be split into a training test and a test set. This avoids overfitting on the existing dataset and ensures that the model is evaluated on a representative test set. The following ratio has been used for retraining the model:

- 1:22 - increasing the number of legitimate tokens by creating artificial samples for the minority class
- 1:1 - increasing the number of legitimate tokens by creating artificial samples for the minority class until the dataset becomes balanced

- 22:1 - dropped a significant number of scam tokens entries from the syntethic generated balanced dataset making the scam token class become the minority class

Class Ratio	**Accuracy**	**Recall**	**Precision**	**F1**	**F1 Minority**
1:22	0.995	0.89	0.90	0.89	0.89
1:1	0.993	0.94	0.80	0.87	-
22:1	0.994	0.85	0.92	0.89	0.89

Table 4.10: Performance of XGBoost Model on different class distributions

Based on the results from Table 4.10, it appears that the model has consistent performance, with an F1 score over 87% and an accuracy of 99% in all three scenarios. In the imbalanced datasets, the F1 minority score is the same AS the F1 score, meaning that it does a good job of classifying the minority class.

While the overall performance is not as good as the results obtained on the original dataset distribution (see section 4.2.2 for the results), one reason for the lower performance on the new class distribution could be that the model was hyper-tuned for that specific dataset.

4.4 Feature subset selection

In this section, we trained plain XGBoost models on different categories of features to evaluate their performance and identify the most important features for the detection of rug pulls in cryptocurrency markets. By dividing the input features into distinct categories and training separate models on each, we are able to gain insights into which categories are most informative for making accurate predictions.

The features have been split into the following categories:

- Blockchain Metrics (see Table 4.11): Features related to the token characteristics on the market.

- Social Metrics (see Table 4.12): Features related to the token community data.

num_transactions	n_unique_addresses	cluster_coeff	tx_curve
liq_curve	Mint	Burn	Transfer
difference_token_pool	n_syncs	WETH	prices
liquidity	burn	lock	yield
contract_status	honeypot	max_tx_amount	buy_tax
sell_tax	buy_gas	sell_gas	totalSupply

Table 4.11: Blockchain Metrics - Feature group

blueCheckmark	description	website	email
blog	reddit	slack	facebook
twitter	bitcointalk	github	telegram
wechat	linkedin	discord	whitepaper

Table 4.12: Social Metrics - Feature group

In Figure 4.6, it can be observed that the overall performance of the model trained on the blockchain metrics subset of features is significantly higher than the model trained on the social metrics subset of features, with the F1 score being 3x higher.

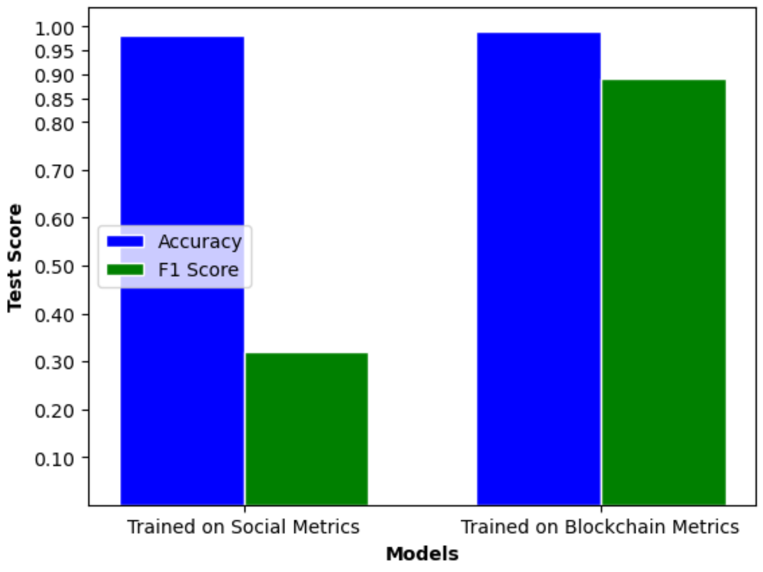


Figure 4.6: Feature subsets - Performance

4.5 CoinGecko features

As noted in 3.1, data was gathered from the CoinGecko API. Although this data provided valuable insights, it was not included in the final presented dataset as it only covered around 2% of the existing liquidity pools from the original dataset. Despite the limited coverage, the insights obtained from the CoinGecko API data were taken into consideration during the analysis, which helped to enrich the overall findings of the study.

Although some of the descriptions are brief due to a lack of details provided in their documentation, the following new features are obtained from this API:

- tickers: The number of exchanges where the token listed
- telegram_users: The number of telegram users on the token's channel
- market_cap_rank: The market capitalization rank
- com_score: Community score
- dev_score: Developers score
- liq_score: Liquidity score
- cg_score: The CoinGecko score
- cg_rank: The CoinGecko rank

Dataset	**Accuracy**	**F1**	**F1 minority**
Without CG features	0.98	0.98	0.13
With CG features	0.99	0.99	0.76

Table 4.13: Small dataset experiment with/without CG data

A dataset of size 498 was formed by removing all entries from the original dataset that did not have features available through the CoinGecko API. It is worth mentioning that, similar to the previous dataset, the presented dataset is also imbalanced. However, the imbalance is different in that there are more legitimate tokens than scams.

The enhanced dataset obtained in this study (evaluated in Section 4.1) was used to train a plain XGBoost model, both with and without additional features provided by the CG API. As shown in Table 4.13, the model trained solely on the enhanced dataset without CG features had a very low F1 minority score, indicating poor performance in classifying the minority class.

However, when trained on the dataset that includes CG features, the F1 minority score improved significantly to 76%. Feature importance analysis revealed that *cg_rank* was the most important feature, with a score of 0.61. This feature provided by CG had the greatest impact on improving the model's performance, increasing the F1 minority score from 13% to 76%. These results highlight the importance of incorporating CG features in the classification of rug pulls and demonstrate the significant impact they can have on model performance.

Chapter 5

Discussion

5.1 Achievements

The main contributions of this study include:

- Presented a background on the fundamental principles of blockchain technology, utilizing relatable analogies from traditional financial systems to enhance understanding, classified different categories of rug pulls, and explained how each one operates.
- Contributed to enhancing the existing dataset in the literature by increasing the number of features up to 40, which enable more accurate predictions.
- Various machine learning models were evaluated, and their performance was analyzed and presented. The models were further optimized by tuning their hyperparameters, obtaining F1 scores of up to 93%.
- Proposed the first architecture in the literature for a neural network, demonstrating its potential and effectiveness as a useful tool for identifying fraudulent cryptocurrency tokens.
- Through sensitivity analysis, we evaluated the impact of varying proportions of the classes in the dataset on model performance, providing crucial insights into their reliability and effectiveness in diverse market scenarios.
- Performed feature subset selection to identify the most significant categories of features that should be prioritized for data collection.
- Analyzed the potential impact of incorporating CoinGecko data on improving model performance.

5.2 Findings

Overall, this study demonstrated that artificial intelligence can contribute to the identification of potential rug pulls in cryptocurrencies. By leveraging machine learning techniques, we were able to effectively detect fraudulent transactions and distinguish them from legitimate ones, showcasing the potential of AI in preventing financial scams in the cryptocurrency space. We looked in depth at three different models: Random Forest, XGBoost and a Feedforward Neural Network model with two hidden layers.

The best performance was achieved by the XGBoost model, with a F1 score of 93% on unseen data. By examining the five most important features identified by the XGBoost model (see Figure 4.4), we can make several assumptions about their significance in detecting cryptocurrency scams. The first two features represent the number of transactions and the number of unique addresses that interacted with the token. From this, it can be assumed that the duration of time that a token has existed on the blockchain and the number of interactions it has received are important factors in distinguishing legitimate tokens from rug pulls. Specifically, the popularity of legitimate tokens tends to grow over time as more people interact with them, while rug pulls are short-lived and typically experience a decline in interactions after the rug is pulled. The blue checkmark feature is an award received by Etherscan if the project meets certain criteria[1] such as operating for a longer period of time, which in theory should be awarded only to reliable cryptocurrencies. The Mint feature may be an important indicator of legitimate tokens, as these tokens are more likely to progressively add liquidity and trigger more mint events on Uniswap V2. The Honeypot feature is a crucial indicator of potential rug pulls, as tokens that are unable to be sold are likely fraudulent. Additionally, we evaluated the features provided by the CoinGecko API and found that the CoinGecko ranking feature can significantly contribute to classifying tokens. However, it should be noted that the CoinGecko platform manually analyzes tokens before listing them, making it unlikely for rug pulls to be available on their API.

In order to evaluate the robustness of the proposed XGBoost model, we conducted a sensitivity analysis, which demonstrated good performance under different scenarios. By investigating the model's performance under diverse circumstances, we were able to assess its potential for real-world deployment and its ability to adapt to changing market conditions. These changes could arise from government regulations on cryptocurrencies, which may lead to a decrease in the number of scams.

The results of this study demonstrated that the proposed neural network architecture can achieve a high level of accuracy in detecting potential rug pulls in cryptocurrencies. With an F1 score of 91%, the model was able to effectively classify fraudulent tokens from legitimate ones. These findings highlight the potential of using neural networks as a powerful tool for fraud detection in the cryptocurrency market.

The feature subset selection process grouped the features into two subsets and revealed that Blockchain Metrics of the token were overall more important than its Social Metrics. These findings suggest that future studies may benefit from integrating additional data into the Blockchain Metrics category in order to enhance the performance of the models trained for detecting rug pull cryptocurrencies.

In Section 3.3, low feature importance values suggest that certain platforms such as Discord, LinkedIn, Slack, Bitcointalk, and WeChat are not commonly used in ERC-20 tokens. It is also noted that the price of the token and liquidity are not important features because they may not be good indicators of cryptocurrency rug pulls as they can be easily manipulated by scammers. Total token supply and buy tax may not be important because each token can have its own unique tokenomics (the economics of a cryptocurrency token) structure based on the project's plan. By looking at the most highly correlated features, we can assume that the presence of information such as description, email, Twitter, website, and Telegram channel is a common characteristic among legitimate cryptocurrencies. It is also worth noting that the pair of correlated features, buy gas and sell gas, have a strong correlation with the honeypot feature. This might be because a honeypot often comes with a super high gas price, making it really hard for people to buy or sell the token.

5.3 Future work

In addition to the findings discussed earlier, there are several areas for potential improvement in future studies. One of these areas is the creation of a better dataset: incorporating additional features that can be included in the Blockchain Metrics category can bring significant performance improvements to the model. In this study, the majority of the data from this group is obtained from the Uniswap V2 liquidity provider for the tokens that operate on the Ethereum blockchain. However, many cryptocurrencies operate on multiple chains, so there is more data that can be analyzed and used to enhance the detection of rug pulls.

Furthermore, exploring different neural network model architectures could lead to improved performance. For example, experimenting with deep learning models such as Recurrent Neural Networks (RNNs) may provide better results for detecting rug pulls, but this implies incorporating sequential data within the features of the dataset.

The ultimate goal of this research is to create a pipeline for constantly extracting data from the blockchain network and training the model in real-time. By doing so, we can continuously improve the accuracy of the model and provide a more effective solution for detecting rug pulls in the cryptocurrency market. Moreover, incorporating the model into an accessible platform for users can be a game-changer in preventing people from falling prey to fraudulent schemes and losing their hard-earned money. Ultimately, the goal is to make the cryptocurrency market a safer place for all participants and enhance trust in blockchain technology.

Bibliography

[1] The blue checkmark. https://info.etherscan.com/the-blue-checkmark/. Accessed on May 1, 2023.

[2] Coingecko api. https://www.coingecko.com/en/api/documentation. Accessed on May 1, 2023.

[3] Curve docs. https://resources.curve.fi/base-features/understanding-curve. Accessed on May 1, 2023.

[4] Decision tree. https://scikit-learn.org/stable/modules/tree.html. Accessed on May 1, 2023.

[5] Dogecoin. https://dogecoin.com/. Accessed on May 1, 2023.

[6] Elliptic curve digital signature algorithm. https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages. Accessed on May 1, 2023.

[7] Erc-20 token standard. https://ethereum.org/en/developers/docs/standards/tokens/erc-20/. Accessed on May 1, 2023.

[8] Etherdelta. https://crypto.marketswiki.com/index.php?title=EtherDelta. Accessed on May 1, 2023.

[9] The ethereum blockchain explorer. https://etherscan.io/. Accessed on May 1, 2023.

[10] Etherscan api pro. https://docs.etherscan.io/api-pro/etherscan-api-pro. Accessed on May 1, 2023.

[11] Honeybadger. https://github.com/christoftorres/HoneyBadger. Accessed on May 1, 2023.

[12] Honeypot detector. https://honeypot.is/ethereum. Accessed on May 1, 2023.

[13] Keccak-256 hash function. https://monerodocs.org/cryptography/keccak-256/. Accessed on May 1, 2023.

[14] Logistic regression. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Accessed on May 1, 2023.

[15] Machine learning for fraud detection: fighting crime with algorithms. https://www.itransition.com/machine-learning/fraud-detection. Accessed on May 1, 2023.

[16] Optuna: A hyperparameter optimization framework. https://optuna.readthedocs.io/en/stable/. Accessed on May 1, 2023.

[17] Random forest. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Accessed on May 1, 2023.

[18] The rise of digital currencies and their potential impact on traditional banking. https://www.digipay.guru/blog/rise-of-digital-currencies-and-impact-on-traditional-banking/. Accessed on May 1, 2023.

[19] Rugpulldetector. http://rugpulldetector.com. Accessed on May 1, 2023.

[20] Support vector machine. https://scikit-learn.org/stable/modules/svm.html.

[21] Sushiswap docs. https://docs.sushi.com/. Accessed on May 1, 2023.

[22] Token sniffer. http://tokensniffer.com. Accessed on May 1, 2023.

[23] Top blockchain platforms 2023. https://hacken.io/discover/blockchain-platforms/. Accessed on May 1, 2023.

[24] Unicrypt - liquidity lockers. ”https://docs.uncx.network/guides/for-investors/liquidity-lockers/”. Accessed on May 1, 2023.

[25] Uniswap v2 events. https://docs.uniswap.org/contracts/v2/reference/smart-contracts/pair. Accessed on May 1, 2023.

[26] Uniswap whitepaper. https://uniswap.org/whitepaper.pdf. Accessed on May 1, 2023.

[27] What is a dex? https://www.coinbase.com/learn/crypto-basics/what-is-a-dex. Accessed on May 1, 2023.

[28] What is an order book? https://www.investopedia.com/terms/o/order-book.asp. Accessed on May 1, 2023.

[29] What is blockchain technology? https://www.ibm.com/topics/blockchain. Accessed on May 1, 2023.

[30] What is meant by turing-complete in ethereum? https://www.geeksforgeeks.org/what-is-meant-by-turing-complete-in-ethereum/. Accessed on May 1, 2023.

[31] What is yield farming? https://coinmarketcap.com/alexandria/article/what-is-yield-farming. Accessed on May 1, 2023.

[32] Xgboost. https://xgboost.readthedocs.io/en/stable/tutorials/model.html. Accessed on May 1, 2023.

[33] Nick Crow. James Olsen. Ben Bassa. Douglas Atherton. The journey of an ethereum transaction. https://mycelium.xyz/research/the-journey-of-an-ethereum-transaction. Accessed on March 1, 2023.

[34] Massimo Bartoletti, Salvatore Carta, Tiziana Cimoli, and Roberto Saia. Dissecting ponzi schemes on ethereum: identification, analysis, and impact. *Future Generation Computer Systems*, 102:259–277, 2020.

[35] Brenton Blanchet. Scammers behind ’squid game’. https://www.complex.com/pop-culture/squid-game-cryptocurrency-scammers. Accessed on May 1, 2023.

[36] Pascale Davies. Crypto will be universally adopted within 10 years and overtake traditional investments - new survey. https://www.euronews.com/next/2022/04/27/cryptos-will-be-univerally-adopted-within-10-years-and-overtake-traditional-investments
Accessed on May 1, 2023.

[37] Enzo Grossi and Massimo Buscema. Introduction to artificial neural networks. *European journal of gastroenterology & hepatology*, 19(12):1046–1054, 2007.

[38] Shafaq Naheed Khan, Faiza Loukil, Chirine Ghedira-Guegan, Elhadj Benkhelifa, and Anoud Bani-Hani. Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-peer Networking and Applications*, 14:2901–2925, 2021.

[39] Shaurya Malwa. Meme coin teddy doge soft rug pull. https://www.coindesk.com/tech/2022/07/25/memecoin-teddy-doge-soft-rug-pulls-45m-worth-of-tokens-peckshield-says/. Accessed on May 1, 2023.

[40] Bruno Mazorra, Victor Adan, and Vanesa Daza. Do not rug on me: Leveraging machine learning techniques for automated scam detection. *Mathematics*, 10(6):949, 2022.

[41] Gustavo A Oliva, Ahmed E Hassan, and Zhen Ming Jiang. An exploratory study of smart contracts in the ethereum blockchain platform. *Empirical Software Engineering*, 25:1864–1904, 2020.

[42] Mate Puljiz, Stjepan Begušic, and Zvonko Kostanjcar. Market microstructure and order book dynamics in cryptocurrency exchanges. In *Crypto Valley Conference on Blockchain Technology*, 2018.

[43] Stephen A Rhoades. The herfindahl-hirschman index. *Fed. Res. Bull.*, 79:188, 1993.

[44] Christof Ferreira Torres, Mathis Steichen, and Radu State. The art of the scam: Demystifying honeypots in ethereum smart contracts. *arXiv preprint arXiv:1902.06976*, 2019.

[45] Marie Vasek and Tyler Moore. There's no free lunch, even using bitcoin: Tracking the popularity and profits of virtual currency scams. In *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*, pages 44–61. Springer, 2015.

[46] Friedhelm Victor and Tanja Hagemann. Cryptocurrency pump and dump schemes: Quantification and detection. In *2019 International Conference on Data Mining Workshops (ICDMW)*, pages 244–251. IEEE, 2019.

[47] Yuanchao Wang, Z. Pan, J. Zheng, L. Qian, and Li Mingtao. A hybrid ensemble method for pulsar candidate classification. *Astrophysics and Space Science*, 364, 08 2019.

[48] Will Warren and Amir Bandeali. 0x: An open protocol for decentralized exchange on the ethereum blockchain. *URl: https://github. com/0xProject/whitepaper*, pages 04–18, 2017. Accessed on May 1, 2023.

[49] DR. GAVIN WOOD. Ethereum: A secure decentralised generalised transaction ledger. https://ethereum.github.io/yellowpaper/paper.pdf, 2022. Accessed on May 1, 2023.

[50] Pengcheng Xia, Haoyu Wang, Bingyu Gao, Weihang Su, Zhou Yu, Xiapu Luo, Chao Zhang, Xusheng Xiao, and Guoai Xu. Trade or trick? detecting and characterizing scam tokens on uniswap decentralized exchange. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 5(3):1–26, 2021.

Appendix A

Table with dataset features

Feature	**Description**
num_transactions	total number of transfers
n_unique_addresses	total number of unique addresses
cluster_coeff	clustering coefficient
tx_curve	HHI applied to each token
liq_curve	HHI applied to liquidity pool tokens
Mint	total number of mint events on Uniswap
Burn	total number of burn events on Uniswap
Transfer	total number of transfer events on Uniswap
difference_token_pool	no. of transactions between token and pool creation
n_syncs	total number of Sync Events on Uniswap
WETH	the total value of ETH existing in pool
prices	price of the token
liquidity	total liquidity
burn	1 if liquidity burned, 0 otherwise
lock	1 if liquidity locked, 0 otherwise
yield	1 if yield farming involved, 0 otherwise
contract_status	1 if the contract is verified, 0 otherwise
honeypot	1 if token is a honeypot, 0 otherwise
max_tx_amount	maximum tax amount that can be set
buy_tax	tax for buying the token
sell_tax	tax for selling the token
buy_gas	gas to be used for executing buy transaction
sell_gas	gas to be used for executing buy transaction
totalSupply	the total supply of the token
blueCheckmark	Etherscan blue checkmark (1 if verified, 0 if not)
description	Project description (1 if exists, 0 if not)
website	Website (1 if exists, 0 if not)
email	Email (1 if exists, 0 if not)
blog	Blog (1 if exists, 0 if not)
reddit	Subreddit (1 if exists, 0 if not)
slack	Slack (1 if exists, 0 if not)
facebook	Facebook page (1 if exists, 0 if not)
twitter	Twitter account (1 if exists, 0 if not)
bitcointalk	Bitcointalk discussion (1 if exists, 0 if not)
github	Github account (1 if exists, 0 if not)
telegram	Telegram channel (1 if exists, 0 if not)
wechat	Wechat (1 if exists, 0 if not)
linkedin	Linkedin organization (1 if exists, 0 if not)
discord	Discord channel (1 if exists, 0 if not)
whitepaper	Whitepaper (1 if exists, 0 if not)