CAPSTONE PROJECT

TECHNOLOGY PARK MALAYSIA

CT085-5-M-CP2
CAPSTONE PROJECT

HAND OUT DATE : 15TH JULY 2023

HAND IN DATE : 13TH MAY 2024 (EXTENDED)

WEIGHTAGE: 100%

SUPERVISOR NAME : MR. DHASON PADMAKUMAR

2ND MARKER NAME : MS. CHONG MIEN MAY

STUDENT NAME : TEOH KHENG HONG

STUDENT TP NUMBER : TP030562

STUDENT INTAKE : APDMP2201DSBA (DE)(PR)

PROGRAMME OF STUDY : MSC IN DATA SCIENCE AND BUSINESS ANALYTICS

PROJECT TITLE : IDENTIFYING FRAUDULENT ACTIVITY IN BLOCKCHAIN TRANSACTIONS USING MACHINE LEARNING ALGORITHMS

TABLE OF CONTENTS

Identifying Fraudulent Activity in Blockchain Transactions using Machine Learning Algorithms

ABSTRACT

Blockchain technology has unique characteristics that is implemented in different industries and applications. In financial markets, blockchain solutions are introduced for different purpose, while they may vary in implementation, the major components and principles are remained the same. However, the technology does not prevent the illicit activities made which tarnish the reputation of the blockchain technology. A lot of fraud detection techniques has been existing in the industry and machine learning algorithms are introduced to perform fraud detection on the blockchain transactions which currently growing in volume and complexity. A research study has been conducted to understand the features of patterns of the blockchain transactions that has been contributed to the community. The research has contributed to provide neural networks, especially deep neural networks to discover a new area in the dataset provide to compare with existing algorithms provided in the similar domain with following the performance metrics shared by the researchers. The results shown that neural networks can keep on par on the existing machine learning algorithms across multiple cross validation folds and the works have been given using hyperlinks to allow future research from the same implementation.

KEYWORDS

blockchain, machine learning, Ethereum, fraud detection, neural networks, cryptocurrency

CHAPTER 1: INTRODUCTION

Blockchain is widely recognized as a novel method for conducting transactions, notably with a white paper, which was the first decentralized peer-to-peer electronic cash system (Satoshi, 2008). The release of the paper has led to emergence of new financial systems, due to the fact not only it can be purely doing transactions but can be further made into other applications according to the needs. Blockchain

resolved the major problems of the digital technology that it provides the transparency, immutability, anonymity, auditability, consistency while decentralized throughout the network (Pandey, A. C., Verma, A., Rathor, V. S., Singh, M., & Singh, A. K. , 2023). The characteristics of blockchain has changed the way of how industries manage trusts, and reduce the obstacles that slow down business processes, such as banking and finance, medical records, logistics and others (Consensus Cloud Solutions Inc, 2023). Blockchain technologies are known in many areas that tells how disruptive the technology is, include:

- Finance: Cryptocurrencies, Non-Fungible Tokens (NFTs), digital assets etc.
- Healthcare: Electronic Health Records
- Mobile: Decentralized Finance (DeFi) mobile apps.
- Internet of Things (IoT): Better data exchange between devices with limited resources (Tran, N. K. , Babar, M. A. , Boan, J. , 2020).
- Supply Chain: Secure validation of material and supply flow from faulty product (Ahmad et al., 2022).

Blockchain has come a long way from its original implementation. Many industries and enterprises have invested into the technology to build their solutions, enhancing the current business model to next level. According to CoinMarketCap, there are over 24,000 cryptocurrencies, trading over 600 exchanges, with capitalization exceeds $1 trillion (CoinMarketCap, 2024). More than 60% of blockchain worldwide market value comes from financial sector (Petrov, 2023). The adoption of blockchain in financial transactions is increasing as the cost of transaction using blockchain are cheaper than transaction costs in the traditional economy, as it reduces security, auditing, 3rd party safeguarding and other related risks (Schmidt & Wagner, 2019). Some researchers found out that blockchain can reduce banks' infrastructure costs up to 30 % (Warren & Treat, 2019).

Not only that, but the financial industries also have advanced technologies to help them identify fraud transactions. As the transaction grows, many financial entities such as banks are regulating the flows to prevent large scale thefts that may jeopardise the reputation and economy of the customers. Daily transactions are voluminous and contains highly complex data structure that requires analysis to analyse the transaction, stored in different types of high availability, secure, production grade database clusters with

heavy data regulations. Big Data come into the picture with many tools that forms the pipelines that perform analysis in real time for multiple applications, such as identity verification, loan applications, background check and lot more. Our daily life is surrounded by transactions in different forms which signifies the importance of financial services in the daily operations. The advancement of technologies has led to new forms creativity and innovation in Fintech technology (Fintech), that create tense competition in the business and consumer markets, such as Buy Now Pay Later (BNPL), mobile banking, cryptocurrency, smart contracts and more. Artificial Intelligence/Machine Learning (AI/ML) are implemented to perform predictive behavioural analytics, and data-driven marketing, which result better financial services to the customers.

However, it's not without the problems and challenges, such as the crime activities that has been causing the huge loss to financial assets and the attacks on blockchain network that caused loss in reputation and value of assets. In crypto crime report as of year 2023, there's a $24.2 billion transaction worth of illicit activities which covers 0.34% of the total on-chain transaction volume (Chanalysis, 2024). In further analysis, the illicit activities were going down in percentage compared year on year (YoY), especially crypto scamming and hacking. Sanctioned entities drive most of the illicit activities of the blockchain. However, the illicit activities can be identified using blockchain analysis tools, which helps to prevent the transaction to proceed using machine learning and artificial intelligence.

To understand the fundamentals of blockchain, the research background of blockchain with underlying architecture are studied to identify the aim and objective of the project.

CHAPTER 2: LITERATURE REVIEW

2.1 BLOCKCHAIN

2.1.1 TYPES OF BLOCKCHAIN NETWORK

2.1.1.1 PUBLIC BLOCKCHAIN

A public blockchain fully embraces the concept of decentralization. Anyone can join and contribute transactions without encountering any restrictions or oversight by governing authorities. Transactions are entirely anonymous since participants are not required to disclose any information in the blocks they create. Public blockchains promote trustworthiness and security through continuous review by an open-source community as a default practice. However, this type of blockchain has certain drawbacks such as slower processing, high energy consumption, and reluctance from governments to adopt it expeditiously.

2.1.1.2 PRIVATE BLOCKCHAIN

Unlike public blockchains, private blockchains are designed for use within a single organization. They offer more control over the network, allowing for restrictions on who can participate in the network and who can view the transactions. This type of blockchain is faster and more efficient than public blockchains, making it suitable for enterprise use cases. However, the downside is that private blockchains are less transparent and may not be as secure due to the centralized control (Mohan, 2019).

2.1.1.3 CONSORTIUM BLOCKCHAIN

A consortium blockchain, also known as a federated blockchain, falls between the public and private blockchain models. It is a permissioned network where multiple organizations or entities share the responsibility of maintaining the network. Consortium blockchains are suitable for industries or sectors where collaboration and trust between a select group of participants are essential. For example, Hyperledger Fabric is a popular example of a consortium blockchain. It is an open-source enterprise-grade framework that enables the development of permissioned blockchain networks. Organizations can form a consortium and define the rules governing the network, such as consensus mechanisms, membership criteria, and access controls. Hyperledger Fabric is widely used in supply chain management, healthcare, and finance sectors for its versatility and adaptability (Yang et al., 2020).

2.1.2 COMPARISON BETWEEN BLOCKCHAIN NETWORK

ASPECT	PUBLIC BLOCKCHAIN	PRIVATE BLOCKCHAIN	CONSORTIUM BLOCKCHAIN
DECENTRALIZATION	Fully decentralized	Centralized within organization	Shared responsibility among entities
PARTICIPATION	Anyone can join and contribute	Restricted participation	Multiple organizations involved
TRANSPARENCY	Transactions are entirely anonymous	More control over network and data	Defined rules and access controls
EFFICIENCY	Slower processing speed	Faster and more efficient	Versatile and adaptable
SECURITY	Continuous review by open-source community	May not be as secure	Joint responsibility for network security
ENERGY CONSUMPTION	High	Lower	Moderate

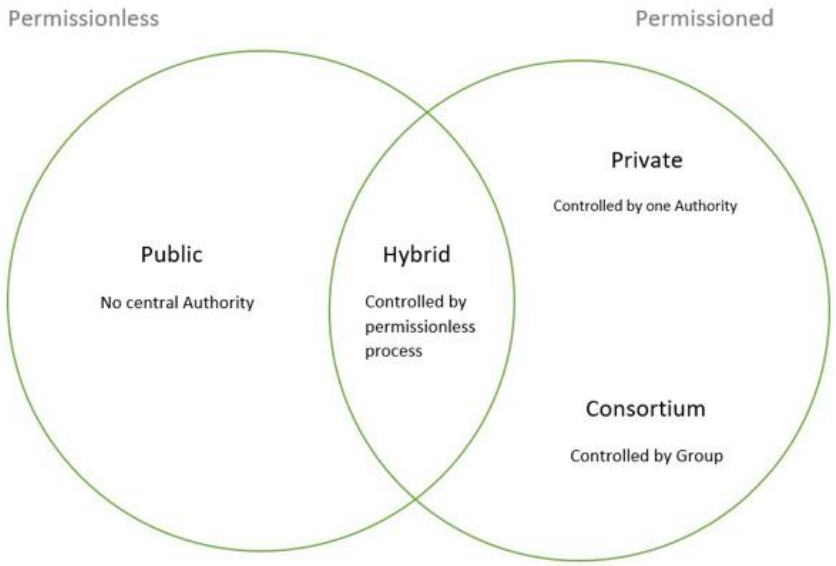


Figure 1: Types of Blockchain (GeeksforGeeks.org, 2022)

By comparing different blockchains, it is evident that each type has its own set of advantages and limitations. Public blockchains offer full decentralization and trustworthiness but suffer from slower processing and high energy consumption. On the other hand, private blockchains provide more control and efficiency but may lack transparency and security due to centralized control. Consortium blockchains strike a balance, allowing multiple organizations to collaborate and share network responsibilities, suited for use cases where trust and collaboration between specific entities are vital.

2.1.3 BLOCKCHAIN ARCHITECTURE

2.1.3.1 DISTRIBUTED LEDGER

To understand blockchain with a very simplified concept, just think it as a distributed ledger. A ledger in basic accounting is a book that records the transactions, and blockchain system behaves similarly in terms of how ledger works, but it's immutable and visible to everyone. Transactions includes purchases, sales, receipts and payments by an individuals or organizations. In here, the transactions are defined as blocks, each block contains one to many transactions and linked together, forming block-chain with each block contains the cryptographic identifiers of previous block.

2.1.3.2 CRYPTOGRAPHIC HASH FUNCTION

Cryptographic hash functions play a crucial role in securing the integrity of blockchain data. Every block in the chain contains a unique cryptographic hash of the previous block, creating a chain of interconnected blocks. This ensures that any alteration to a block will be immediately apparent, maintaining the security of the entire transaction history (Zhang, 2020).

2.1.3.3 CONSENSUS MECHANISM

Consensus mechanisms are used to achieve agreement on the state of the blockchain and the validity of transactions. They enable the network to reach a consensus without the need for a central authority. Proof of Work and Proof of Stake are some common consensus mechanisms used in blockchain networks (Sharma et al., 2020).

2.1.3.4 PROOF OF WORK (POW)

When a transaction is made, a validation is made to ensure the transaction is genuine. This is important as there is double spending, 51% attack, sybil attack and lot more. So, a challenge is made to ensure that the blockchain node resolve difficult mathematical puzzles to find the correct

answer first. This challenge requires to add a guess number (nonce) as input of hashing function. The difficulty of the challenge will either increase or decrease every time when a node is joins or leaves the network. The first node that founds the answer will spread to network and earns a block reward, for example a few numbers of Bitcoin which has huge monetary value (Albertorio, 2020).

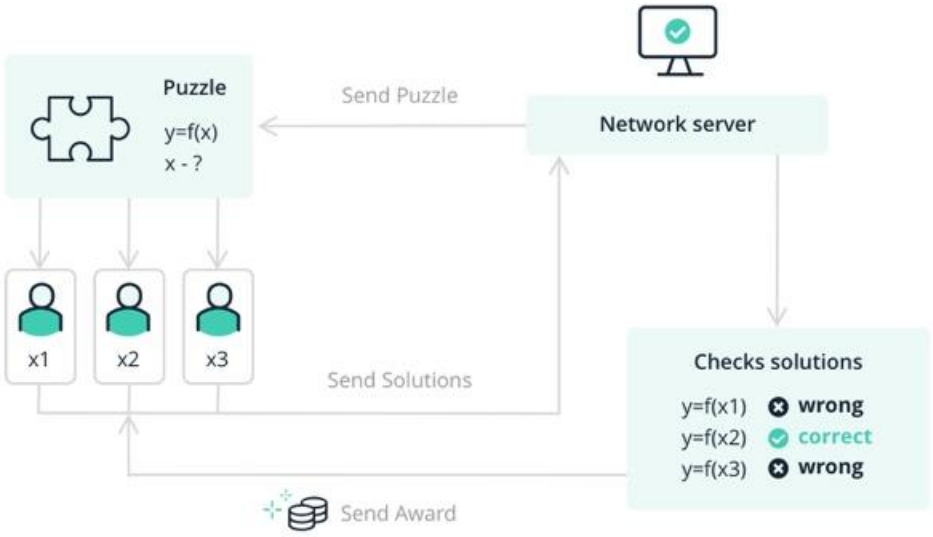


Figure 2: Proof of Work Process (Moreland, 2023)

2.1.3.5 PROOF OF STAKE (POS)

As the network grow large, PoW challenge has become difficult for the miners to resolve as energy efficiency is low For Proof of Stake (PoS), several validators are selected to validate the transaction, instead of fighting to resolve the math problem (Frankenfield, 2023). The table below shows the difference between PoW and PoS:

Proof of Stake	Proof of Work
Block creators are called validators	Block creators are called miners
Participants must own coins or tokens to become a validator	Participants must buy equipment and energy to become a miner
Energy efficient	Not energy efficient
Security through community control	Robust security due to expensive upfront requirement
Validators receive transactions fees as rewards	Miners receive block rewards

Figure 3: Difference between PoW and PoS (Frankenfield, 2023)

In summary, the consensus mechanism is implemented in different blockchain solution to ensure the balance of energy usage, security, and design.

2.1.3.6 SMART CONTRACTS

Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They automatically enforce the terms of the agreement when certain predefined conditions are met. Smart contracts enhance the efficiency and security of transactions in blockchain systems while reducing the need for intermediaries (Zhang, 2020).

2.1.4 BLOCK DATA STRUCTURE

In general, the data structure of blockchain contains the following:

- **Block Headers**: Metadata that helps to identify the block itself in the network. As blockchain network expands, some basic information is needed in these headers, which are:
    - **Previous Block Hash**: A hash value that is resulted from the previous block is made with using a cryptographic function of SHA 256, output a string of 32 bytes.
    - **Merkle Root**: A simple way to verify the block data is whole, undamaged and unaltered (Frankenfield, What Is a Merkle Root (Cryptocurrency)? How It Works in Blockchain, 2021).
    - **Timestamp**: The time when the current block is created.
    - **Difficulty Target**: The difficulty of the Proof-of-Work algorithm of the block. The network determines the

difficulty of how the nonce is to be resolved.

  - **Nonce**: A random number that the participants are trying to resolve to receive the block reward, which is a part of the **mining** process (Gülen, 2022).

(Antonopoulos, 2014)

- **Block Height**: Another way to identify the block itself in the blockchain. There may have other blocks that have the same block height value that compete for the same position in blockchain.
- **Genesis Block**: The first block in blockchain.
- **Merkle Trees**: A hash tree that each node is labelled with hash value of a block. Most of the Merkle trees are binary, and it allows quick and secure content verification across big blockchain network (Ravikiran A S, 2023).

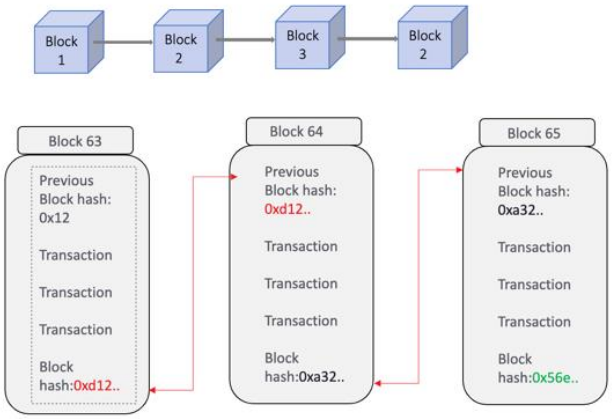


Figure 4: How each block looks like and how they are chained together (R3, 2023).

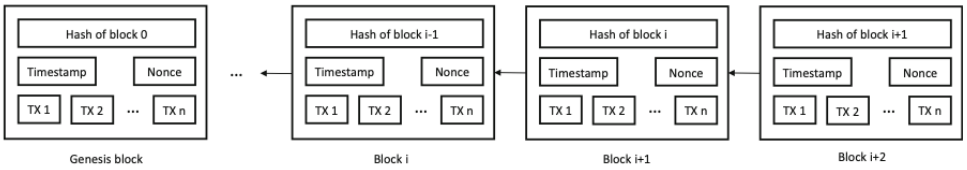


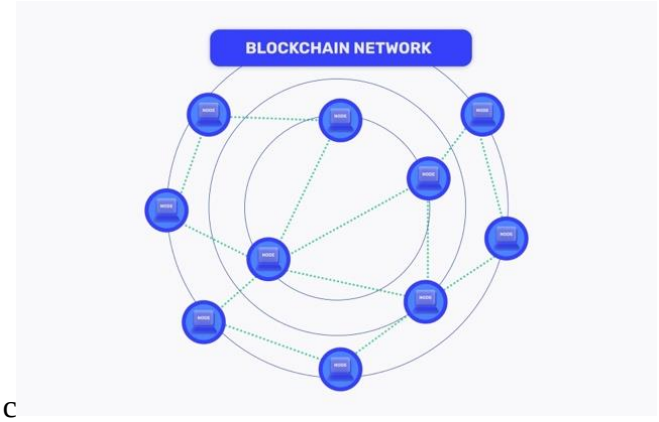
Figure 5: 2nd Example of a Blockchain (Zheng et al., 2018)

Figure 6:Visual Representation of the Block Data Structure (Narayanan, 2015)

2.1.5 HARDWARE

A node is a server that has configured to join into blockchain network for processing purpose, with meeting minimum requirements of hardware. The solution can be custom made by participants who have

joined into the Blockchain system to contribute into the process itself. A node in the system can be virtual (virtual machines) or physical (self-own) that may have partial or full copy of the transaction throughout the network. When adding the node to the Blockchain, it must be compliant with the algorithms and mechanisms of the network.



c

Figure 7: Decentralized, Blockchain Network, made up using a collection of computers run by individuals, with compliance with Blockchain rules. (Cheng, 2021)

2.1.6 APPLICATIONS

2.1.6.1 SUPPLY CHAIN MANAGEMENT

Blockchain technology can significantly improve transparency and traceability in supply chain management. By recording the journey of products from their source to the end consumer, blockchain ensures the authenticity of goods, reduces fraud, and enhances efficiency in supply chain operations (Zhang et al., 2021).

2.1.6.2 FINANCIAL SERVICES

In the financial sector, blockchain is widely used for secure and transparent transactions, cross-border payments, and digital asset management. It eliminates the need for intermediaries, reduces transaction costs, and facilitates faster and more secure financial transactions (Lu, 2018).

2.1.6.3 HEALTHCARE

Blockchain technology has the potential to revolutionize healthcare by securely storing and managing patient records, enabling interoperability among different healthcare providers, and ensuring the integrity and privacy of sensitive medical data (Guo & Liang, 2016).

2.1.6.4 VOTING SYSTEMS

The immutability and transparency of blockchain make it suitable for creating secure and tamper-proof voting systems. By leveraging blockchain technology, elections and voting processes can become

more secure, transparent, and resistant to fraud (Abuidris et al., 2019).

2.1.6.5 IDENTITY MANAGEMENT

Blockchain serves as a secure and decentralized solution for identity management, offering individuals control over their personal data and mitigating the risks associated with identity theft and fraud (Ma et al., 2021).

2.1.6.6 INTELLECTUAL PROPERTY PROTECTION

Blockchain technology can be utilized to create a transparent and secure system for managing and protecting intellectual property rights, such as patents, trademarks, and copyrights (Yang et al., 2019).

2.1.7 BLOCKCHAIN ATTACKS

While blockchain technology offers robust security features, it is not immune to attacks. Various forms of attacks can compromise the integrity and functionality of a blockchain network, posing significant risks to the stored data and the transactions processed within the system.

2.1.7.1 51% ATTACK

A 51% attack occurs when a single entity or a group of collaborating entities controls more than half of the mining power on a blockchain network. This enables them to manipulate transaction history, reverse transactions, and double-spend coins. Such attacks undermine the decentralization and trustworthiness of the network (Sayeed & Marco-Gisbert, 2019).

2.1.7.2 SYBIL ATTACK

In a Sybil attack, a malicious actor creates multiple false identities to gain control or influence over a network. This can be particularly problematic in public blockchain networks where nodes are anonymous, as the attacker can disrupt the consensus mechanism and manipulate transaction validation (Chow & Ma, 2021).

2.1.7.3 ECLIPSE ATTACK

An eclipse attack aims to isolate a specific node or a group of nodes from the rest of the network by controlling the information they receive. This can lead to manipulation of the data and disruption of the network's normal operation (Staff, 2020).

2.1.7.4 DENIAL OF SERVICE ATTACK

Denial of Service attacks can also target blockchain networks, aiming to disrupt their normal functioning by overwhelming them with a high volume of traffic or transactions. This can lead to network

slowdowns, transaction delays, and in severe cases, complete network unavailability (Wu et al., 2024).

2.1.7.5 DOUBLE SPEND ATTACK

A double spend attack involves spending the same cryptocurrency more than once. By exploiting vulnerabilities in a blockchain's consensus mechanism, an attacker can fraudulently spend the same funds in multiple transactions, undermining the integrity of the transaction ledger (Peng & Chen, 2018).

2.1.8 FRAUD CATEGORIES

There are many types of fraud occurred in blockchain transaction across different cryptocurrencies, depending on the solution type. A few of types of frauds that are most mentioned will be discussed to understand how the scams are identified.

2.1.8.1 PUMP-AND-DUMP SCHEMES

Pump-and-dump schemes involve artificially inflating the value of a cryptocurrency through false or misleading statements, then selling off the inflated assets to unsuspecting investors. This fraudulent scheme manipulates market prices, leading to financial losses for investors who are left holding devalued assets once the manipulators cash out (Gandal et al., 2018).

2.1.8.2 PHISHING

Phishing scams target users by tricking them into revealing their private keys, seed phrases, or login credentials for cryptocurrency wallets and exchanges. Once obtained, these sensitive details enable fraudsters to gain unauthorized access to users' funds, leading to theft and fraudulent transactions.

2.1.8.3 SMART CONTRACT VULNERABILITIES

Smart contracts, while innovative, are susceptible to vulnerabilities that can be exploited by fraudsters. Flaws in smart contract code can be manipulated to execute unauthorized or unintended transactions, leading to financial losses and breaches of trust. Smart contract vulnerabilities can occur due to various factors, including errors in the contract code, misconfigured permissions, and unforeseen interactions with other smart contracts. These vulnerabilities are often targeted by malicious actors to exploit the weaknesses in the code and execute unauthorized transactions. One example of a smart contract vulnerability is the re-entrancy attack. In 2016, the decentralized autonomous organization experienced a major security breach due to a re-entrancy

vulnerability in its smart contract code. This vulnerability allowed an attacker to repeatedly withdraw funds from the DAO before the transactions were properly recorded, resulting in a substantial financial loss (Daian, 2016). Another common vulnerability is the lack of proper input validation, which can lead to unexpected or malicious inputs causing the smart contract to behave in unintended ways. For example, if a smart contract does not adequately validate user inputs, it may be susceptible to manipulation through input data that exploits vulnerabilities in the contract's logic (Daian, 2016). Furthermore, the use of outdated or insecure programming languages and libraries in smart contract development can introduce vulnerabilities that attackers can exploit. For instance, the infamous Parity multi-sig wallet bug in 2017 resulted in the loss of a significant amount of Ether due to a vulnerability in the smart contract code that was inadvertently triggered by a user (Trace, 2021). These examples illustrate the critical importance of thoroughly auditing and testing smart contracts to identify and address vulnerabilities before deployment. It is essential for developers and organizations utilizing smart contracts to stay informed about potential risks and continuously update their security practices to mitigate the ever-evolving threats in the blockchain space.

2.1.8.4 INSIDER ATTACKS

Insider trading in blockchain networks involves the illegal trading of cryptocurrencies based on non-public, confidential information. This activity undermines the fairness and integrity of the market, providing unfair advantages to those with privileged information. According to Félez-Viñas et al. (2022), insider trading in cryptocurrency markets occurs when individuals use private information to buy coins prior to exchange listing announcements. The typical pattern involves an individual buying cryptocurrencies in advance of the public release of news that a coin is going to be listed on a significant exchange, which often leads to a price surge that follows the announcement (Félez-Viñas et al., 2022).

2.1.8.5 PONZI SCHEMES

Ponzi schemes are fraudulent investment schemes that promise high returns with little or no risk to investors. These schemes use funds from new investors to pay returns to earlier investors, creating the illusion of a profitable investment. As the scheme grows, it relies on a continuous influx of new investors to sustain the high returns, eventually collapsing when new investors can no longer be recruited, or existing investors demand their funds. Ponzi schemes in the context of blockchain

technology often misuse the hype around cryptocurrencies and blockchain projects to create fraudulent investment opportunities, deceiving individuals with promises of unrealistic returns (Bartoletti et al., 2020)

2.2 ETHEREUM

Ethereum is a decentralized platform that enables developers to build and deploy smart contracts and decentralized applications. Unlike Bitcoin, which is primarily a digital currency, Ethereum's blockchain serves as a programmable platform, allowing for the creation of self-executing contracts and applications. Ethereum's native cryptocurrency, Ether, is used to pay for transaction fees and computational services on the network (Devi et al., 2021).

One of Ethereum's distinguishing features is its use of smart contracts, which are self-executing contracts with the terms of the agreement directly written into code. These smart contracts automatically enforce and execute the terms of an agreement when predefined conditions are met, providing a transparent and tamper-resistant way to automate various processes.

Ethereum also introduced the concept of decentralized autonomous organizations, which are organizations represented by rules encoded as a computer program that is transparent, controlled by the organization members, and can manage and distribute funds without the need for centralized control.

2.2.1 ETHEREUM VIRTUAL MACHINE (EVM)

Ethereum Virtual Machine (EVM) plays a pivotal role in enabling the execution of smart contracts and decentralized applications on the Ethereum network. EVM is a Turing-complete virtual machine that enables the execution of smart contracts and decentralized applications on the Ethereum network. In simple meaning, EVM can perform any computation that a conventional computer can, allowing for the development and deployment of a wide range of complex applications on the Ethereum network. Moreover, EVM processes and executes smart contracts, which are coded in high-level languages like Solidity, and compiles them into bytecode that can be run on the EVM. Once deployed on the Ethereum network, these smart contracts autonomously execute their predefined functions and operations when triggered by specified conditions. Not only that, EVM also enables the execution of decentralized applications. DApps run on the Ethereum network, utilizing the EVM to execute code and interact with smart contracts, providing a decentralized and transparent environment for various applications, such as finance, gaming, and governance (Devi et al., 2021).

2.2.2 GAS AND TRANSACTIONS

To facilitate the resource allocation and execution of operations on the EVM, Ethereum uses a concept called "gas." Gas is the unit for measuring the computational effort required to perform operations on the network. Each operation on the EVM consumes a specific amount of gas, and users need to pay for these operations by using Ether, the native cryptocurrency of the Ethereum network.

When users initiate transactions or deploy smart contracts, they specify a gas limit and a gas price. The gas limit defines the maximum amount of gas they are willing to consume, while the gas price determines the cost per unit of gas. Miners on the Ethereum network process these transactions and smart contract deployments and are compensated with the gas fees paid by users.

2.2.3 ECOSYSTEM

Ethereum's support for tokens has led to the creation of a vast ecosystem of digital assets and decentralized finance applications, allowing for the issuance and management of various types of tokens, including security tokens and utility tokens.

2.2.3.1 DECENTRALIZED APPLICATIONS (DAPPS)

Ethereum's platform has been instrumental in the development of decentralized applications or DApps. These applications operate on a peer-to-peer network rather than a centralized server, enabling various functionalities such as decentralized finance, gaming, marketplaces, and social networking on the blockchain. Notable examples of DApps built on Ethereum include Uniswap, a decentralized cryptocurrency exchange, and Compound, a protocol for algorithmic, efficient money markets.

2.2.3.2 INITIAL COIN OFFERINGS (ICO)

Ethereum's smart contract functionality has been utilized for the creation and execution of Initial Coin Offerings – a fundraising method in which new cryptocurrencies or tokens are sold to early investors. ICOs have facilitated the launch of numerous blockchain projects and startups, making Ethereum a prominent platform for token creation and distribution.

2.2.3.3 STABLECOINS

Through Ethereum's infrastructure, stablecoins have gained popularity as digital assets pegged to a stable value, often a fiat currency like the US dollar.

Stablecoins provide stability in the volatile cryptocurrency market and are widely used for trading and as a store of value within the ecosystem. Examples of stablecoins built on Ethereum include Tether and USD Coin.

2.2.3.4 DECENTRALIZED FINANCE (DEFI)

Ethereum has played a pivotal role in the rise of Decentralized Finance – a movement aimed at creating open and permissionless financial services accessible to anyone with an internet connection. DeFi protocols built on Ethereum enable activities such as lending, borrowing, yield farming, and decentralized exchanges, revolutionizing traditional finance through blockchain technology.

Ethereum's impact extends beyond these specific examples, with its versatile ecosystem fostering innovation and reshaping the landscape of digital finance and decentralized applications. From smart contracts and tokens to groundbreaking applications, Ethereum continues to serve as a catalyst for the advancement of blockchain technology.

By harnessing the potential of Ethereum's ecosystem, developers and entrepreneurs can pioneer transformative solutions across diverse industries, further solidifying Ethereum's position as a leading platform for decentralized innovation.

With its robust infrastructure and emphasis on programmability, Ethereum has become a prominent platform for innovation and development across diverse domains, including finance, gaming, supply chain management, and identity verification. Its flexibility and extensibility make it an attractive platform for building next-generation decentralized applications with various use cases.

2.2.4 TYPES OF ETHEREUM ACCOUNTS

2.2.4.1 EXTERNALLY OWNED ACCOUNTS (EOA)

In Ethereum, there are two types of accounts: Externally Owned Accounts (EOA) and Contract Accounts (CA). EOA are controlled by private keys and external to the Ethereum network, meaning they are owned and operated by individuals. These accounts are used for initiating transactions, deploying smart contracts, and interacting with decentralized applications. The control of an EOA is entirely in the hands of the private key holder, providing a high degree of personal control and security.

2.2.4.2 CONTRACT ACCOUNTS (CA)

Contract Accounts (CA), on the other hand, are controlled by their code rather than a private key. They are responsible for executing smart contracts and managing their state. These accounts are created when a smart contract is deployed on the Ethereum network. The code within a Contract Account can define rules, store data, and interact with other smart contracts and EOAs. Contract Accounts enable the decentralized execution of operations and the establishment of automated processes without the need for human intervention.

These two types of accounts play distinct roles in the Ethereum ecosystem, with Externally Owned Accounts serving as the conduits for human interaction and Contract Accounts facilitating the execution of automated processes and smart contracts. Both account types contribute to the programmable nature of the Ethereum blockchain, providing the foundation for a wide range of decentralized applications and financial instruments (Introduction to smart contracts, 2023).

2.2.5 TYPES OF ETHEREUM TOKENS

2.2.5.1 UTILITY TOKENS

Utility tokens are designed to provide access to a particular product or service within a specific platform. They are often used to facilitate transactions and interactions within decentralized applications running on the Ethereum network. Utility tokens hold intrinsic value within the ecosystem they operate in and can be utilized for various purposes such as accessing premium features, paying for services, or participating in governance mechanisms of DApps (Council, 2020).

2.2.5.2 SECURITY TOKENS

Security tokens represent ownership of real-world assets, such as equity in a company, debt, or revenue share. These tokens are typically subject to regulations and legal compliance and are designed to offer investors financial interests or dividends. Security tokens on the Ethereum blockchain are governed by smart contracts that enforce regulatory compliance and enable automated distribution of dividends or other financial instruments (Timachevsky, 2018).

2.2.5.3 NON-FUNGIBLE TOKENS (NFT)

Non-Fungible Tokens, or NFTs, are unique digital assets that represent ownership or proof of authenticity of a specific item or piece of content. They are indivisible and not interchangeable, making each NFT distinct and irreplaceable. NFTs have gained prominence for their applications in digital art, collectibles, virtual real estate, and in-game assets within the Ethereum ecosystem. Smart contracts governing NFTs ensure scarcity, provenance, and ownership rights, enabling creators and collectors to trade and transfer unique digital assets with cryptographic authenticity (ERC-721 Non-Fungible Token Standard, 2023).

2.2.5.4 ERC-20 TOKENS

ERC-20 is a widely adopted standard for fungible tokens on the Ethereum network. These tokens are interchangeable and can be subdivided into smaller units, making them suitable for representing and exchanging various forms of value. ERC-20 tokens have become the foundation for numerous initial coin offerings, crowdfunding campaigns, and the creation of tokenized assets. They adhere to a set of predefined functions, including transfer, approval, and allowance, ensuring interoperability and compatibility with a wide range of Ethereum-based services and decentralized applications(ERC-20 Token Standard, 2023) .

2.3 RELATED WORKS

2.3.1 MAIN RESEARCH

With the base knowledge on the domain, the data scientist has reviewed on literature on the domain that has similar background and analyse the techniques that the researchers are using.

The emergence of blockchain technology and cryptocurrencies has precipitated novel challenges in the detection and prevention of illegal financial activities. Considering these developments, the work by Farrugia et al. (2020) represents a significant contribution to the domain of cryptocurrency fraud detection. The researchers address the pressing issue of illicit transactions on the Ethereum blockchain by employing machine learning techniques to distinguish between legal and illegal account activities.

Based on the study, Farrugia et al. (2020) believes that the important features or patterns that the Ethereum transactions that the illicit and normal accounts that involved in have the following:

- **Interaction and Transaction Values**: The amount of the Transactions made between parties.

- **Node Degrees Features**: The mean node degree and variance of node degree with other users with which transactions were executed are identified as important features. Node degree here relates to the number of transactions an account has with different users.
- **Transaction Amount Variance**: Some clusters showed a large variance in transaction amounts, indicating accounts that participate in transactions of widely varying values, which is sometimes a sign of illicit activity.
- **Smart Contract Features**: For smart contracts suspected of being Ponzi schemes, the study classifies based on account features, code features, and a combination of both. Opcode usage in these contracts is also examined.
- **Successful Transactions**: Only transactions that were successfully executed (determined by the 'ISERROR' field in the transaction data being set to 0) are considered for further analysis, excluding unsuccessful transactions.
- **Illicit Identification Markers**: Transactions involving fake ICOs, phishing, and other scams are analysed as identified by community-maintained lists of known scam addresses and activities.

How the Illicit Identification Markers are created? according to Farrugia et al. (2020), According to Farrugia et al. (2020), the Ethereum accounts that are flagged for illicit activities with the following pattern:

- Imitate other contract addresses.
- Scam lotteries.
- Fake ICOs
- Imitating other users
- Ponzi schemes
- Phishing
- Mirroring websites

These accounts are flagged along with their labelled transactions. Farrugia et al. (2020) curated a dataset comprising 2,179 accounts flagged for questionable activities by the Ethereum community, in contrast to 2,502 normal accounts, to underpin their analysis. Through meticulous data compilation and applying the XGBoost classifier, a machine learning algorithm renowned for its predictive proficiency, the researchers achieved a noteworthy average accuracy of 96.3% with an AUC of 0.994 in identifying illicit accounts.

Three features were underscored as particularly influential in the predictive model: the time interval between the first and last transactions, the total Ether balance,

and the minimum value received. These findings underscore the nuanced behavioural patterns that differentiate illicit transactions from typical network activities.

The study by Farrugia (2020) and his colleagues are integral to the ongoing discourse on the applicability of machine learning in financial security, offering a robust methodological framework and a public benchmark dataset for future research endeavours in the field. The high degree of accuracy achieved by their model sets a precedent for subsequent research and practical applications aimed at fortifying the integrity of the Ethereum blockchain against fraud (Farrugia et al., 2020).

Table 1: Metrics of XGBoost in n fold of cross validation (Farrugia et al., 2020).

# of Folds	***Optimal Depth***	***Optimal # estimators***	***Accuracy***	***FI***	***AUC***	***Execution time (s)***
3	4	300	0.960 (+- 0.002)	0.957 (+- 0.002)	0.993 (+- 0.0005)	62.03
4	4	200	0.960 (+- 0.002)	0.957 (+- 0.002)	0.993 (+- 0.0003)	91.13
5	3	250	0.962 (+- 0.004)	0.959 (+- 0.005)	0.994 (+- 0.0008)	105.4
10	4	150	0.963 (+- 0.006)	0.960 (+- 0.006)	0.994 (E 0.0007)	230.6

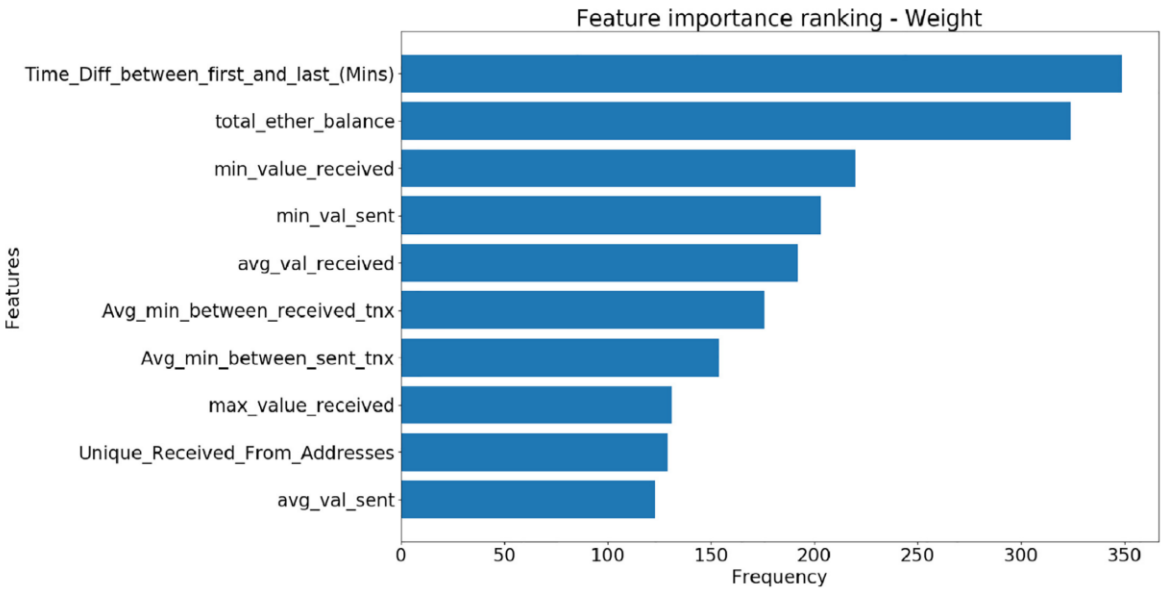


Figure 8: Feature Importance Ranking from the dataset (Farrugia et al., 2020).

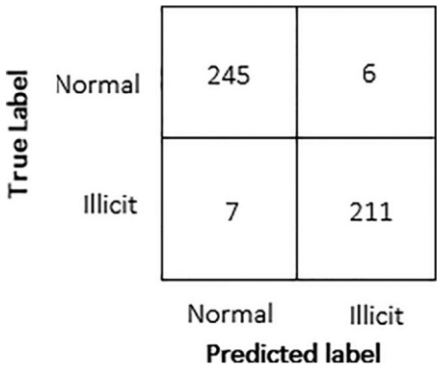


Figure 9: Confusion Matrix of Proposed XGBoost algorithm (Farrugia et al., 2020).

In addressing the prevalent issue of fraudulent activities within blockchain networks, the study by Sallam et al. presents a pioneering approach towards identifying fraudulent accounts on the Ethereum blockchain. Recognizing the inherent challenges posed by the pseudo-anonymity of blockchain systems, Sallam et al. employed a variety of machine learning techniques to scrutinize and classify accounts as either fraudulent or legitimate. The research leveraged a substantial dataset of 4,681 instances, encompassing both fraudulent and regular accounts, to train and test three distinct machine learning models: K-Nearest Neighbour, Random Forest, and XGBoost (Sallam et al., 2022).

The results from Sallam et al.'s (2022) study are particularly notable; the XGBoost model yielded an impressive average

accuracy of 96.80%, along with an average Area Under the Curve of 0.995. Random Forest and K-Nearest Neighbour models also demonstrated robust performance with average accuracies of 94.88% and 87.85%, respectively. Not only do these results underscore the high efficacy of ensemble learning techniques in fraud detection on Ethereum, but they also mark a significant step towards the automated and intelligent surveillance of blockchain transactions to pre-emptively detect and deter fraudulent actors.

This paper's findings are pivotal for the advancement of security measures in decentralized financial platforms, providing insights into the prospects of machine learning in enhancing the safety and integrity of the Ethereum network. It stands as a critical reference in the expanding corpus of literature focused on the confluence of machine learning and blockchain technology for fraud detection. The methodologies and outcomes articulated by Sallam et al. (2022) are of considerable relevance to this research, principally due to the parallels in objectives and the potential applicability of similar machine learning techniques to other blockchain environments.

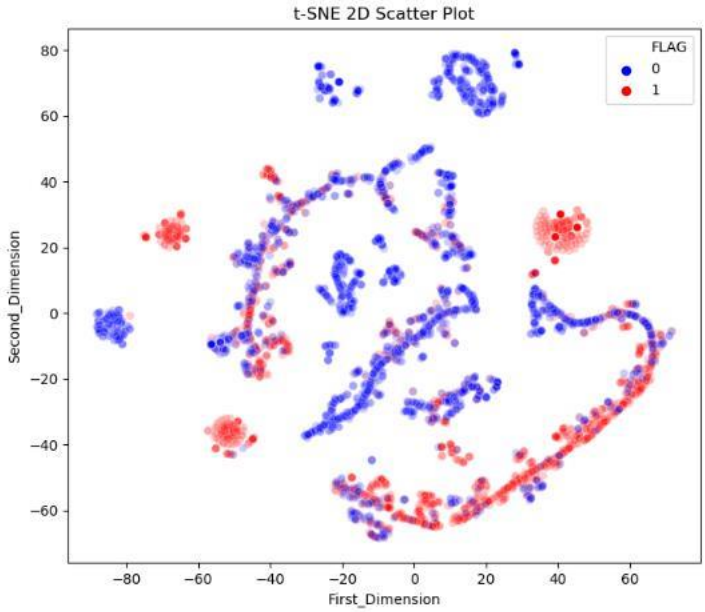


Figure 10: 2D scatter plot of the dataset using t-SNE (Sallam et al., 2022).

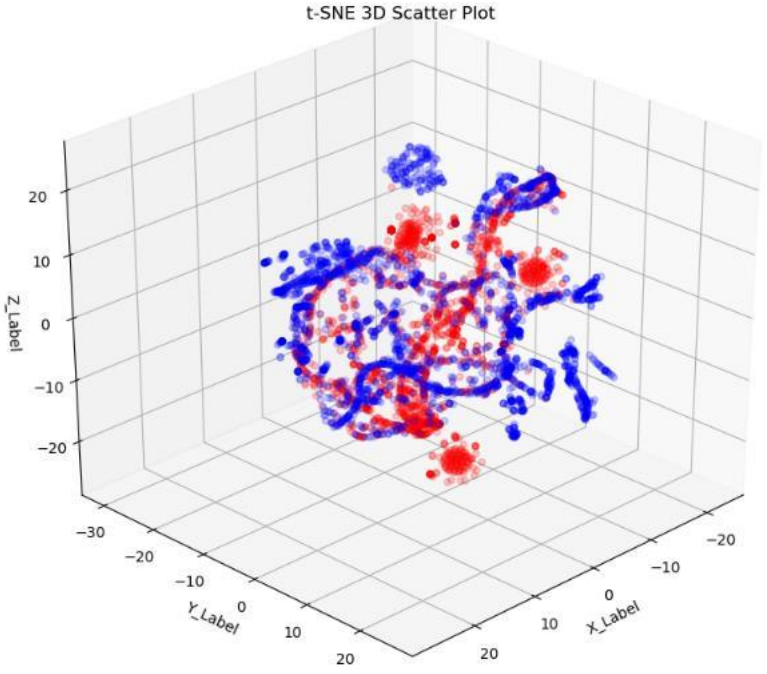


Figure 11: 3D scatter plot of the dataset using t-SNE (Sallam et al., 2022).

The figures above that shows different classes of Ethereum account, the researchers found that there are a few distinguishable clusters of blues and reds, but some clusters are not linearly possible. This signifies the importance of Machine Learning algorithms to find the significant features among account clusters.

Table 2: Evaluation of the machine algorithm models (Sallam et al., 2022).

Model Name	***Accuracy (%)***	***Precision (%)***	***Recall (%)***	***F1-score (%)***	***Execution Time (Sec.)***
XGBoost	96.8	96.5	96	96.5	1.232
Random Forest	94.88	95	94.5	95	3.268
K-Nearest Neighbour	87.85	88.00	87.5	88	0.206

Table 3: Hyperparameters of machine learning algorithms (Sallam et al., 2022).

CLASSIFIER	**HYPERPARAMETERS**
XGBOOST	Maximum Tree Depth = 3 Number of Trees = 200
RANDOM FOREST	N Estimators = 600 Min_Samples_Split = 5 Min Samples Leaf = 2
K-NEAREST NEIGHBOUR	K =6

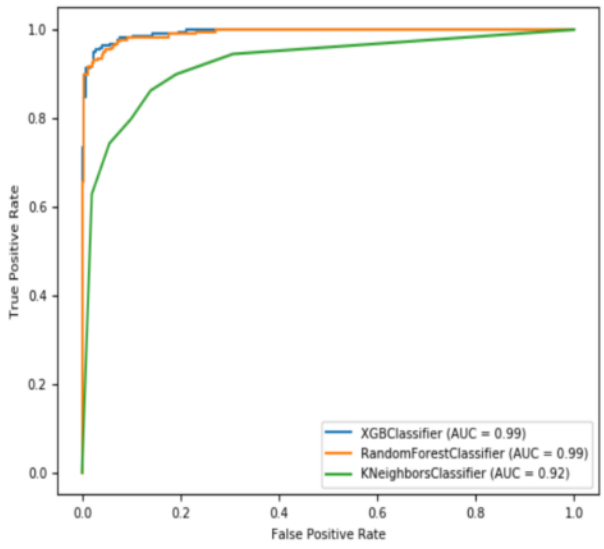


Figure 12: AUC performance of different machine algorithm models (Sallam et al., 2022).

K-nearest neighbour is an unsupervised machine learning algorithm that is belonged to instance-based learning, by classifying a new sample based on the class labels of its k nearest neighbours in the training data (IBM, 2023). It's simple and intuitive, but it may have sensitive towards irrelevant

features and high computational cost for the large datasets (IBM, 2023). When dealing with the dataset produced by Farrugia et al. (2020), the performance metrics are falling behind the other models that are under supervised machine learning category, while XGBoost might be better than Random Forest in all performance aspects listed, with the given hyperparameters.

In recent advancements within the domain of blockchain security, the study by Norouzi (year) presents a pioneering approach to enhancing the detection of fraudulent accounts on the Ethereum blockchain. This research stands out by introducing a novel feature selection method that synergizes Recursive Feature Elimination Cross-Validation with Random Forest Feature Importance to isolate the most pivotal features for fraud detection. Significantly, the investigation employed two datasets, one balanced and the other imbalanced, comprising normal and illicit accounts to assess various supervised machine learning models including NuSVC, Logistic Regression (LR), Random Forest (RF), and Multilayer Perceptron (MLP). Norouzi's findings underscore the exceptional performance of the Random Forest model, achieving an impressive average AUC score of 0.998. This was closely followed by the Multilayer Perceptron classifier with an average AUC of 0.995. These results highlight not only the high accuracy but also the excellent generalization capabilities of the models in identifying illicit activities. Furthermore, this work provides insightful analysis into the behaviour of fraudulent accounts, such as a noticeably shorter average usage duration compared to legitimate accounts, suggesting a pattern of creation solely for nefarious activities. This empirical evidence positions Norouzi's study as a critical reference for researchers seeking effective and computationally efficient methods for fraud detection within the Ethereum network (Norouzi, 2023).

Table 4: Performance of Machine Learning Models in Detecting Fraudulent Accounts on Ethereum (Norouzi, 2023)

Model	Feature Set	Accuracy	F1-Score	AUC	Execution Time (s)
RF	42 Features	0.978	0.978	0.998	8.482
RF	16 Features	0.979	0.979	0.998	2.88
RF	6 Features	0.972	0.972	0.996	0.976
MLP	42 Features	0.964	0.964	0.995	341.02
MLP	16 Features	0.975	0.975	0.987	334.759
MLP	6 Features	0.955	0.955	0.98	274.12

Multi-Layer Perceptron (MLP) is a type of artificial neural network that is commonly used for classification tasks. It consists of multiple layers of interconnected nodes, where each node in one layer is connected to every node in the next layer. The first layer is the input layer, which receives the data to be classified, and the last layer is the output layer, which produces the classification result. The layers in between are called hidden layers, and they perform computations on the input data to extract features that are useful for classification. MLPs are trained using a supervised learning algorithm called backpropagation, which adjusts the weights of the connections between the nodes to minimize the difference between the predicted output and the actual output.

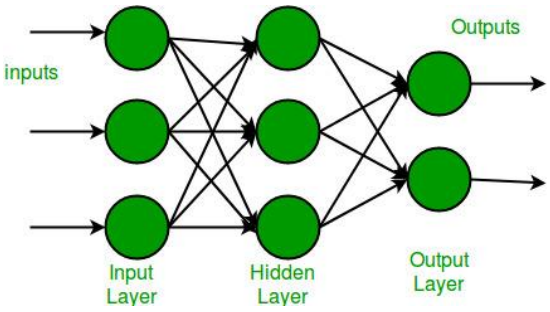


Figure 13: Multi-Layer Perceptron (MLP) Schematic Diagram (geeksforgeeks, 2023)

Next, in addressing the prevalent challenge of fraud detection within the blockchain environment, the study by Taher et al. (2024) represents a significant contribution. The researchers focused on the Ethereum network, employing various machine learning techniques to detect fraudulent transactions. Their innovative use of a hard voting ensemble model reports an impressive 99% accuracy, showcasing the

potential of ensemble learning in this context (Taher et al., 2024). Their methodology leverages ensemble learning, combining the predictions of multiple machine learning models such as Random Forest, AdaBoost, and Decision Tree, and integrates explainable AI to provide insights into the models' decision processes, thereby addressing the need for transparency in AI-based fraud detection systems. This approach aligns with current trends in the field that call for enhanced interpretability alongside predictive performance in anti-fraud systems. Furthermore, the application of ensemble learning to the specific challenges of smart contract transactions within Ethereum adds a valuable dimension to the literature, as it reflects a targeted response to the complexities of blockchain-based financial activities. Taher et al.'s research not only advances the technical arsenal available for fraud detection but also provides a framework for future studies to build upon, emphasizing the critical role of data pre-processing and the inventive use of resampling techniques to balance datasets. The paper serves as a benchmark for my research into blockchain security, informing both the selection of machine learning techniques and the integration of transparency mechanisms vital for user trust (Taher et al., 2024).

Table 5: Ethereum fraud detection algorithms results (Taher et al., 2024). PREC = Precision, REC = Recall, F-S = F1 Score.

CLASSIFIER	ACC	TRAINING TIME (S)	LABEL	PREC	REC	F-S
RF	98.70%	3.35	Fraudulent	0.99	0.99	0.99
			Legitimate	0.99	0.99	0.99
DT	97.30%	0.2	Fraudulent	0.97	0.97	0.97
			Legitimate	0.97	0.97	0.97
ADABOOST	97.60%	2.32	Fraudulent	0.98	0.97	0.98

Figure 14: Ethereum Fraud Detection Model Results (Taher et al., 2024). PREC = Precision, REC = Recall, ACC = Accuracy F-S = F1 Score.

REFERENCE	ACC
PROPOSED	99%
FARRUGIA ET AL., 2020	96.30%
BHOWMIK ET AL., 2021	98%
Y. ELMOUGY, 2021	80.20%
KABLA ET AL., 2022	98.11%
AMPONSAH ET AL., 2022	97.96%

Figure 15: Comparative analysis between proposed and other researchers in terms of accuracy (Taher et al., 2024).

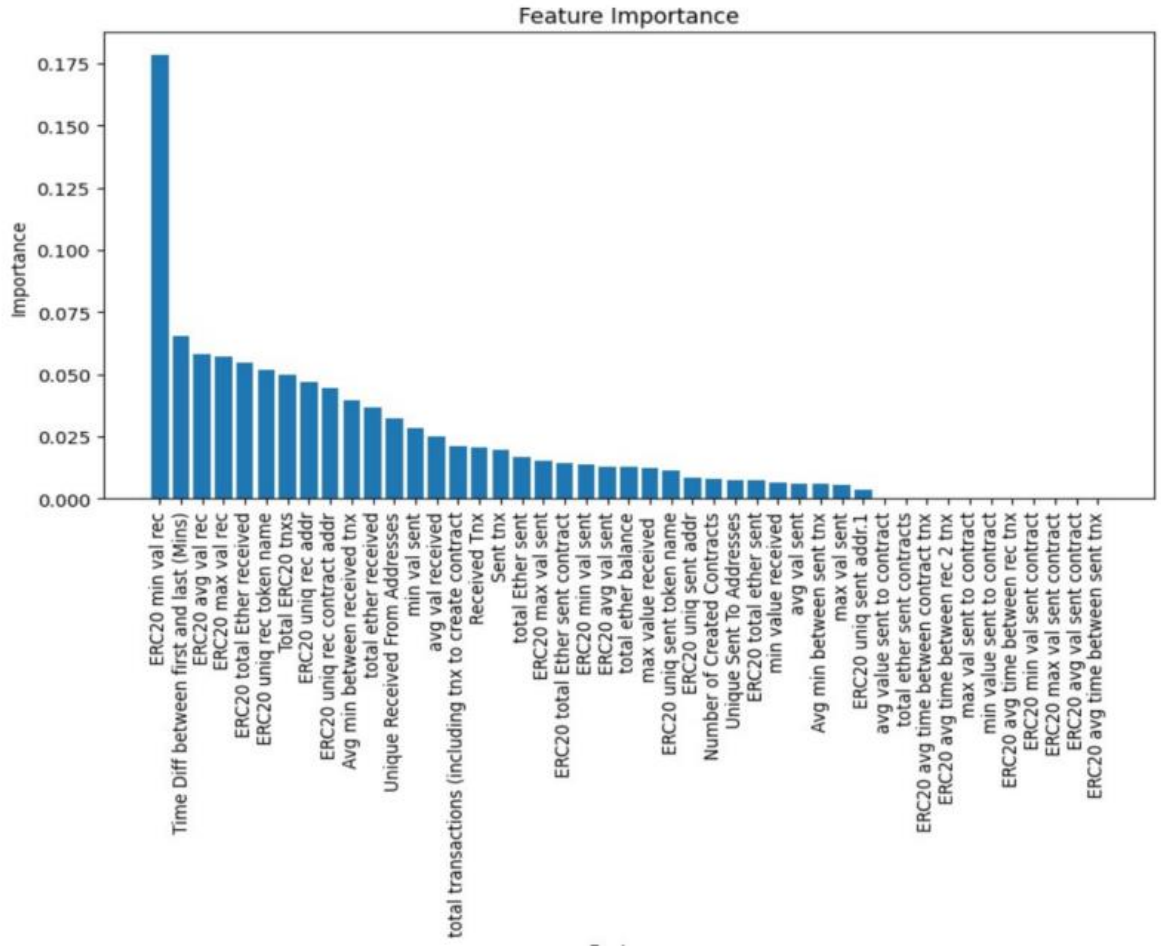


Figure 16: Feature importance rankings utilizing the RF classifier (Taher et al., 2024).

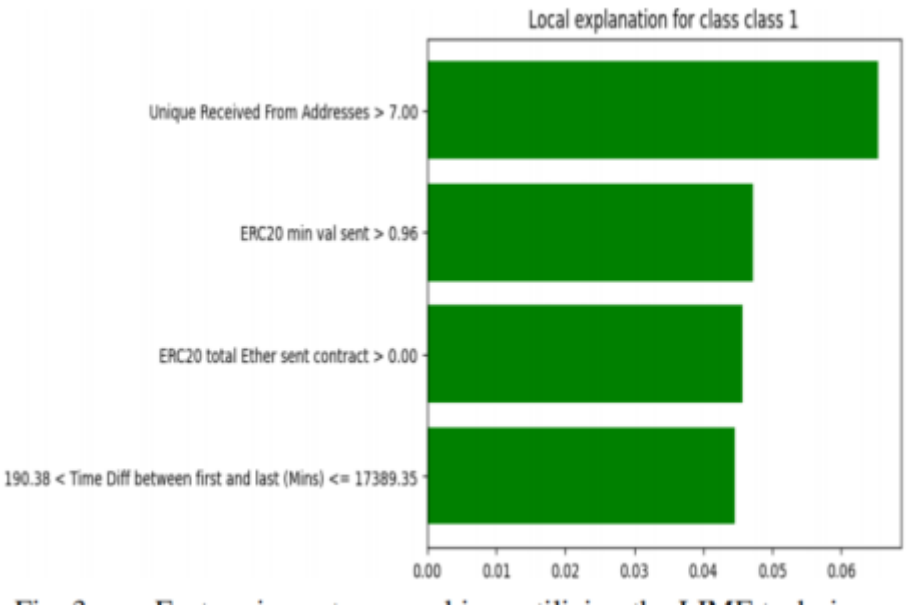


Figure 17: Feature importance ranking utilizing LIME technique (Taher et al., 2024).

In summary, from the results of the models that given by the researchers from the same data original dataset made by Farrugia et al. (2020), the machine learning algorithms that are shown to be effective are *Decision Trees, XGBoost, Random Forest and MLP*. Some detailed steps for the preparation of the dataset are neither mentioned across the researcher, such as data cleaning, balancing and some arguments in detail, nor all of them given links to access their repository to show their work. However, the data scientist can gather some information below, such as:

- **Clustering using t-SNE visualization**: To understand the proposition Ethereum account contained inside the dataset and distinguish the classes that the transactions belonged to.
- **Feature importance ranking**: To get which feature are deemed as significant to the performance of the model building, for dimension reduction purpose.
- **Hyperparameter tunings**: To understand how the researchers has input their modelling.
- **Cross Validations**: To ensure that the performance metrics are as close as their results as possible when the data scientist further research works on the same path. The identified folds of cross validation are 3, 4, 5 and 10.
- **Performance Metrics**: To measure the performance of the dataset with the previous researchers later in the section.

- **Visualizations**: To understand how the researchers conduct their research with other illustrations, such as step diagrams, result comparisons, formulas etc.

The data scientist will use the above performance metrics along with mentioned hyperparameters to build the models, ensuring that the model comparison is made with under the same conditions.

2.3.2 SIMILAR RESEARCH

As the main research papers have shown effectiveness of different machine learning algorithms, other researchers also pursuing the fraud detection techniques using different data collections, transformations, and algorithms as needed. In general, Ethereum transaction blocks may be gathered from the same public Ethereum blockchain using different tools, but the directions may go wary depend on the features that the researchers found and choose to suit the machine learning algorithms that they are using. Hence, the data scientist has done more comparative research to validate the claims of the core research papers and discover the other machine learning algorithms that the researchers has done on this domain. The data scientist would like to perform new proposed model on the dataset that has made to continue the work of Farrugia et al. (2020) on the data source.

First, some research comparative study across multiple algorithms on the dataset from UCSD-FICO competition, which contained 100,000 instances of credit card transactions with 20 attributes from an e-commerce website. The results are calculated using Area Under the ROC Curve (AUC) and Matthews Correlation Coefficient (MCC). But in overall, all machine and deep learning techniques have performed well in detecting fraudulent transactions, with accuracy ranging from 95% to 99% (Raghavan & Gayar, 2019).

Table 6: MCC and AUC of European Dataset (Raghavan & Gayar, 2019)

TABLE I. EUROPEAN DATASET RESULTS

METHOD	MCC	AUC	Cost of Failure
RBM	0.176	0.9109	227360
AUTOENCODERS	0.2315	0.8943	127220
RANDOM FOREST	0.7947	0.8507	30340
CNN	0.8096	0.8764	25700
SVM	0.8145	0.9004	21220
KNN	0.8354	0.8887	22660
ENSEMBLE (KNN, SVM AND CNN)	0.8226	0.8964	21740

Table 7: MCC and AUC of Australian Dataset (Raghavan & Gayar, 2019)

TABLE II. AUSTRALIAN DATASET RESULTS

METHOD	MCC	AUC	Cost of Failure
RBM	0.15	0.5546	24600
AUTOENCODERS	0.2318	0.6174	12220
CNN	0.6408	0.8227	6430
RANDOM FOREST	0.684	0.8416	4700
KNN	0.6905	0.8425	6460
DBN	0.6999	8441	6790
SVM	0.7085	0.8551	3380
ENSEMBLE 1 (KNN, SVM, DBN)	0.7144	0.8573	5290
ENSEMBLE2 (KNN, SVM, RANDOM FOREST)	0.7281	0.8655	3470

Table 8: MCC and AUC of German Dataset (Raghavan & Gayar, 2019)

TABLE III. GERMAN DATASET RESULTS

METHOD	MCC	AUC	Cost of Failure
RBM	0.0984	0.5524	14160
AUTOENCODERS	0.139	0.5614	22640
KNN	0.2487	0.6047	21100
DBN	0.2725	0,5873	23640
RANDOM FOREST	0.2912	0.6437	16970
SVM	0.4038	0.6857	16400
CNN	0.4291	0.7056	14220
ENSEMBLE (SVM, CNN, RANDOM FOREST)	0.4439	0.7011	15620

Table 9: Top Performing Models (Raghavan & Gayar, 2019)

TABLE IV. TOP PERFORMING MODELS

METHOD	Number of times in Top 3
SUPPORT VECTOR MACHINES (SVM)	3 Times
K-NEAREST NEIGHBOURS (KNN)	2 Times
CONVOLUTIONAL NEURAL NETWORKS (CNN)	2 Times
RANDOM FOREST (RF)	2 Times
DEEP BELIEF NETWORK	1 Time

SVM is a machine learning algorithm used for classification and regression analysis. In short, the algorithm separates all the points of data of the origin by maximizing the distance of this hyperplane at the origin. There might be different variants of SVM, such as One-class SVMs, C-Support SVMs, Nu-Support SVMs, epsilon-Support SVMs etc. Some researchers introduced multi stages to enhance the fraud detection performance, such as proposed both One Class Support Vector Machine (OCSVM) combine with K-Means algorithm used together to resolve the vulnerabilities associated with the blockchain technology, as they claimed that the lack of accuracy

and precision of attack type in existing research works on anomaly detection over electronic transactions. One-class SVMs is defined as a binary function that captures the regions of the input space and the probability density of the data in these regions (Sayadi et al., 2019).

From the above results, for individual models, SVM, CNN and KNN come into Top 3 of the most performing model, which is an indicator that SVM has the top spot, but the neural networks are standing in Top 3 for fraud detection, either using them to classifying classifier or to be used for supervised training or using both for better reliability (Raghavan & Gayar, 2019).

In Bitcoin theft detection based on supervised machine learning algorithms, the researchers claimed that SVM is effective for detecting anomalous Bitcoin transactions, as they were able to detect most anomalous transactions while avoiding false positives (B. Chen et al., 2021).

Not only that, the combination of RIPPER and Gradient Boosting (Ensemble Learning model) provides a very reliable result. RIPPER is a data mining technique that is commonly used in fraud detection systems due to its high reliability and coverage. A variant of sequential covering algorithm that is based on incremental reduced error pruning (IREP). In simple terms, the algorithm works by analysing patterns in the data and identifying any anomalies or outliers that may indicate fraudulent activity (Molnar, 2022).

Table 10: RIPPER with Gradient Boosting, Bagging and Stacking (Hoang Khang et al., 2023).

TABLE XI. COMPARE CRITERIA BETWEEN METHODS

CRITERIA	RIPPER (1)	Ripper combines Boosting (2)	Ripper combines Bagging (3)	Ripper combines Stacking (4)
EXECUTION TIME	6m 58s	10m 57.3s	14m 41.7s	8m 46.8s
THE SET OF RULES	32	21	44	45
PRECISION	0.97	0.98	0.95	0.95
RECALL	0.77	0.92	0.95	0.9
F1-SCORE	0.86	0.95	0.95	0.93
ROC	0.883	0.961	0.976	0.952

Not only that, but there are also more studies of RIPPER compared with Bayes Network and Random Forest, which achieved the following results for detecting Bitcoin Ponzi schemes (table of Random Forest section). The result indicated that RIPPER algorithm achieved true positive rate of 80% and false positive rate of 5.6% when ratio of Ponzi scheme is 20:80 (fraud: non-fraud). It is a good classifier to some extent, when dealing with imbalanced classification tasks. However, they have concluded that Random Forest is slightly more effective than RIPPER in their studies (Bartoletti et al., 2018).

In the unlabelled dataset such as Bitcoin network, it was used on comparative studies against K-means and kd-tree algorithms in separating global and local outliers and scored the top as it exhibited near-perfect results on both outlier categories (P. M. Monamo et al., 2017). Random Forest is also used on detecting Ponzi schemes on Bitcoin with other algorithms such as RIPPER and Bayes Network and found out Random Forest may be the most effective algorithms based on multiple settings made between experiments, as although the result performance is heavily dependent on the sampling techniques and the algorithms used.

Table 11: Model Performance between RIPPER, Bayes Network and Random Forest with Cost Matrix of 20 (Bartoletti et al., 2018).

	ACCURACY	**RECALL**	**PRECISION**	**F1 SCORE**
RIPPER	0.965	0.625	0.833	0.714
BAYES NETWORK	0.947	0.719	0.719	0.719
RANDOM FOREST (WITH COST MATRIX)	0.959	0.594	0.846	0.698

Although the Random Forest has recall of 0.594 (59.4%), the researchers suggested that it has the best performance because they have applied the sample-based and the

cost-effective approaches to classify Ponzi schemes correctly. They have built the cost matrix to assign different cost for different types of misclassifications, the correct classification of the Ponzi schemes is the most paramount importance, which Random Forest scored highest by correctly identified 31 out of 32 of cases (Bartoletti et al., 2018).

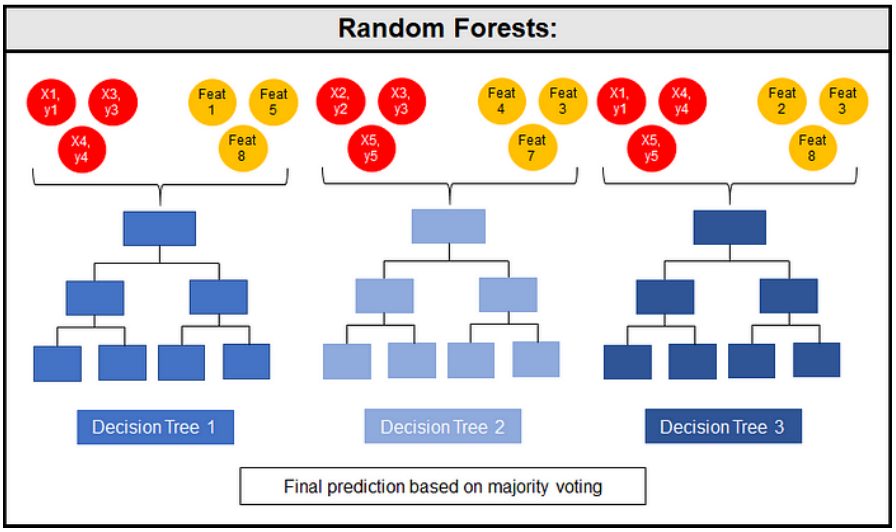


Figure 18: Random Forest in visualization. Using multiple subsamples of the sample to grow multiple weak learners, therefore building an accurate, robust classifier (Nikulski, 2020).

Not only that, but some researchers have also done fraud detection on the Ethereum datasets, with the similar machine learning techniques as well. Random Forest classifier is suitable to find out fraudulent transactions in the dataset, as it can reveal the common characteristics of the Ponzi schemes in the network (W. Chen et al., 2018).

In some highly imbalanced datasets, such as Ethereum blockchain from Etherscan.io website with over *2200 fraud and 349,999 non-fraud transactions*, where the researchers used Random Forests with *SVM* and *XGBoost* with *Grid Search* to get the best configurations with sensitivity analysis. They have concluded that *Random Forest* can be perceived as good candidate for automated anti-fraud system (Ostapowicz & Żbikowski, 2019).

Furthermore, there are researchers who gave accuracy of 98.5%, precision ranging from 75% to 100%, segregate 28 different categories of illicit users with high recall, ranging from 0.50 to 1.00 and F1 score, ranging from 0.60 to 1.00 (Nerurkar et al., 2021).

The research conducted by Bhowmik et al. (2021) embarks on an empirical evaluation of various supervised machine learning techniques to detect fraudulent transactions

within blockchain networks—a pertinent exploration given the technological landscape.

Table 12: Accuracy of Logistic Regression, Multi-Layer Perceptron (MLP), Naïve Bayes, AdaBoost, Decision Tree, SVM, Random Forest, Deep Neural Network (DNN) (Bhowmik et al., 2021).

SL. NO.	ALGORITHM	ACCURACY
1	Logistic regression	0.96
2	Multi-Layer Perceptron (MLP)	0.91
3	Naive Bayes	0.89
4	Ada Boost	0.97
5	Decision Tree	0.96
6	Support Vector Machine (SVM)	0.97
7	Random Forest Classifier	0.97
8	Deep Neural Network	0.94

The study's primary goal lies in the comparative analysis of machine learning algorithms such as Support Vector Machines, Decision Trees, Naive Bayes, Logistic Regression, and others to identify a superior method based on a trade-off between accuracy and computational efficiency. The researchers employ a range of supervised machine learning techniques, asserting the inherent vulnerability of blockchain networks to fraudulent activities regardless of the robustness of consensus algorithms like proof of work or proof of stake. The research extends our understanding of how machine learning can augment traditional detection protocols, noting its potential in addressing blockchain's limitations in authenticating the nature of users and transactions. From the result figure above, Random Forest continues to stand on top with SVM and AdaBoost algorithms (Bhowmik et al., 2021).

Artificial Neural Network (ANN) is a subset machine learning models that draw inspiration from the structure and functionality of the human brain. They comprise interconnected artificial neurons organized in layers. Each neuron receives inputs, processes them, and generates an output. ANNs are versatile and can be

employed for tasks such as image recognition, natural language processing, and prediction. These networks excel at capturing intricate relationships between inputs and outputs, enabling them to address complex real-world problems effectively.

In supply chain studies, according to the results table (see XGBoost table), ANN gives the highest accuracy level of 97%, which indicate that the algorithm does well compare with XGBoost (Gu et al., 2023).

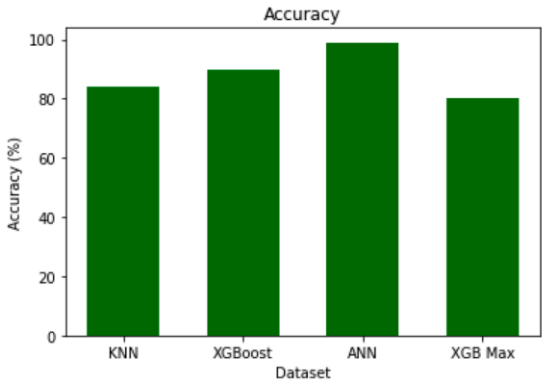


Figure 19: Accuracy Plot (Gu et al., 2023).

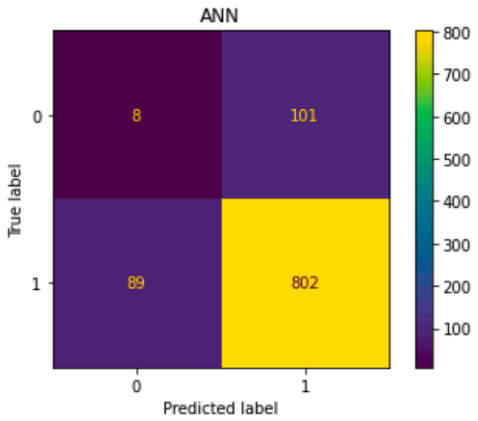


Figure 20: ANN Confusion Matrix (Gu et al., 2023).

Deep Neural Network (DNN) is a type of ANN that has *multiple hidden layers* between the input and output layers. DNN is used for a variety of tasks, including image and speech recognition, natural language processing, and fraud detection. Some comparative studies have been made with the given table (Figure 42). In summary, they claimed that deep learning models are less accurate than machine learning models in terms of accuracy, as DNN get 94 % in accuracy metric (Bhowmik et al., 2021).

Another type of ANN is Convolutional Neural Network (CNN). CNN is a type of neural network that are widely employed for image and video recognition tasks. CNNs are specifically designed to learn hierarchical spatial features from input data, such as images. They accomplish this by utilizing convolutional layers that apply filters to the input data and pooling layers that down sample the output. This architecture enables CNNs to effectively capture local patterns and spatial relationships in the input data. As a result, CNNs excel in tasks such as object recognition, image classification, and image segmentation.

Although CNN use case is not intended for fraud detection due to majority of the data sources involves transactional data, but

there are some other neural networks that is based on CNN, such as *Graph Convolutional Network (GCN)*, which uses graph as input data instead of 2D image data. Some research that involved acquiring dataset from a Bitcoin transaction graph from company Elliptic with consent, has utilized neural network algorithms proved that it performs better than Logistic Regression and Random Forest.

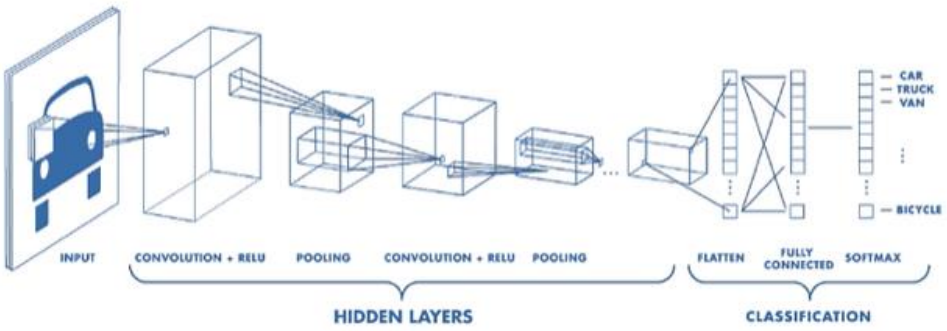


Figure 21: Simple illustration of how CNN works (The MathWorks, 2017).

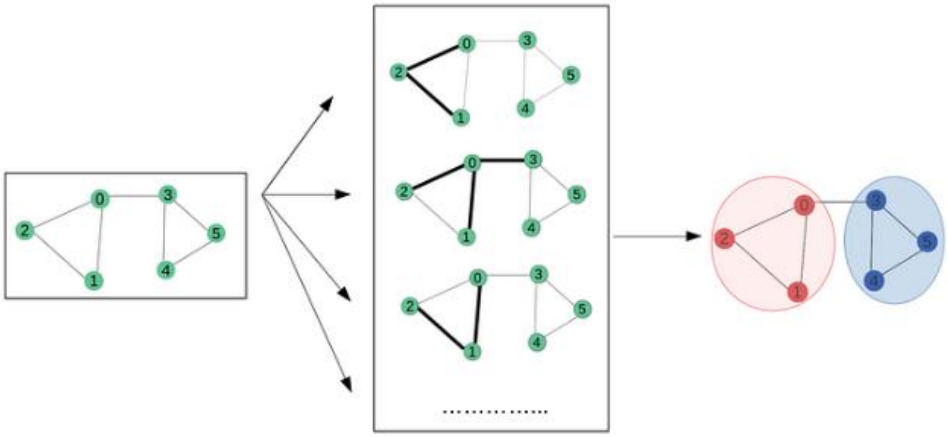


Figure 22: Simple illustration of for GCN (Mayachita, 2020).

Table 3. Overall Performance Comparison

MODEL	Graph size=30,000				Graph size=40,000				Graph size=50,000			
	AUC	Recall	Precision	F1	AUC	Recall	Precision	F1	AUC	Recall	Precision	F1
Features	0.5525	0.1053	0.6143	0.1729	0.5524	0.1053	0.4509	0.1673	0.5776	0.1557	0.5062	0.2327
Deep Walk	0.5624	0.1251	0.7108	0.2049	**0.5725**	**0.1453**	0.5754	0.2227	0.5786	0.1575	0.5945	0.2426
Node2Vec	0.5546	0.1094	0.6956	0.1832	0.5711	0.1424	**0.6689**	0.2267	0.5775	0.1554	**0.6475**	0.2426
LINE	0.5702	0.1409	0.5352	0.2163	0.5664	0.1332	0.5597	0.2087	0.5861	0.1726	0.5222	0.2538
GCN	**0.5725**	**0.1453**	**0.7294**	**0.2357**	**0.5725**	**0.1453**	0.6648	**0.2289**	**0.5866**	**0.1735**	0.6278	**0.2636**

Table 23: Model Performance Comparison between GCN, DeepWalk, Node2Vec and LINE for detecting phishing accounts in Ethereum Transaction Network (L. Chen et al., 2021).

The result above shows that GCN could be potentially useful for detecting phishing scams in Ethereum transaction network, while comparing with random walk-based algorithms such as DeepWalk, Node2Vec and LINE. Based on the table above, the AUC, Recall, Precision and F1 score of GCN does stands up the top when compared with other algorithms (L. Chen et al., 2021).

Neural networks are also versatile. Some research is done for Ethereum that uses *Long Short-Term Memory (LTSM)*, a type of Recurrent Neural Network (RNN) that can be used on sequential datasets, such as financial or time-series transactions (IBM, 2023). The result of the model used gave the following result:

Table 13: Result of LGBM, Ensemble Model, GBM, XGBoost and RNN with LTSM (Proposed) (Sankar Roy et al., 2022).

TABLE IV: COMPARATIVE ANALYSIS BETWEEN DIFFERENT STATE OF THE ART APPROACHES

METHOD	Accuracy	Precision	Recall	F1 score
LGBM	98.6	97.18	92.54	94.86
ENSEMBLE MODEL [13]	89.97	97.44	81.48	88.74
GBM [13]	86.12	95.38	75.92	84.55
XGBOOST [14]	96.34	-	-	99
PROPOSED	99.59	98.59	98.13	98.35

With the results above, the researcher claimed that proposed model is very performant in deep learning algorithms for classifying fraudulent schemes (Sankar Roy et al., 2022).

In summary, the data scientist has done a detailed literature review on the Ethereum network from the context with Ethereum blockchain implementation, to the ways of building the efficient models with the datasets that are compiled in different forms, like transaction graphs that may base on transactions of involved different addresses. Then, the dataset is used to build the proposed machine learning algorithms, with the table and figure that show results the algorithms used with different metrics. The data scientist understood that gathering the transactions of Ethereum network, which type of transaction token or account makes difference in how model performs in real life scenario. From all research papers that have been reviewed by the data scientist, a few observations have been found, which are:

- **XGBoost**, **Random Forest** and **SVM** are frequently used in anomaly and fraud detection research studies, regardless of its variant, part of the procedure or comparison studies, it is referred by the researchers with their proposed models as a benchmark.
- Ethereum and Bitcoin are the frequently used datasets for conducting algorithms in blockchain network.
- Supervised learning methods are generally preferred over unsupervised learning methods due to their superior performance in various performance metrics.
- Many studies have presented inconsistent evaluation metrics, making it challenging to compare and assess the performance of newer deep learning algorithms. An indirect model performance comparison can be made between the existing models collected in this review and any proposed machine learning algorithm if sufficient details are available. It's also better to contribute to the research of previous researchers, such as Farrugia et al. (2020) contributed dataset.
- **XGBoost algorithm is selected** as the machine learning algorithm, as many research study found that it is efficient and proven to be performant in the fraud detection in both Bitcoin and Ethereum transaction across timelines, comparative studies, and experiments.
- **Decision Trees** have been utilized by XGBoost and Random Forest, where they are the improvement of DT with ensemble learning methods

that resolve overfitting, nonlinear relationship handling, and feature importance. Hence, DT is not utilized in the model building in later section.

- **Random Forest and SVM** are also good candidates for fraud detection as they are great performers in the studies and sometimes overlapping each other. Hence, they are used for model comparison during later stages.
- **Artificial Neural Network (ANN)** models are yet to be discovered by the previous researchers on the transformed dataset of Farrugia et al. (2020), and there are multiple variants of neural networks such as RNN, CNN, DNN are tested by other researchers with unspecific hyperparameters.

It is not conclusive that ANN is not suitable for fraud detection of blockchain transactions, as some researchers did claim better results when reviewing on research about detecting phishing accounts on Ethereum network. The data scientist thinks that although machine learning algorithms are proven to give good performance metrics based on the results, but the deep learning neural networks have an edge when the given data input is larger and more complex, which provides more data points to the deep learning model to improve the accuracy of the model.

PROBLEM STATEMENT

Blockchain technology has the potential to revolutionize the way transactions are conducted, making them more secure and transparent. However, it is not immune to fraudulent activities such as money laundering, illicit trade, and other financial crimes. The problem is that traditional methods of detecting fraudulent activities in financial transactions are not well suited for the decentralized and distributed nature of blockchain data. Although some models have been proposed by the researchers with the same dataset or using other methods that produces the dataset that they feel suited for their model building, the model performance is not consistent when applying them in different situations. Dataset complexity also cause the difficulty of analysing the data used for the model which impact the performance as well. Hence, there are still other areas they can be discovered such as neural networks that can effectively detect fraudulent activity in blockchain transactions. In the project lifecycle, data pre-processing and engineering will be implemented and ensure most models are on the same baseline to prevent bias on the results. Not only that, Farrugia et al. (2020) compiled dataset is contributed to continue act as a good benchmark for other model algorithms.

PROJECT QUESTIONS

- What patterns in blockchain transactions that can indicate fraudulent activity?
- How machine learning models detect and flag suspicious transactions in blockchain data accurately?
- Which machine learning algorithm are the most effective for identifying fraudulent activity in blockchain transactions?

AIM

To propose effective method to detect illicit blockchain accounts through transaction history data and develop models that can detect and flag suspicious transactions using machine learning algorithms.

OBJECTIVES

- To identify features of blockchain transaction of the accounts that may indicate fraudulent activity from the blockchain transaction data.

- To develop models that can automatically detect and flag suspicious transactions using machine learning techniques.
- To ensure the model can detect fraudulent activity effectively and efficiently by evaluating the performance of the implemented models.

PROJECT SCOPE

The project involves collecting and pre-processing data on blockchain transactions, exploring and analysing the data to identify patterns and anomalies, and selecting and implementing appropriate machine learning techniques to build a fraud detection model. The data scientist may finalize on one to two ML techniques as the model for fraud detection.

The data scientist used the dataset produced by Farrugia et al. (2020) for fraud detection. It's a dataset that is considered supervised, as there is a target feature to indicate whether the transaction is flagged as fraud. The data scientist will use machine learning algorithm to build the model from the dataset using XGBoost along with other proposed techniques such as DNN, with the given analysis based on the research papers found across timeline. Data preparations, hyperparameter tuning, and visualizations are used to describe the modelling process using various libraries and IDEs available.

The project also includes evaluating the performance of the model and comparing it to some existing fraud detection machine learning algorithms. The evaluation should involve measuring the accuracy, precision, recall, and F1 score of the model and comparing it to baseline methods. These aspects could be useful for other researchers based on the result of this project.

SIGNIFICANCE OF THE STUDY

To further contribute to the previous research of the main research papers, the data scientist has found the data source of the research papers and found the code repository that builds the foundation of the dataset. Since that the Ethereum transactions are valid published on blockchain enquiry website such as etherscan.io, the data scientist will use the dataset given to build new models that are yet to be implemented on the dataset given. The current context of the data source is transactions of the accounts in Ethereum network, after it has been transformed from network graph, which is built from the Ethereum transaction blockchain itself. In the next section, the data scientist will explain on how the dataset is gathered and transformed in detail.

Traditionally, manual fraud detection methods may not be effective in the context of blockchain transactions, due to the unique characteristics of the technology. Machine learning algorithms offer a promising approach for detecting fraud in blockchain transactions, as they can identify complex patterns and anomalies that may be missed by rule-based systems. The study could help to improve the accuracy and efficiency of fraud detection in blockchain transactions, potentially reducing the incidence of financial crimes in the blockchain ecosystem. This could improve the trust and confidence of users in blockchain technology and promote its adoption in various industries. The study could also contribute to broader research on the application of machine learning techniques to blockchain technology, such as deep learning machine algorithm, which is an emerging and rapidly evolving field. The results and insights from this study could inform future research and development in this area in Ethereum transaction network.

CHAPTER 3: PROJECT METHODOLOGY

The chosen methodology for this project is the Cross-Industry Standard Process for Data Mining (CRISP-DM). CRISP-DM is a widely recognized methodology in the industry for conducting standard data mining processes. However, it can also be applied to similar data science projects beyond data mining (IBM, 2021).

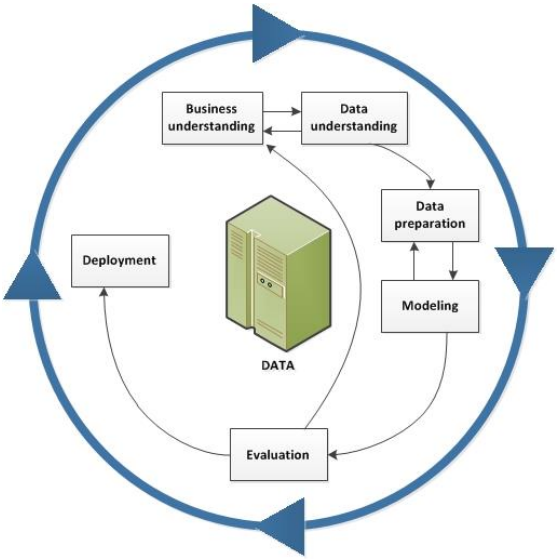


Figure 24: CRISP-DM Data Mining Lifecycle (IBM, CRISP-DM Help Overview, 2021)

The use of the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology in fraud detection of blockchain transactions with machine learning algorithms offers several advantages in this context.

- Structured and systematic approach: CRISP-DM provides a well-defined and structured framework for approaching data mining projects. It consists of six phases: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment. This systematic approach ensures that each phase is properly executed, leading to a comprehensive and reliable fraud detection solution.
- Flexibility for handling complex projects: Fraud detection in blockchain transactions often involves dealing with large and complex datasets, incorporating various machine learning techniques, and addressing specific challenges related to anomaly detection. CRISP-DM's flexibility allows for the adaptation of the methodology to handle the intricacies of fraud detection in blockchain transactions, ensuring that all relevant aspects are considered and effectively addressed.
- Emphasis on data pre-processing and feature engineering: The Data Preparation phase of CRISP-DM focuses on data pre-processing and feature engineering, which are

critical steps in fraud detection. Given the unique characteristics of blockchain transactions, such as their volume, variety, and velocity, effective data pre-processing and feature engineering techniques are necessary to extract meaningful insights and patterns from the data. CRISP-DM's attention to these steps ensures that the data is properly prepared for modelling and analysis.

- Iterative model development and evaluation: CRISP-DM promotes an iterative approach to model development and evaluation. This is particularly valuable in fraud detection, as the models need to be refined and optimized to effectively identify and classify fraudulent transactions. The iterative nature of CRISP-DM allows for continuous feedback and improvement of the models, enhancing their performance over time.

Overall, the adoption of CRISP-DM in fraud detection of blockchain transactions with machine learning algorithms provides a structured and adaptable methodology that addresses the unique challenges of this domain. It facilitates effective data pre-processing, feature engineering, model development, and evaluation, ultimately leading to more accurate and reliable fraud detection solutions.

3.1 RESOURCES

3.1.1 HARDWARE

To perform data science development on the dataset, the data scientist has used the computer hardware:

- CPU: Intel i7-14700K (20 cores, 28 threads)
- RAM: 128 Gigabytes of DDR4
- GPU: Nvidia RTX4080 Super, 16 GB GDDR6 VRAM

3.1.2 SOFTWARE

- Base environment: Python 3.9.19
- IDE: JetBrains PyCharm 2024.1
- Dependencies:
    - Keras
    - TensorFlow
    - Scikit Learn (sklearn)
    - Imbalanced Learn (imblearn)
    - Jupyter Notebook (Jupyter)
    - CUDA
    - Transformers
    - XGBoost
    - Matplotlib
    - Seaborn
    - Sweetviz
    - Pandas
    - SciPy
    - Pandas

    - Streamlit
- Version Control Software: Git with GitHub

The stated libraries above are used for the implementation. There might be other libraries used for data pre-processing and visualisations.

3.2 PROCEDURES

The methodology would likely involve several stages, which will be displayed as sub section with details about the process in each stage.

3.2.1 BUSINESS UNDERSTANDING

In this stage, the data scientist will try to understand the current context of blockchain, and its relative research done in the industry. The data scientist will conduct literature reviews to understand the context and techniques used by other researchers to perform fraud detection in financial transactions (database-based and blockchain-based). In this stage, the research project proposal is made to present and justify the need of the research project. As the previous section, the data scientist has undertaken the following:

- Understanding of the current context of blockchain technology.
- Scope one of the implementations of blockchain technology, and how it is implemented.
- Literature review of the blockchain solution and data sourcing the contributions made by the researchers over time.
- Discover of machine learning algorithms implemented by other researchers in the same domain, while may vary in implementation, while results which can indicate potential new algorithms that can be contributed to the core research on the contributed dataset, to contribute to the core research.

3.2.2 DATA UNDERSTANDING

In this stage, the data scientist will source the dataset contributed by researchers from the literature review. searching process of the suitable data source with the following:

- Deeper understanding on which dataset to be used, confirm the data source that are used by the researchers.
- Identifying the chosen data and reasons.
- Understand the scope of the dataset.
- Understand the included features of the dataset and reasoning of they are included for.
- Verify data integrity to check data quality.
- Understand the target variable of the dataset on how it was based on.
- Check implementation of the found dataset with the literature reviews.

3.2.3 DATA PREPARATION

In this stage, the author will load dataset, to understand the dataset feature with

distributions, cleaning, feature engineering to make sure dataset is ready for modelling process. Along the way, proper justifications are made to perform data imputations, transformations and balancing to ensure the models are not biased and high data quality input.

3.2.4 MODELLING

This stage involves generate models. There are a lot of existing techniques used by the researchers based on the literature reviews in the previous sections. Some given information is used in the model building implementation to ensure that the models performance integrity. Multiple techniques are used including hyperparameters, grid search with multiple fold of cross validation according to the existing research made to the dataset.

3.2.5 EVALUATION

This stage evaluates the performance of the created model(s) and review the process. The models are made with the given hyperparameters given and model performance metrics are calculated, such as confusion matrix, accuracy, precision, recall, F1 score, AUC-ROC score etc.

3.2.6 DEPLOYMENT VISUALIZATION

Deploy the built models with visualizations of the final model(s) performance with analysis and findings. Compare the model with other algorithms for validation analysis. Not only that, but tables are also created with performance metrics that are essential for benchmarking.

3.2.7 DOCUMENTATION & SUBMISSION

The final documentation of the previous stages is organized and documented in formal documentation with presentations to share the results. The documentation provides a way to allow further contributions to the current research.

3.3 DATASET

3.3.1 DATA COLLECTION

To collect the data source that used by previous researchers, the data scientist has acquired the original data source from the following GitHub repository link based on the published research paper:

https://github.com/sfarrugia15/Ethereum_Fraud_Detection

Based on the original researcher, the dataset is collected from and transformed in the following steps:

- Get **API key** from EtherscamDB and Etherscan websites.
- Establish Connection to Neo4j Database: Connect to a Neo4j graph database, providing the URI and authentication credentials.
- **Retrieve Illicit Account Addresses**: Obtain illicit account addresses from an external source from EtherscamDB API.
- **Identify Illicit Transactions**:
    - Iterate over the list of illicit account addresses.
    - Query the Neo4j database for transactions associated with each illicit address to potentially identify illicit transactions.
- **Retrieve Normal Account Addresses**: Extract normal account addresses from transaction data stored in a CSV file.
- **Check for New Illicit Accounts**: Compare previously saved illicit account addresses with the current list to check for new illicit accounts.
- Interact with the Etherscan API:
    - Use the Etherscan API to **retrieve transaction data** for a list of account addresses.
    - Process both **normal and ERC20 token transfer transactions** for each address.
- **Calculate Account Balances**: Utilize the Etherscan API to determine the balance of each account by querying the API endpoint.
- **Retrieve Transaction Data and Compute Statistics**:
    - Define a query to retrieve transaction data from the Neo4j database.
    - Compute various statistics on the transactions, including total ether received/sent, transaction balance, average received/sent ether, etc.
    - Execute the query and compute statistics using a session object.
- **Additional Utility Functions**:
    - Define helper functions to calculate various statistics (e.g., average time between transactions, minimum/maximum/average transaction values, etc.).

- Handle uniqueness of addresses and find the most frequent element in a list.

With the main steps above, the result dataset is made. There are more steps needed to make the local machine to be an Ethereum node, by using a tool called Geth client and connect it to IPC pipe. Then, the dataset is synchronized to the local directory and saved into the database.

What is the dataset with the target value based on? According to Farrugia et al. (2020), flag target feature is based on the Ethereum accounts that are flagged for illicit activities with the following pattern:

- Imitate other contract addresses.
- Scam lotteries.
- Fake ICOs
- Imitating other users
- Ponzi schemes
- Phishing
- Mirroring websites

Not only that, but the dataset will also contain the following features that the Ethereum transactions that the illicit and normal accounts that involved in have the following:

- Interaction and Transaction Values
- Node Degrees Features
- Transaction Amount Variance
- Smart Contract Features
- Successful Transactions
- Illicit Identification Markers

The features have specific patterns that can be input to the model building.

The original data source was gathered the researcher in CSV file format, and it has 2179 illicit accounts *flagged by the Ethereum community* for their illegal activity with 2502 normal accounts which has total 4681 observations.

Here are the following dimensions of the dataset, as quoted from the dataset description:

No	Field	Description
1	Index	The row index number.
2	Address	The Ethereum account address.
3	FLAG	Indicates whether the transaction is fraudulent or not.
4	Avg min between sent tnx	Average time between sent transactions for the account in minutes.
5	Avg min between received tnx	Average time between received transactions for the account in minutes.
6	Time Diff between first and last (Mins)	Time difference between the first and last transaction in minutes.
7	Sent_tnx	Total number of normal transactions sent.
8	Received_tnx	Total number of normal transactions received.
9	NumberofCreated_Contracts	Total number of contract transactions created.
10	UniqueReceivedFrom_Addresses	Total number of unique addresses from which the account received transactions.
11	UniqueSentTo_Addresses20	Total number of unique addresses to which the account sent transactions.
12	MinValueReceived	Minimum value of Ether received.
13	MaxValueReceived	Maximum value of Ether received.
14	AvgValueReceived	Average value of Ether received.
15	MinValSent	Minimum value of Ether sent.
16	MaxValSent	Maximum value of Ether sent.
17	AvgValSent	Average value of Ether sent.
18	MinValueSentToContract	Minimum value of Ether sent to a contract.
19	MaxValueSentToContract	Maximum value of Ether sent to a contract.
20	AvgValueSentToContract	Average value of Ether sent to contracts.
21	TotalTransactions(IncludingTnxtoCreate_Contract)	Total number of transactions, including transactions to create contracts.
22	TotalEtherSent	Total Ether sent from the account address.
23	TotalEtherReceived	Total Ether received to the account address.
24	TotalEtherSent_Contracts	Total Ether sent to contract addresses.
25	TotalEtherBalance	Total Ether balance after executing transactions.
26	TotalERC20Tnxs	Total number of ERC20 token transfer transactions.
27	ERC20TotalEther_Received	Total Ether received from ERC20 token transactions.
28	ERC20TotalEther_Sent	Total Ether sent in ERC20 token transactions.
29	ERC20TotalEtherSentContract	Total Ether sent to other contracts in ERC20 token transactions.
30	ERC20UniqSent_Addr	Number of unique account addresses to which ERC20 token transactions were sent.
31	ERC20UniqRec_Addr	Number of unique addresses from which ERC20 token transactions were received.
32	ERC20UniqRecContractAddr	Number of unique contract addresses from which ERC20 token transactions were received.
33	ERC20AvgTimeBetweenSent_Tnx	Average time between ERC20 token sent transactions in minutes.
34	ERC20AvgTimeBetweenRec_Tnx	Average time between ERC20 token received transactions in minutes.
35	ERC20AvgTimeBetweenContract_Tnx	Average time between ERC20 token transactions sent to other contracts.
36	ERC20MinVal_Rec	Minimum value of Ether received from ERC20 token transactions for the account.
37	ERC20MaxVal_Rec	Maximum value of Ether received from ERC20 token transactions for the account.
38	ERC20AvgVal_Rec	Average value of Ether received from ERC20 token transactions for the account.
39	ERC20MinVal_Sent	Minimum value of Ether sent in ERC20 token transactions for the account.
40	ERC20MaxVal_Sent	Maximum value of Ether sent in ERC20 token transactions for the account.
41	ERC20AvgVal_Sent	Average value of Ether sent in ERC20 token transactions for the account.
42	ERC20UniqSentTokenName	Number of unique ERC20 tokens transferred.
43	RC20UniqRecTokenName	Number of unique ERC20 tokens received.
44	ERC20MostSentTokenType	Most frequently sent token for the account via ERC20 transactions.
45	ERC20MostRecTokenType	Most frequently received token for the account via ERC20 transactions.

Figure 25: Features of dataset with Description.

There are 43 features in the dataset for each address, excluding Index and FLAG fields.

3.4 PROJECT PLAN

Based on the chosen methodology, the data scientist has proposed a Gantt Chart that describe the project schedule of how the project will be conducted over the span of a year. The chart contains multiple stages, and each stage has multiple subtasks and milestones. There are percentage bars to mark the progress of each sub task and milestone percentages to view the average progress of the stage itself. The chart is flexible to be edited to estimate the progress of the project with some time to spare. The timeline is day basis, but the subtasks are mostly executed by weeks, for example, in Data Understanding stage, Identify, Select and Prepare Data Source, the subtask will be conducted for 14 days, which are 2 weeks in time.

As the Gantt Chart is updated over time and for better management. the data scientist has shared the Gantt chart online (offline copy during submission) for better viewing. Link:

Capstone Project Gantt Chart TP030562.xlsx

STAGE 1: BUSINESS UNDERSTANDING

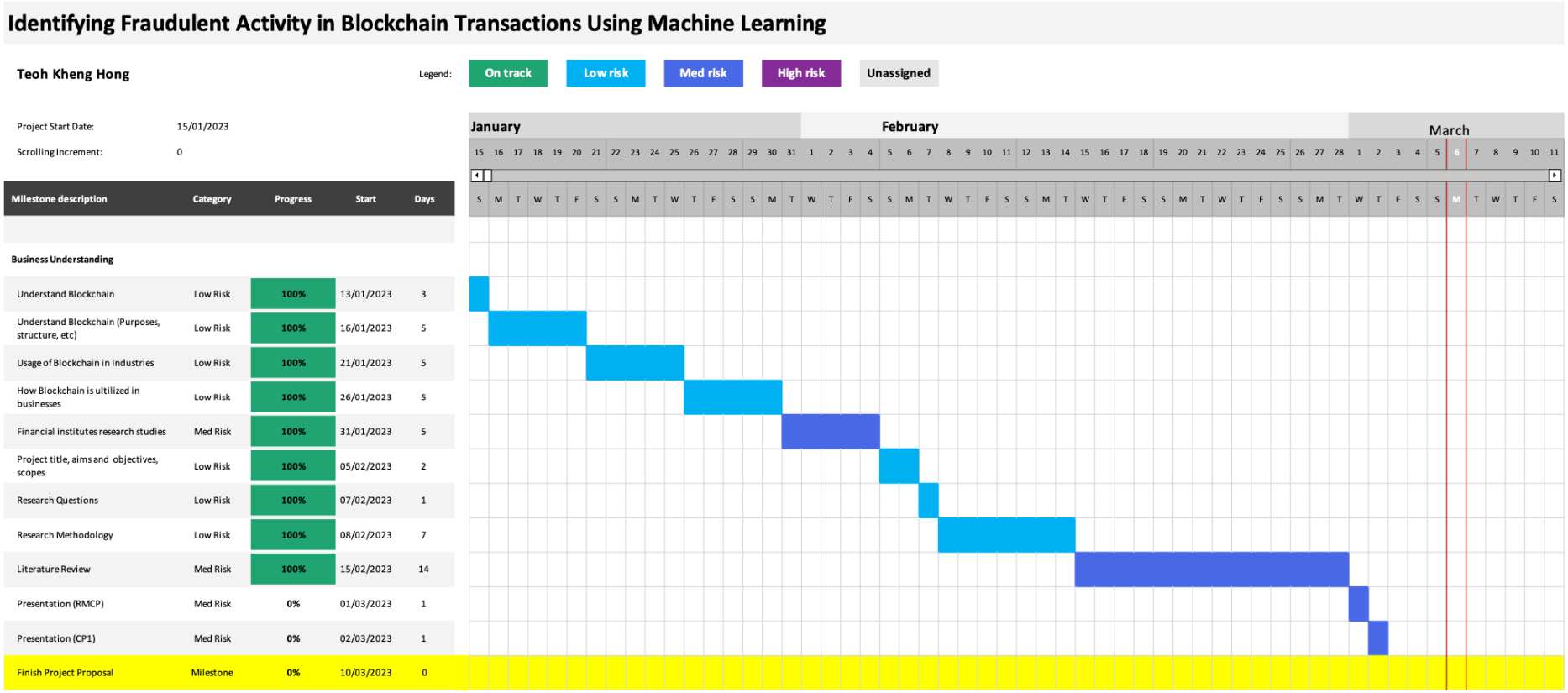


Figure 26: Business Understanding

In this stage, the milestone has been set to create a project proposal to confirm the direction of the research.

STAGE 2: DATA UNDERSTANDING

Identifying Fraudulent Activity in Blockchain Transactions Using Machine Learning

Teoh Kheng Hong

Legend: On track | Low risk | Med risk | High risk | Unassigned

Project Start Date: 15/01/2023

Scrolling Increment: 45

Milestone description	Category	Progress	Start	Days
Data Understanding				
Identify, Select and Prepare Data Source	Low Risk	0%	02/03/2023	14
Describe data source	Low Risk	0%	16/03/2023	9
Verify Data Integrity	Med Risk	0%	25/03/2023	11
Initial Dataset analysis	Med Risk	0%	05/04/2023	7
Dataset Intitial Review	Med Risk	0%	12/04/2023	14
Ready Dataset for Next Process	Milestone	0%	12/04/2023	0

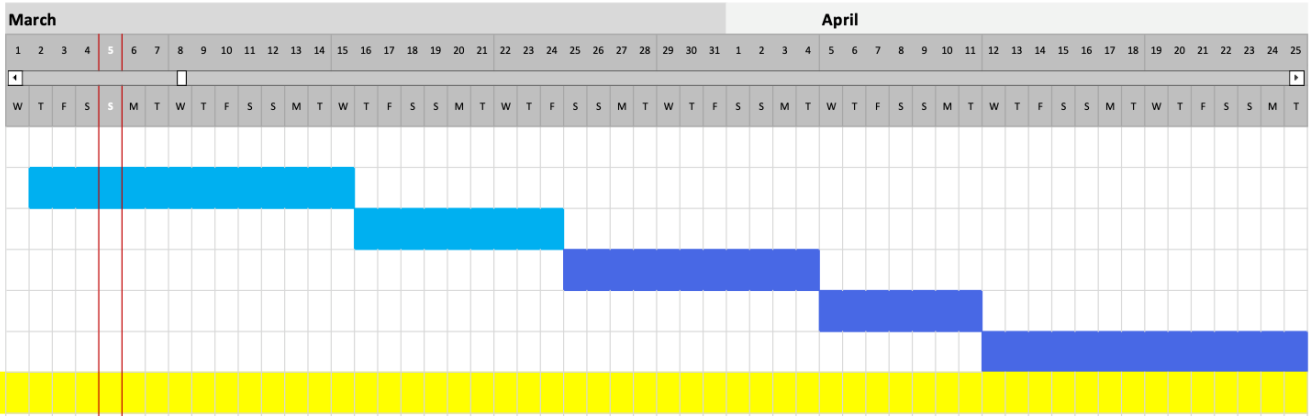


Figure 27: Data Understanding Screenshot

Identifying Fraudulent Activity in Blockchain Transactions Using Machine Learning

Teoh Kheng Hong

Legend: On track | Low risk | Med risk | High risk | Unassigned

Project Start Date: 15/01/2023

Scrolling Increment: 100

Milestone description	Category	Progress	Start	Days
Data Understanding				
Identify, Select and Prepare Data Source	Low Risk	0%	10/03/2023	14
Describe data source	Low Risk	0%	24/03/2023	9
Verify Data Integrity	Med Risk	0%	02/04/2023	11
Initial Dataset analysis	Med Risk	0%	13/04/2023	7
Dataset Intitial Review	Med Risk	0%	20/04/2023	14
Ready Dataset for Next Process	Milestone	0%	20/04/2023	0

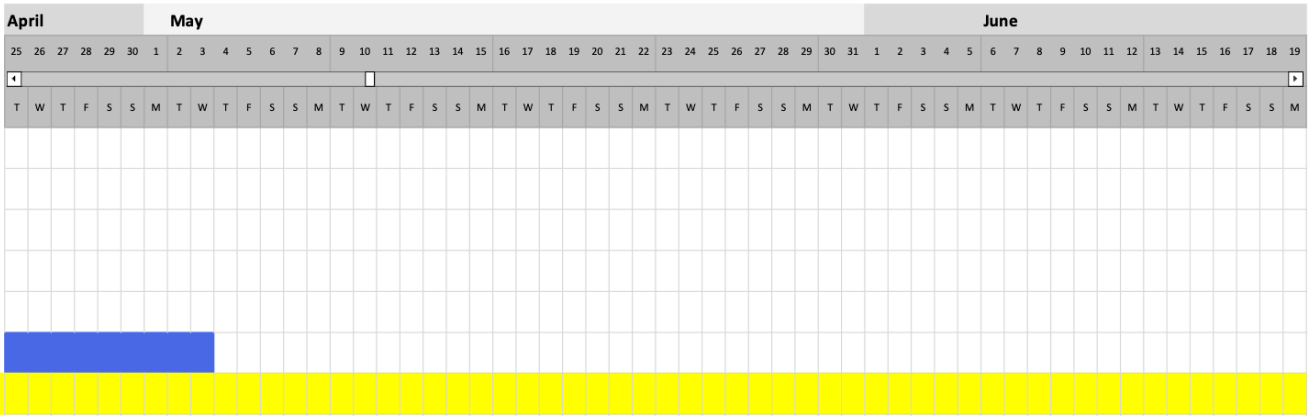


Figure 28: Data Understanding

STAGE 3: DATA PREPARATION

Identifying Fraudulent Activity in Blockchain Transactions Using Machine Learning

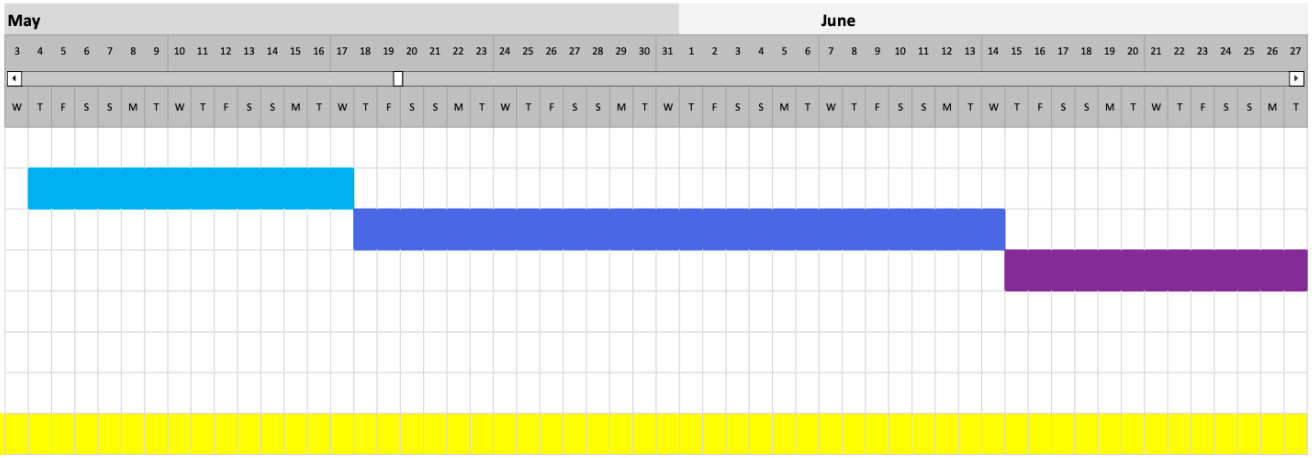
Teoh Kheng Hong

Legend: On track | Low risk | Med risk | High risk | Unassigned

Project Start Date: 15/01/2023

Scrolling Increment: 108

Milestone description	Category	Progress	Start	Days
Data Preparation				
Data Selection	Low Risk	0%	04/05/2023	14
Data Cleaning	Med Risk	0%	18/05/2023	28
Feature Engineering	High Risk	0%	15/06/2023	42
Data Integration	Low Risk	0%	27/07/2023	14
Data Formatting	Med Risk	0%	10/08/2023	42
Exploratory Data Analysis	Med Risk	0%	21/09/2023	14
Ready Dataset for ML	Milestone	0%	05/10/2023	0



Identifying Fraudulent Activity in Blockchain Transactions Using Machine Learning

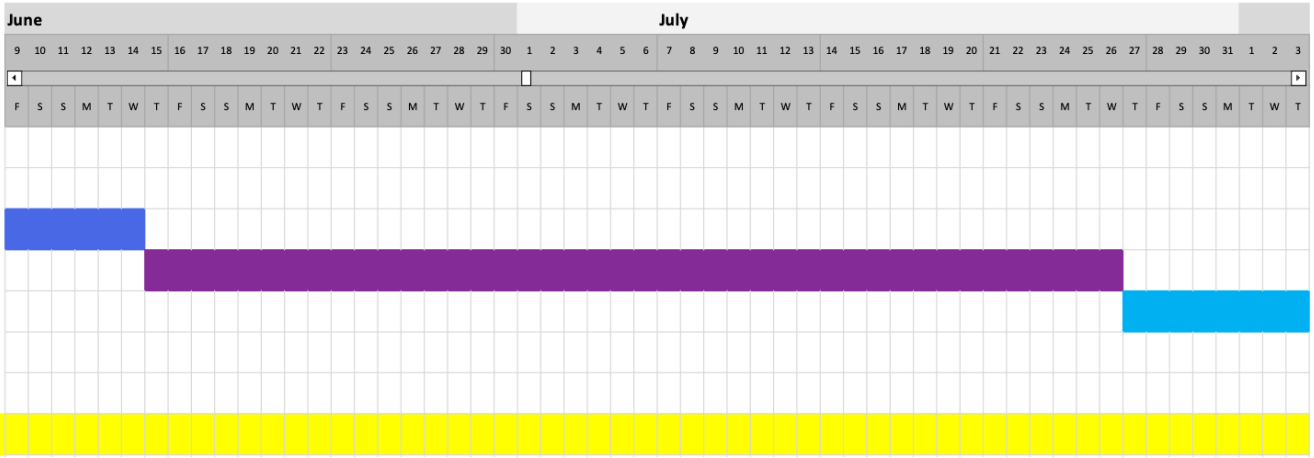
Teoh Kheng Hong

Legend: On track | Low risk | Med risk | High risk | Unassigned

Project Start Date: 15/01/2023

Scrolling Increment: 145

Milestone description	Category	Progress	Start	Days
Data Preparation				
Data Selection	Low Risk	0%	04/05/2023	14
Data Cleaning	Med Risk	0%	18/05/2023	28
Feature Engineering	High Risk	0%	15/06/2023	42
Data Integration	Low Risk	0%	27/07/2023	14
Data Formatting	Med Risk	0%	10/08/2023	42
Exploratory Data Analysis	Med Risk	0%	21/09/2023	14
Ready Dataset for ML	Milestone	0%	05/10/2023	0



Identifying Fraudulent Activity in Blockchain Transactions Using Machine Learning

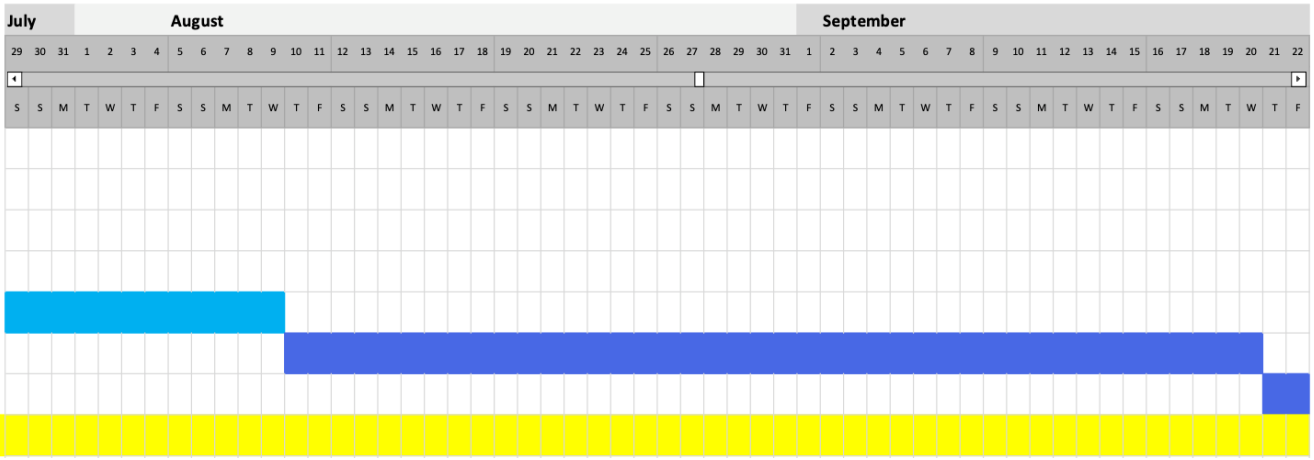
Teoh Kheng Hong

Legend: On track | Low risk | Med risk | High risk | Unassigned

Project Start Date: 15/01/2023

Scrolling Increment: 195

Milestone description	Category	Progress	Start	Days
Data Preparation				
Data Selection	Low Risk	0%	04/05/2023	14
Data Cleaning	Med Risk	0%	18/05/2023	28
Feature Engineering	High Risk	0%	15/06/2023	42
Data Integration	Low Risk	0%	27/07/2023	14
Data Formatting	Med Risk	0%	10/08/2023	42
Exploratory Data Analysis	Med Risk	0%	21/09/2023	14
Ready Dataset for ML	Milestone	0%	05/10/2023	0



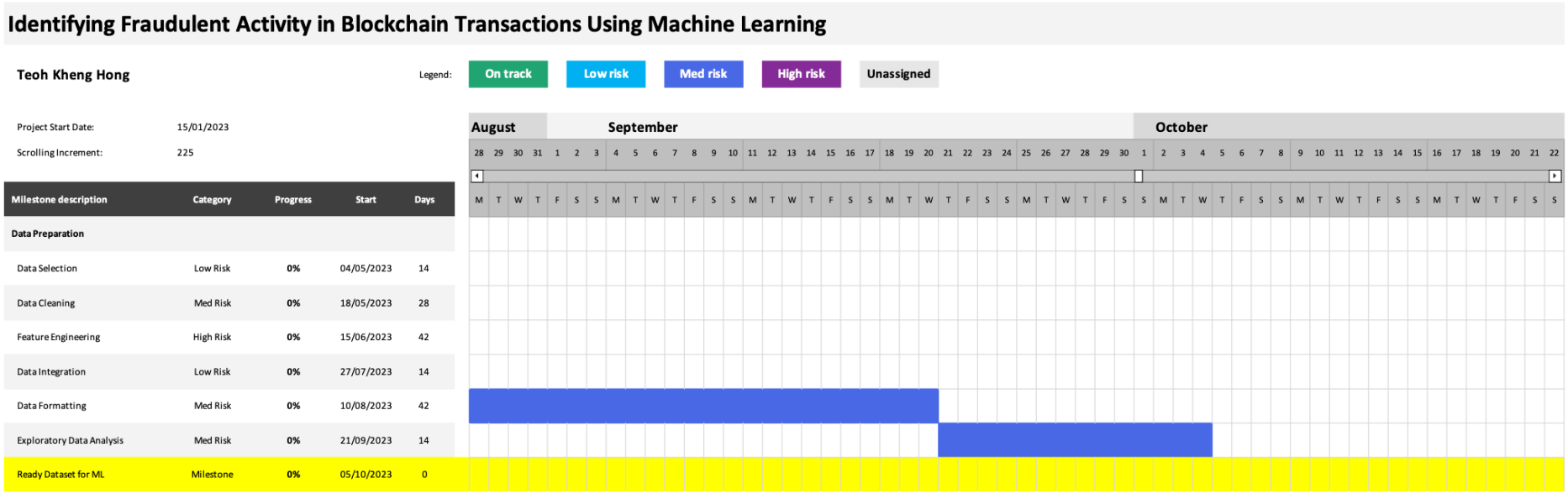


Figure 29: Data Preparation

The data scientist may perform cycling process between Stage 3 and Stage 4, to perform extensive data preparation and modelling on different datasets. The time has been stretched to allow suitable model to be ready for evaluation.

STAGE 4: MODELLING

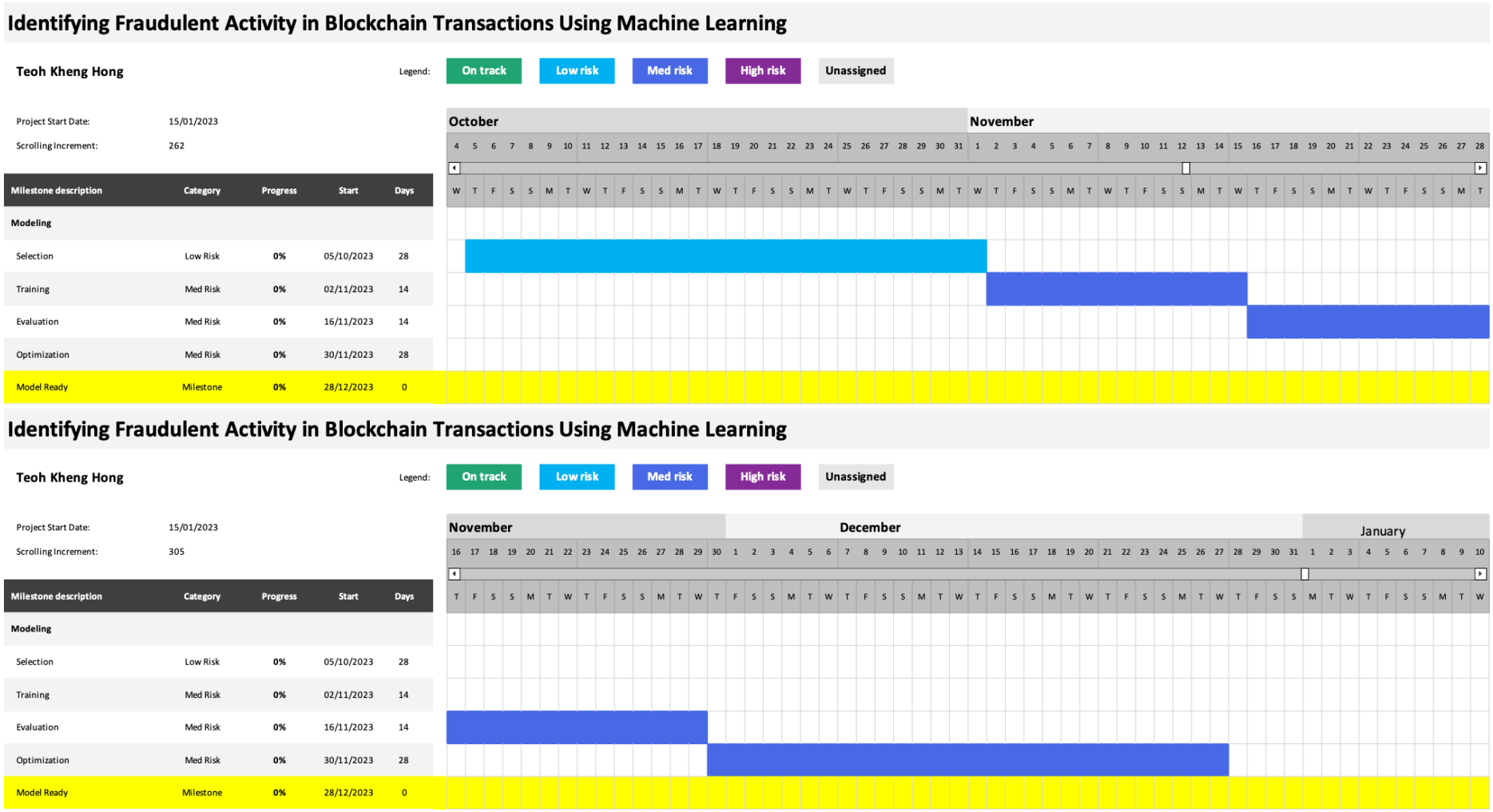


Figure 30: Modelling.

STAGE 5: EVALUATION

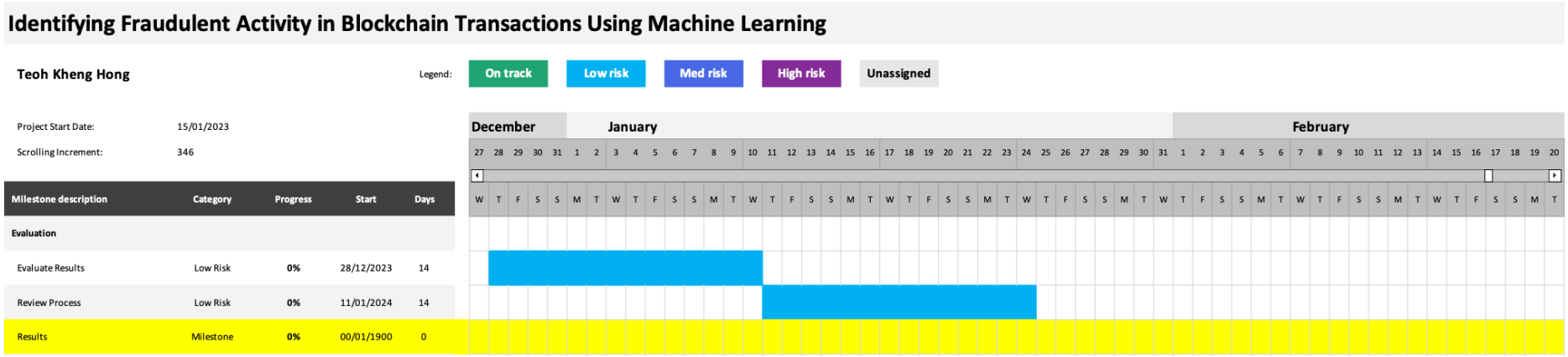


Figure 31: Evaluation.

In this stage, the evaluation process is taken to evaluate and understand the results thoroughly before using it in later stages to ensure accurate results on the model itself.

STAGE 6: DEPLOYMENT & VISUALIZATIONS

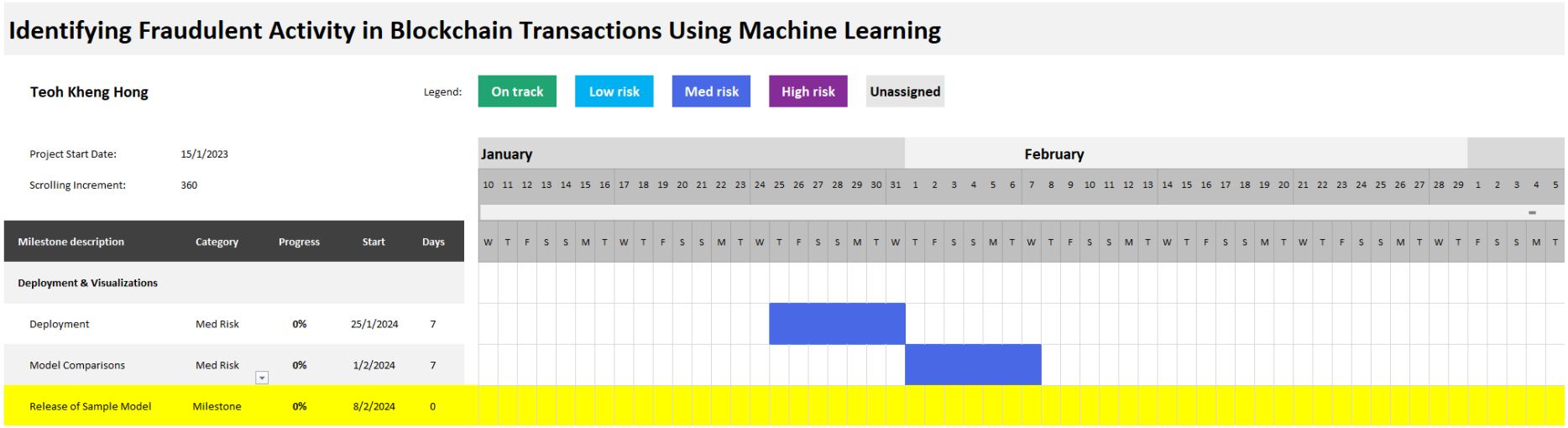


Figure 32: Visualization.

In this stage, the model will be deployed, visualized and comparisons made between models are given for further analysis and justifications.

STAGE 7: DOCUMENTATION & SUBMISSION

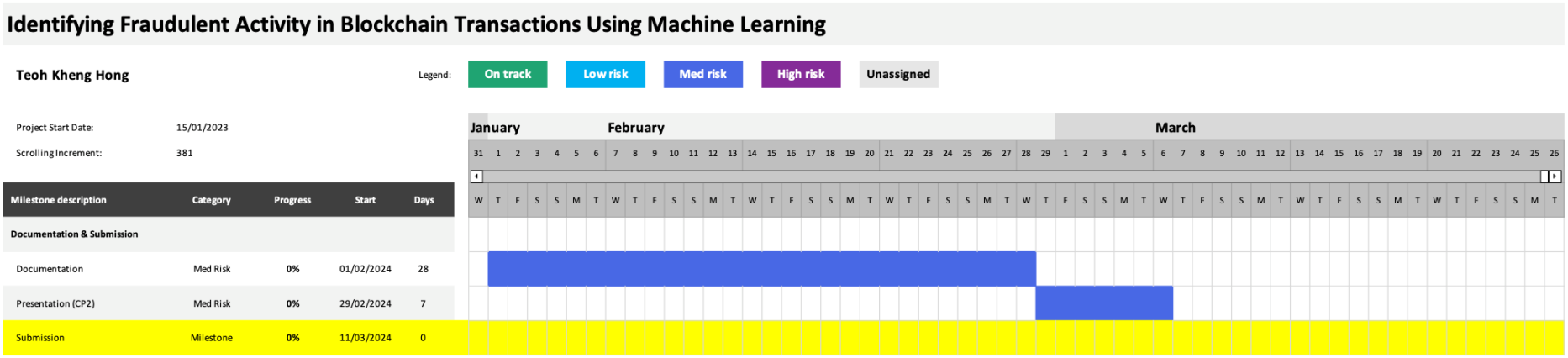


Figure 33: Documentation & Submission

In the last stage, the whole process of the model from previous stages are compiled into the documentation and arranged to illustrate the conducted research project better. Presentation is made to show the summary of the processes and some extended times are included in the timeline for proofreading and error checking.

Note: The assignment has extended by the data scientist with approval of external circumstances (EC) application by the university. However, the original assignment timeline is not changed due to the complexity of the time management if brought the extended timeline into the process. Hence, the visualization is not changed.

CHATPER 4: IMPLEMENTATION

In this stage, the data scientist will perform data selection, cleaning, feature engineering to make sure dataset is ready for modelling process.

A code repository is made by the data scientist has been published, along with other instructions to initialize the environment correctly and reproduce the output. Please visit the following link:

https://github.com/tkh-apu-master/capstone-project

The code has been organized in the way that the user can execute some iterations easily with skipping of certain processing that requires long time to execute (for easier debugging).

4.1 IMPORT GENERAL LIBRARIES

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

The above libraries are executed to provide basic utilities to data preprocessing, imputation, cleaning, and visualization process.

4.2 LOAD DATASET

```
filename = 'transaction_dataset.csv'
# Count dataset number of lines to inform the dataset size first
def count_dataset_rows(file_name):
    fp = open(file_name,'r')
    for line_count, line in enumerate(fp):
        pass
```

```
    return line_count


file_line_count = count_dataset_rows(filename)
print('file_line_count for transaction_dataset.csv: ', file_line_count)
```

Some reading of the dataset is required to find out the number of lines in the file. This is to inform the user that the dataset is big before decided to import the records to the memory.

Output:

```
file_line_count for transaction_dataset.csv:  4681
```

Then, check the records of the dataset to confirm the data is loaded correctly.

```
df = pd.read_csv(filename, index_col=0)
print(df.shape)
df.head()
```

Output:

```
(4681, 49)
```

	Index	Address	FLAG	Avg min between sent tnx	Avg min between received tnx	Time Diff between first and last (Mins)	Sent tnx	Received Tnx	Number of Created Contracts	Unique Received From Addresses
0	1	0x00009277775ac7d0d59eaad8fee3d10ac6c805e8	0	844.26	1093.71	704785.63	721	89	0	40
1	2	0x0002b44ddb1476db43c868bd494422ee4c136fed	0	12709.07	2958.44	1218216.73	94	8	0	5
2	3	0x0002bda54cb772d040f779e88eb453cac0daa244	0	246194.54	2434.02	516729.30	2	10	0	10
3	4	0x00038e6ba2fd5c09aedb96697c8d7b8fa6632e5e	0	10219.60	15785.09	397555.90	25	9	0	7
4	5	0x00062d1dd1afb6fb02540ddad9cdebfe568e0d89	0	36.61	10707.77	382472.42	4598	20	1	7

	...	ERC20 min val sent	ERC20 max val sent	ERC20 avg val sent	ERC20 min val sent contract	ERC20 max val sent contract	ERC20 avg val sent contract	ERC20 uniq sent token name	ERC20 uniq rec token name	ERC20 most sent token type	ERC20_most_rec_token_type
0	...	0.000000	1.683100e+07	271779.920000	0.0	0.0	0.0	39.0	57.0	Cofoundit	Numeraire
1	...	2.260809	2.260809e+00	2.260809	0.0	0.0	0.0	1.0	7.0	Livepeer Token	Livepeer Token
2	...	0.000000	0.000000e+00	0.000000	0.0	0.0	0.0	0.0	8.0	None	XENON
3	...	100.000000	9.029231e+03	3804.076893	0.0	0.0	0.0	1.0	11.0	Raiden	XENON
4	...	0.000000	4.500000e+04	13726.659220	0.0	0.0	0.0	6.0	27.0	StatusNetwork	EOS

Some features are not useful such as Index and Address.

```
# Omit first two columns (Index, Address)
df = df.iloc[:,1:]
```

4.3 DATA PRE-PROCESSING

```
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
Index: 4681 entries, 1 to 2502
Data columns (total 48 columns):
```

```
#   Column                                                  Non-Null Count  Dtype
---  ------                                                 --------------  -----
0   FLAG                                                    4681 non-null   int64
1   Avg_min_between_sent_tnx                                4681 non-null   float64
2   Avg_min_between_received_tnx                            4681 non-null   float64
3   Time_Diff_between_first_and_last_(Mins)                 4681 non-null   float64
4   Sent_tnx                                                4681 non-null   int64
5   Received_Tnx                                            4681 non-null   int64
6   Number_of_Created_Contracts                             4681 non-null   int64
7   Unique_Received_From_Addresses                          4681 non-null   int64
8   Unique_Sent_To_Addresses                                4681 non-null   int64
9   min_value_received                                      4681 non-null   float64
10  max_value_received                                      4681 non-null   float64
11  avg_val_received                                        4681 non-null   float64
12  min_val_sent                                            4681 non-null   float64
13  max_val_sent                                            4681 non-null   float64
14  avg_val_sent                                            4681 non-null   float64
15  min_value_sent_to_contract                              4681 non-null   int64
16  max_val_sent_to_contract                                4681 non-null   int64
17  avg_value_sent_to_contract                              4681 non-null   int64
18  total_transactions_(including_tnx_to_create_contract)   4681 non-null   int64
19  total_Ether_sent                                        4681 non-null   float64
20  total_ether_received                                    4681 non-null   float64
21  total_ether_sent_contracts                              4681 non-null   int64
22  total_ether_balance                                     4681 non-null   float64
23  Total_ERC20_tnxs                                        3852 non-null   float64
24  ERC20_total_Ether_received                              3852 non-null   float64
25  ERC20_total_ether_sent                                  3852 non-null   float64
26  ERC20_total_Ether_sent_contract                         3852 non-null   float64
27  ERC20_uniq_sent_addr                                    3852 non-null   float64
28  ERC20_uniq_rec_addr                                     3852 non-null   float64
29  ERC20_uniq_sent_addr.1                                  3852 non-null   float64
30  ERC20_uniq_rec_contract_addr                            3852 non-null   float64
31  ERC20_avg_time_between_sent_tnx                         3852 non-null   float64
```

```
32  ERC20_avg_time_between_rec_tnx              3852 non-null   float64
33  ERC20_avg_time_between_rec_2_tnx            3852 non-null   float64
34  ERC20_avg_time_between_contract_tnx         3852 non-null   float64
35  ERC20_min_val_rec                           3852 non-null   float64
36  ERC20_max_val_rec                           3852 non-null   float64
37  ERC20_avg_val_rec                           3852 non-null   float64
38  ERC20_min_val_sent                          3852 non-null   float64
39  ERC20_max_val_sent                          3852 non-null   float64
40  ERC20_avg_val_sent                          3852 non-null   float64
41  ERC20_min_val_sent_contract                 3852 non-null   float64
42  ERC20_max_val_sent_contract                 3852 non-null   float64
43  ERC20_avg_val_sent_contract                 3852 non-null   float64
44  ERC20_uniq_sent_token_name                  3852 non-null   float64
45  ERC20_uniq_rec_token_name                   3852 non-null   float64
46  ERC20_most_sent_token_type                  3851 non-null   object
47  ERC20_most_rec_token_type                   3845 non-null   object
dtypes: float64(35), int64(11), object(2)
```

Let’s view the information for the categorical and numerical columns.

```
# Turn object variables into 'category' dtype for more computation efficiency
categories = df.select_dtypes('O').columns.astype('category')
df[categories]
```

Output:

11 rows 4681 rows × 2 columns

Index	ERC20_most_sent_token_type	ERC20_most_rec_token_type
1	NaN	NaN
2		Blockwell say NOTSAFU
3		Blockwell say NOTSAFU
4	OCoin	OCoin
5	NaN	NaN
...	...	...
2498	0	0
2499		VIU
2500	0	0
2501	0	0
2502		Omise GO

```
# Inspect numerical
numericals = df.select_dtypes(include=['float','int']).columns
df[numericals].describe()
```

Output:

8 rows × 46 columns

	FLAG	Avg_min_between_sent_tnx	Avg_min_between_received_tnx	Time_Diff_between_first_and_last_(Mins)	Sent_tnx	Received_Tnx	Number_of_Created_Contracts	Unique_Received_From_Addresses	Uniqu
count	4681.000000	4681.000000	4681.000000	4.681000e+03	4681.000000	4681.000000	4681.000000	4681.000000	
mean	0.465499	3433.476676	5175.318889	1.310965e+05	92.834651	189.607990	4.955992	42.812647	
std	0.498862	16386.028874	18295.726331	2.396813e+05	736.368423	1141.163981	185.494671	355.514463	
min	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	6.965000e+01	0.000000	1.000000	0.000000	1.000000	
50%	0.000000	3.340000	92.650000	1.073972e+04	2.000000	3.000000	0.000000	2.000000	
75%	1.000000	164.890000	2284.990000	1.835153e+05	4.000000	18.000000	0.000000	6.000000	
max	1.000000	377913.420000	348288.300000	1.873350e+06	10000.000000	10000.000000	9995.000000	9999.000000	

8 rows × 46 columns

Unique_Sent_To_Addresses	min_value_received	...	ERC20_max_val_rec	ERC20_avg_val_rec	ERC20_min_val_sent	ERC20_max_val_sent	ERC20_avg_val_sent	ERC20_min_val_sent_contract	ERC20_max_val_sent_contr
4681.000000	4681.000000	...	3.852000e+03	3.852000e+03	3.852000e+03	3.852000e+03	3.852000e+03	3852.0	
25.743004	49.220707	...	2.765535e+08	7.806912e+06	2.852135e+04	2.974736e+07	1.474142e+07	0.0	
248.704643	461.804628	...	1.611679e+10	3.256581e+08	1.614944e+06	1.804656e+09	9.046988e+08	0.0	
0.000000	0.000000	...	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.0	
0.000000	0.000000	...	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.0	
1.000000	0.044000	...	1.337000e+00	8.089090e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.0	
3.000000	1.000000	...	1.337000e+01	1.337000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.0	
6564.000000	10000.000000	...	1.000000e+12	1.724181e+10	1.000000e+08	1.120000e+11	5.614756e+10	0.0	

8 rows × 46 columns

	ERC20_min_val_sent	ERC20_max_val_sent	ERC20_avg_val_sent	ERC20_min_val_sent_contract	ERC20_max_val_sent_contract	ERC20_avg_val_sent_contract	ERC20_uniq_sent_token_name	ERC20_uniq_rec_token_name
+03	3.852000e+03	3.852000e+03	3.852000e+03	3852.0	3852.0	3852.0	3852.000000	3852.000000
+06	2.852135e+04	2.974736e+07	1.474142e+07	0.0	0.0	0.0	1.249221	4.550883
+08	1.614944e+06	1.804656e+09	9.046988e+08	0.0	0.0	0.0	6.293496	16.421821
+00	0.000000e+00	0.000000e+00	0.000000e+00	0.0	0.0	0.0	0.000000	0.000000
+00	0.000000e+00	0.000000e+00	0.000000e+00	0.0	0.0	0.0	0.000000	0.000000
-01	0.000000e+00	0.000000e+00	0.000000e+00	0.0	0.0	0.0	0.000000	1.000000
+01	0.000000e+00	0.000000e+00	0.000000e+00	0.0	0.0	0.0	0.000000	2.000000
+10	1.000000e+08	1.120000e+11	5.614756e+10	0.0	0.0	0.0	125.000000	435.000000

View the features' variances:

```
# Inspect features variance
df[numericals].var()
```

Output:

FLAG 2.488628e-01
Avg_min_between_sent_tnx 2.685019e+08
Avg_min_between_received_tnx 3.347336e+08
Time_Diff_between_first_and_last_(Mins) 5.744710e+10
Sent_tnx 5.392970e+05
Received_Tnx 1.302255e+06
Number_of_Created_Contracts 3.440827e+04
Unique_Received_From_Addresses 1.263905e+05
Unique_Sent_To_Addresses 6.185400e+04
min_value_received 2.132635e+05
max_value_received 3.267452e+08
avg_val_received 1.745874e+07
min_val_sent 3.653959e+04
max_val_sent 1.663396e+08
avg_val_sent 8.581648e+04
min_value_sent_to_contract 0.000000e+00
max_val_sent_to_contract 0.000000e+00
avg_value_sent_to_contract 0.000000e+00
total_transactions_(including_tnx_to_create_contract) 2.225163e+06
total_Ether_sent 7.636154e+10
total_ether_received 7.568166e+10
total_ether_sent_contracts 0.000000e+00
total_ether_balance 1.292148e+11
Total_ERC20_tnxs 6.784651e+04
ERC20_total_Ether_received 2.597794e+20
ERC20_total_ether_sent 3.256827e+18
ERC20_total_Ether_sent_contract 8.770285e+07
ERC20_uniq_sent_addr 1.688499e+03
ERC20_uniq_rec_addr 1.821469e+03
ERC20_uniq_sent_addr.1 5.166491e-03
ERC20_uniq_rec_contract_addr 2.840939e+02
ERC20_avg_time_between_sent_tnx 0.000000e+00

```
ERC20_avg_time_between_rec_tnx                0.000000e+00
ERC20_avg_time_between_rec_2_tnx              0.000000e+00
ERC20_avg_time_between_contract_tnx           0.000000e+00
ERC20_min_val_rec                             3.991777e+08
ERC20_max_val_rec                             2.597508e+20
ERC20_avg_val_rec                             1.060532e+17
ERC20_min_val_sent                            2.608043e+12
ERC20_max_val_sent                            3.256785e+18
ERC20_avg_val_sent                            8.184799e+17
ERC20_min_val_sent_contract                   0.000000e+00
ERC20_max_val_sent_contract                   0.000000e+00
ERC20_avg_val_sent_contract                   0.000000e+00
ERC20_uniq_sent_token_name                    3.960809e+01
ERC20_uniq_rec_token_name                     2.696762e+02
dtype: float64
```

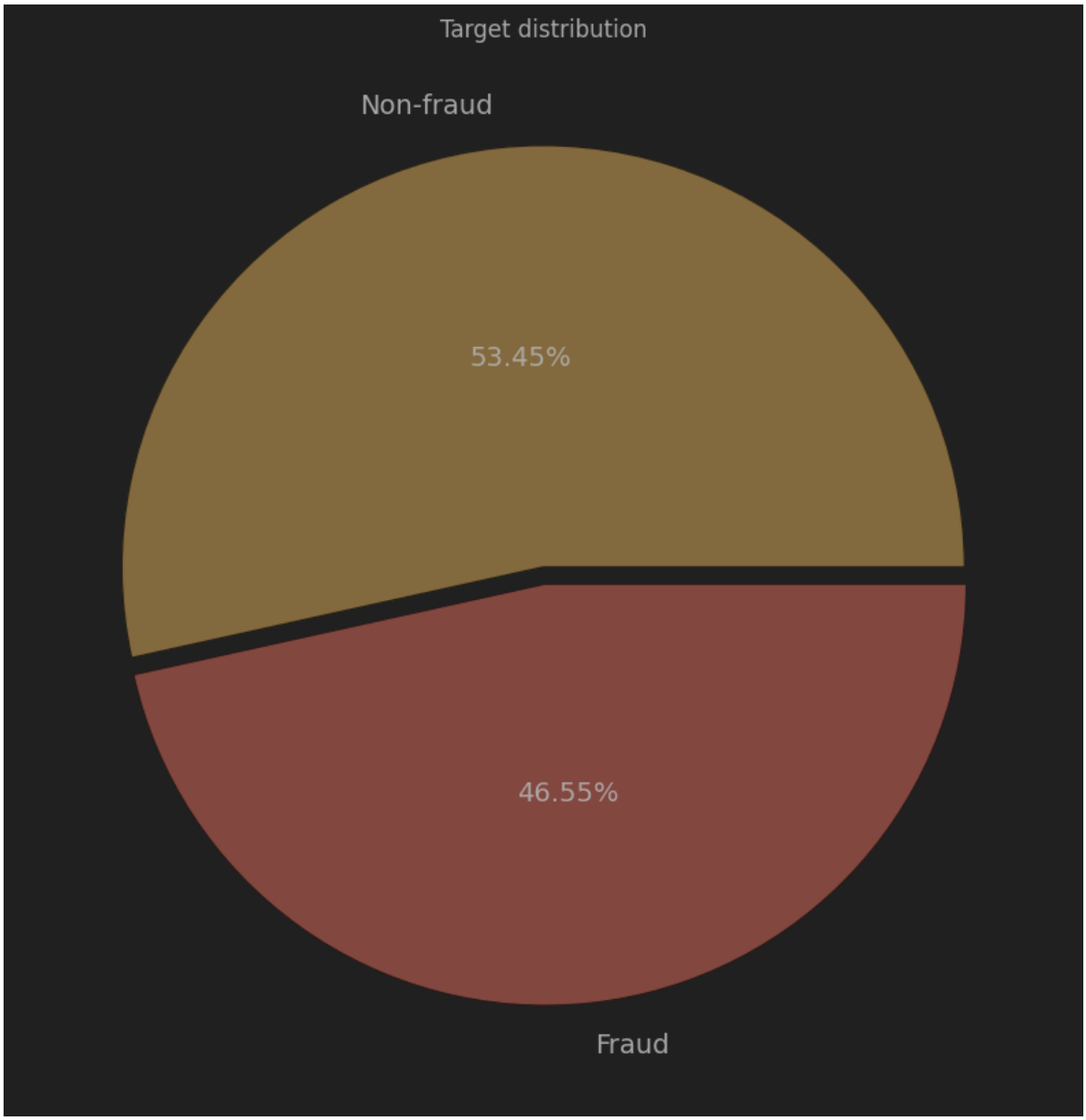
Then, view the target feature ‘FLAG’, to understood it’s distribution using pie chart:

```
# Inspect target distribution
print(df['FLAG'].value_counts())


pie, ax = plt.subplots(figsize=[15,10])
labels = ['Non-fraud', 'Fraud']
colors = ['#f9ae35', '#f64e38']
plt.pie(x = df['FLAG'].value_counts(), autopct='%.2f%%', explode=[0.02]*2, labels=labels,
pctdistance=0.5, textprops={'fontsize': 14}, colors = colors)
plt.title('Target distribution')
plt.show()
```

Output:

```
FLAG
0    2502
1    2179
```

```
Name: FLAG, dtype: int64
```

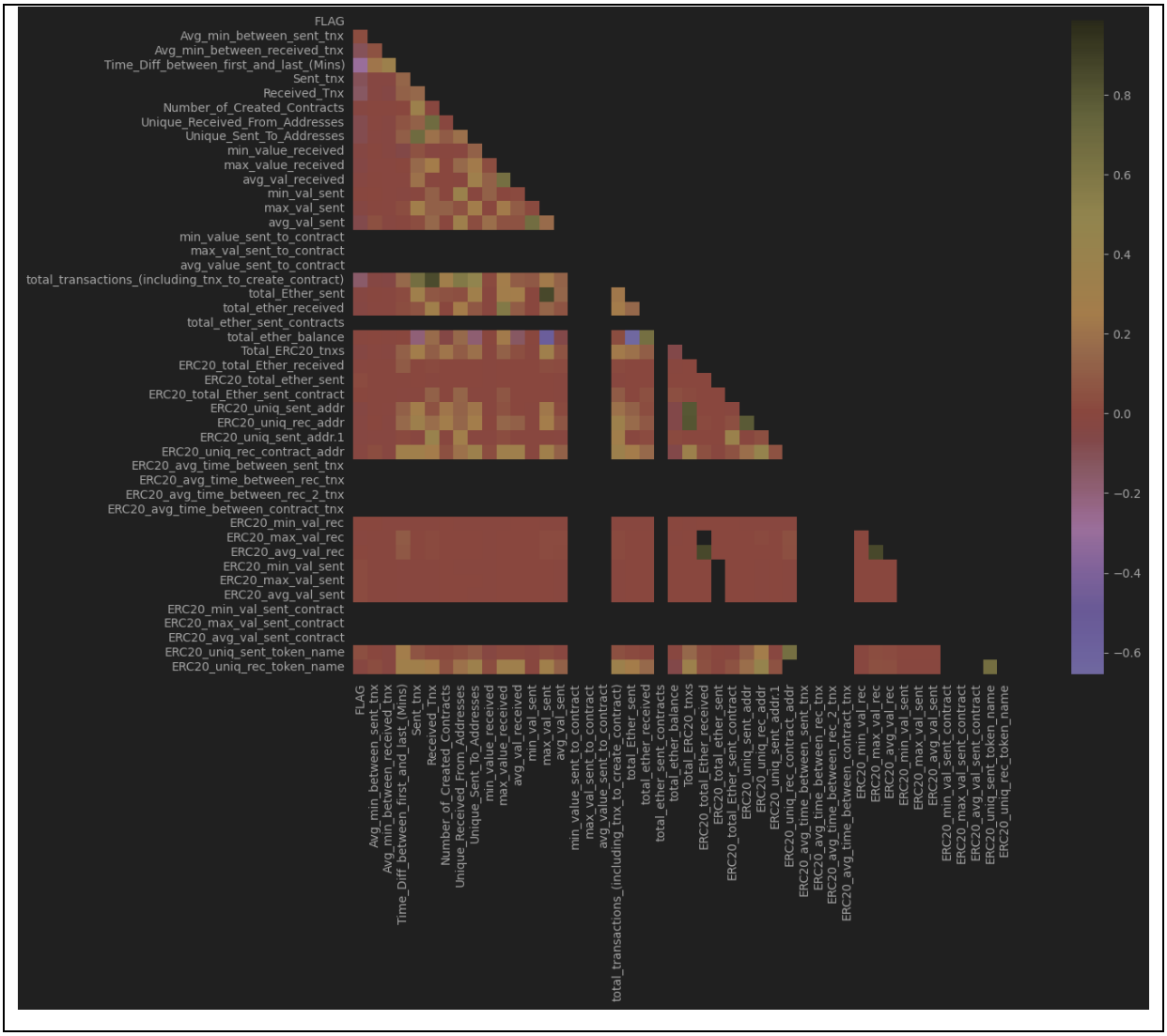


Then, check the Correlation Matrix of the dataset features to find out multicollinearity:

```
# Correlation matrix
corr = df.corr()

mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)]=True
with sns.axes_style('white'):
    fig, ax = plt.subplots(figsize=(18,10))
```

```
sns.heatmap(corr,  mask=mask, annot=False, cmap='CMRmap', center=0, square=True)
```

Output:

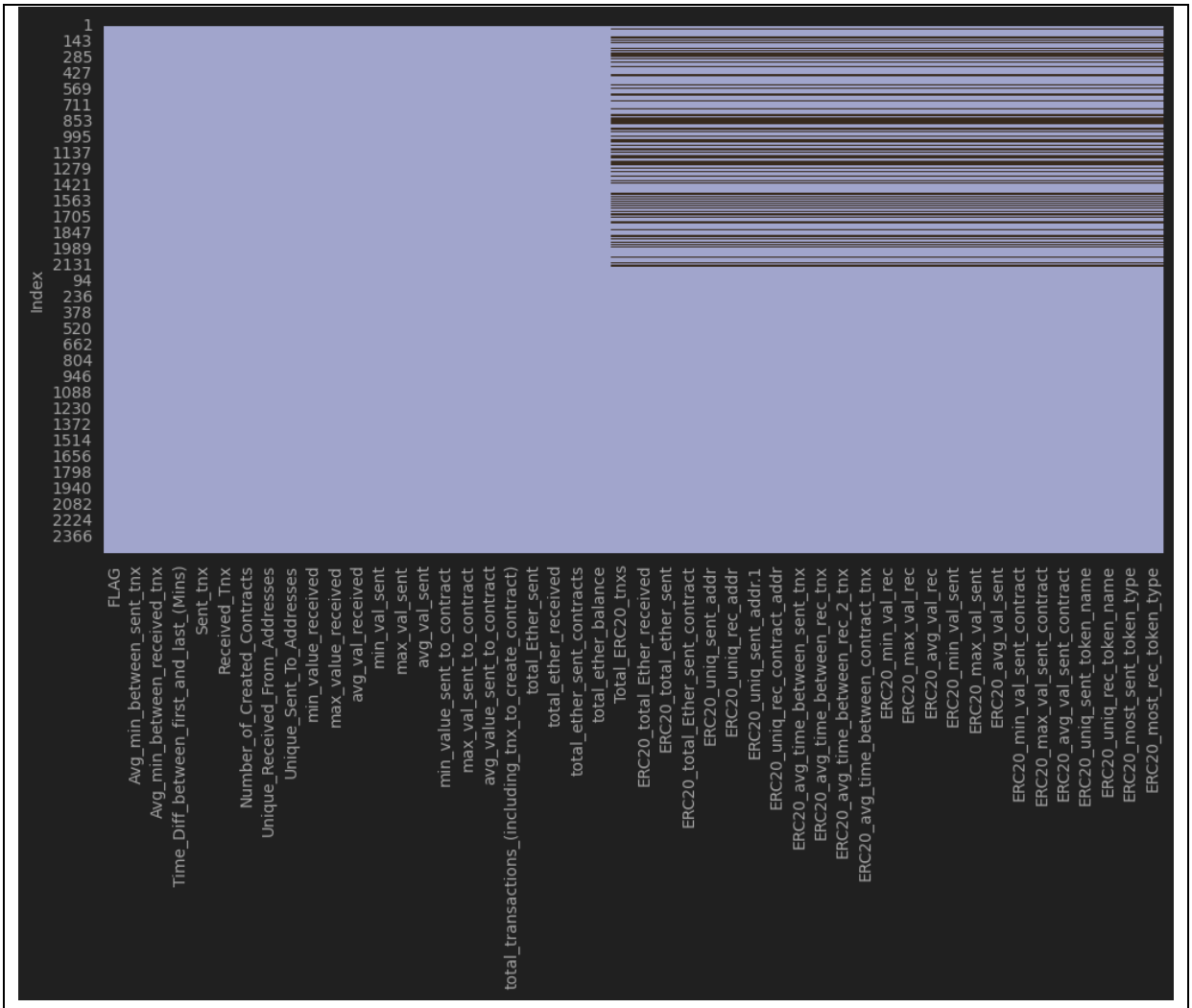


4.4 DATA CLEANING

Check dataset imbalance, normal distributions, outliers and correlations of the between features are important because those can affect the performance of the model.

```
# Visualize missing pattern of the data frame
plt.figure(figsize=(12,6))
sns.heatmap(df.isnull(), cbar=False)
plt.show()
```

Output:



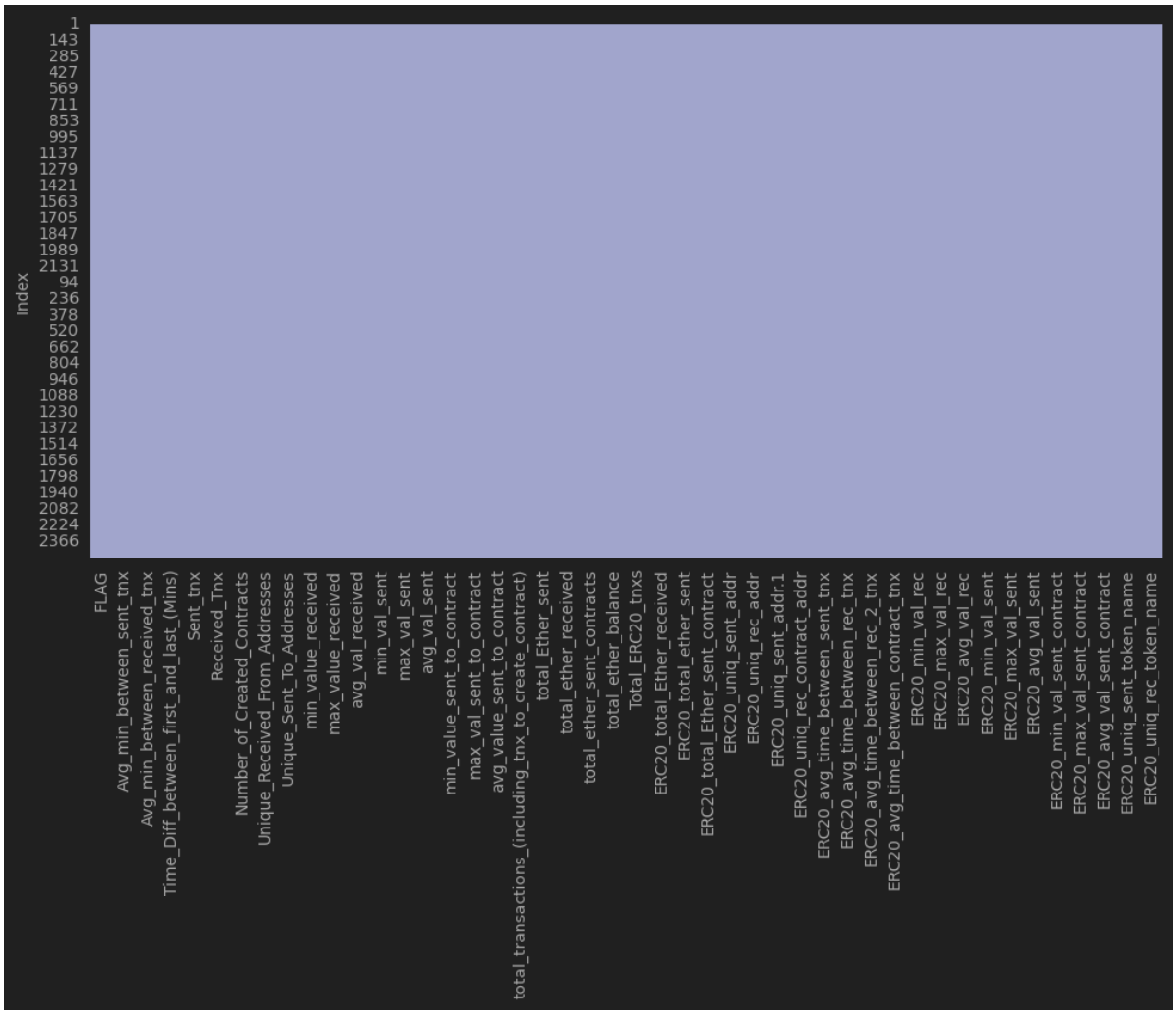
From the figure above, the data scientist has found out that there are some features have missing values. The missing values of numerical features can be replaced by data imputation. The data scientist has filled the values of numerical features with median.

```
# Drop the two categorical features
df.drop(df[categories], axis=1, inplace=True)
# Replace missing values of numerical variables with median
df.fillna(df.median(), inplace=True)
```

```
# Visualize missing values pattern of the dataframe
print(df.shape)
```

```
plt.figure(figsize=(12,6))
sns.heatmap(df.isnull(), cbar=False)
plt.show()
```

Output:



Then, the data scientist has investigated the variance of the features. It was observed that there are some features with a variance = 0. This is because features with zero variance have the same value for all samples in the dataset. Such features do not provide any discriminatory power for classification or regression tasks and can cause overfitting if included in the model. By dropping such features, we can simplify the model, reduce the noise and computation time, and avoid the risk of overfitting. Additionally, removing these features also makes the dataset

smaller and more manageable. Therefore, it's generally a good practice to remove features with zero variance before training a machine learning model.

```
# Filtering the features with 0 variance
no_var = df.var() == 0
print(df.var()[no_var])
print('\n')
```

```
# Drop features with 0 variance --- these features will not help in the performance of the model
df.drop(df.var()[no_var].index, axis = 1, inplace = True)
print(df.var())
print(df.shape)
```

Output:

```
min_value_sent_to_contract          0.0
max_val_sent_to_contract            0.0
avg_value_sent_to_contract          0.0
total_ether_sent_contracts          0.0
ERC20_avg_time_between_sent_tnx     0.0
ERC20_avg_time_between_rec_tnx      0.0
ERC20_avg_time_between_rec_2_tnx    0.0
ERC20_avg_time_between_contract_tnx 0.0
ERC20_min_val_sent_contract         0.0
ERC20_max_val_sent_contract         0.0
ERC20_avg_val_sent_contract         0.0
dtype: float64
```

```
FLAG                                2.488628e-01
Avg_min_between_sent_tnx            2.685019e+08
Avg_min_between_received_tnx        3.347336e+08
Time_Diff_between_first_and_last_(Mins)  5.744710e+10
Sent_tnx                            5.392970e+05
Received_Tnx                        1.302255e+06
```

```
Number_of_Created_Contracts                          3.440827e+04
Unique_Received_From_Addresses                       1.263905e+05
Unique_Sent_To_Addresses                             6.185400e+04
min_value_received                                   2.132635e+05
max_value_received                                   3.267452e+08
avg_val_received                                     1.745874e+07
min_val_sent                                         3.653959e+04
max_val_sent                                         1.663396e+08
avg_val_sent                                         8.581648e+04
total_transactions_(including_tnx_to_create_contract)  2.225163e+06
total_Ether_sent                                     7.636154e+10
total_ether_received                                 7.568166e+10
total_ether_balance                                  1.292148e+11
Total_ERC20_tnxs                                     5.589666e+04
ERC20_total_Ether_received                           2.137745e+20
ERC20_total_ether_sent                               2.680054e+18
ERC20_total_Ether_sent_contract                      7.217687e+07
ERC20_uniq_sent_addr                                 1.390419e+03
ERC20_uniq_rec_addr                                  1.502337e+03
ERC20_uniq_sent_addr.1                               4.255245e-03
ERC20_uniq_rec_contract_addr                         2.356759e+02
ERC20_min_val_rec                                    3.285447e+08
ERC20_max_val_rec                                    2.137506e+20
ERC20_avg_val_rec                                    8.727617e+16
ERC20_min_val_sent                                   2.146181e+12
ERC20_max_val_sent                                   2.680017e+18
ERC20_avg_val_sent                                   6.735287e+17
ERC20_uniq_sent_token_name                           3.281952e+01
ERC20_uniq_rec_token_name                            2.237446e+02
dtype: float64
(4681, 35)
```

So now, let’s check the remaining features of the dataset:

```
df.info()
```

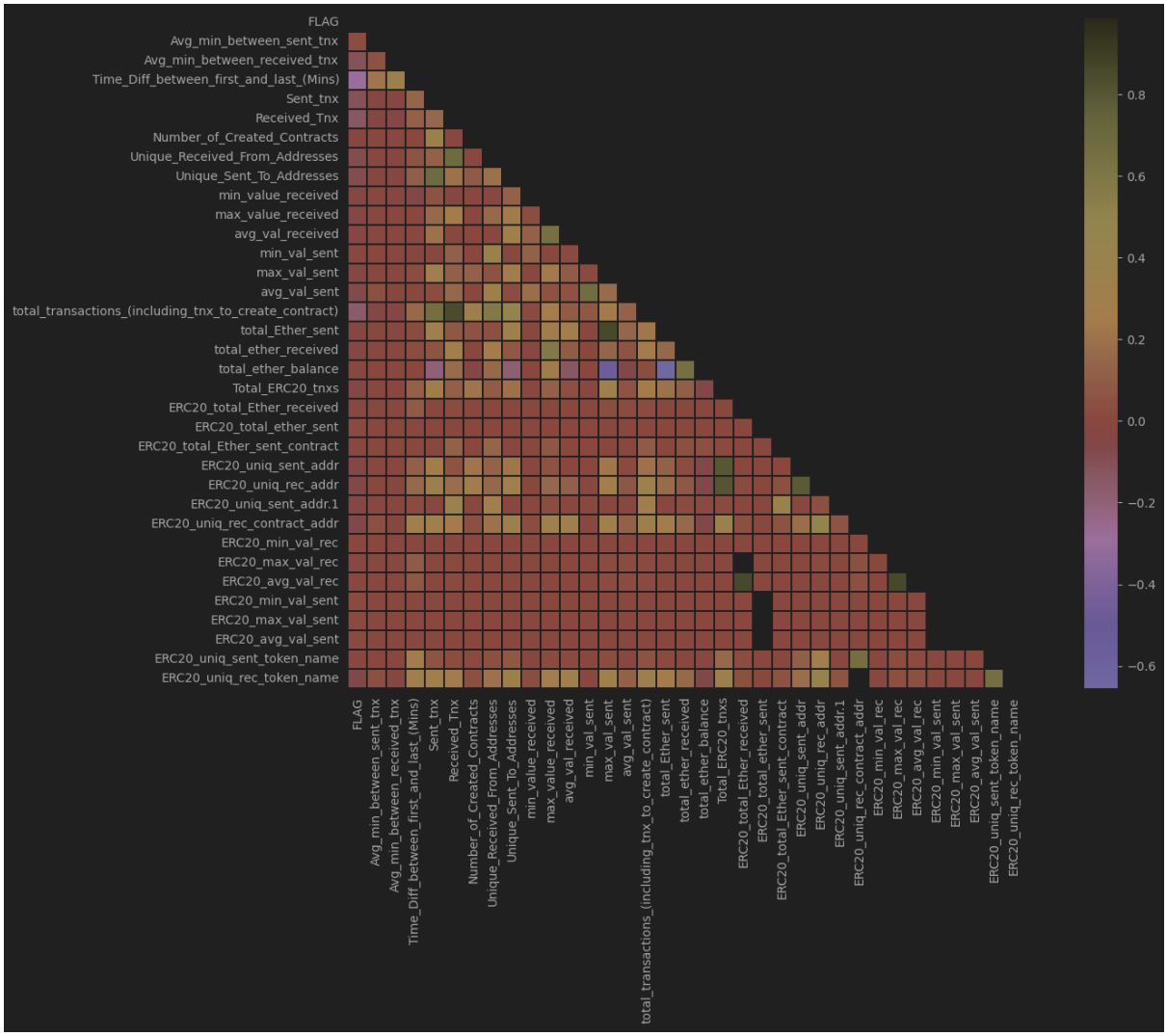
Output:

```
<class 'pandas.core.frame.DataFrame'>
Index: 4681 entries, 1 to 2502
Data columns (total 35 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   FLAG                                          4681 non-null   int64
 1   Avg_min_between_sent_tnx                      4681 non-null   float64
 2   Avg_min_between_received_tnx                  4681 non-null   float64
 3   Time_Diff_between_first_and_last_(Mins)       4681 non-null   float64
 4   Sent_tnx                                      4681 non-null   int64
 5   Received_Tnx                                  4681 non-null   int64
 6   Number_of_Created_Contracts                   4681 non-null   int64
 7   Unique_Received_From_Addresses                4681 non-null   int64
 8   Unique_Sent_To_Addresses                      4681 non-null   int64
 9   min_value_received                            4681 non-null   float64
 10  max_value_received                            4681 non-null   float64
 11  avg_val_received                              4681 non-null   float64
 12  min_val_sent                                  4681 non-null   float64
 13  max_val_sent                                  4681 non-null   float64
 14  avg_val_sent                                  4681 non-null   float64
 15  total_transactions_(including_tnx_to_create_contract)  4681 non-null   int64
 16  total_Ether_sent                              4681 non-null   float64
 17  total_ether_received                          4681 non-null   float64
 18  total_ether_balance                           4681 non-null   float64
 19  Total_ERC20_tnxs                              4681 non-null   float64
 20  ERC20_total_Ether_received                    4681 non-null   float64
 21  ERC20_total_ether_sent                        4681 non-null   float64
 22  ERC20_total_Ether_sent_contract               4681 non-null   float64
 23  ERC20_uniq_sent_addr                          4681 non-null   float64
```

```
 24  ERC20_uniq_rec_addr                4681 non-null   float64
 25  ERC20_uniq_sent_addr.1             4681 non-null   float64
 26  ERC20_uniq_rec_contract_addr       4681 non-null   float64
 27  ERC20_min_val_rec                  4681 non-null   float64
 28  ERC20_max_val_rec                  4681 non-null   float64
 29  ERC20_avg_val_rec                  4681 non-null   float64
 30  ERC20_min_val_sent                 4681 non-null   float64
 31  ERC20_max_val_sent                 4681 non-null   float64
 32  ERC20_avg_val_sent                 4681 non-null   float64
 33  ERC20_uniq_sent_token_name         4681 non-null   float64
 34  ERC20_uniq_rec_token_name          4681 non-null   float64
dtypes: float64(28), int64(7)
memory usage: 1.3 MB
```

Let’s check the multicollinearity again:

```
# Recheck the Correlation matrix
corr = df.corr()

mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)]=True
with sns.axes_style('white'):
   fig, ax = plt.subplots(figsize=(18,10))
   sns.heatmap(corr,  mask=mask, annot=False, cmap='CMRmap', center=0, linewidths=0.1,
square=True)
```



Print the existing feature to confirm deletion:

```
print("Existing columns:", df.columns)
```

Output:

```
Existing columns: Index(['FLAG', 'Avg_min_between_sent_tnx',
'Avg_min_between_received_tnx',
    'Time_Diff_between_first_and_last_(Mins)', 'Sent_tnx', 'Received_Tnx',
    'Number_of_Created_Contracts', 'Unique_Received_From_Addresses',
    'Unique_Sent_To_Addresses', 'min_value_received', 'max_value_received',
    'avg_val_received', 'min_val_sent', 'max_val_sent', 'avg_val_sent',
```

```
    'total_transactions_(including_tnx_to_create_contract)',
    'total_Ether_sent', 'total_ether_received', 'total_ether_balance',
    'Total_ERC20_tnxs', 'ERC20_total_Ether_received',
    'ERC20_total_ether_sent', 'ERC20_total_Ether_sent_contract',
    'ERC20_uniq_sent_addr', 'ERC20_uniq_rec_addr', 'ERC20_uniq_sent_addr.1',
    'ERC20_uniq_rec_contract_addr', 'ERC20_min_val_rec',
    'ERC20_max_val_rec', 'ERC20_avg_val_rec', 'ERC20_min_val_sent',
    'ERC20_max_val_sent', 'ERC20_avg_val_sent',
    'ERC20_uniq_sent_token_name', 'ERC20_uniq_rec_token_name'],
   dtype='object')
```

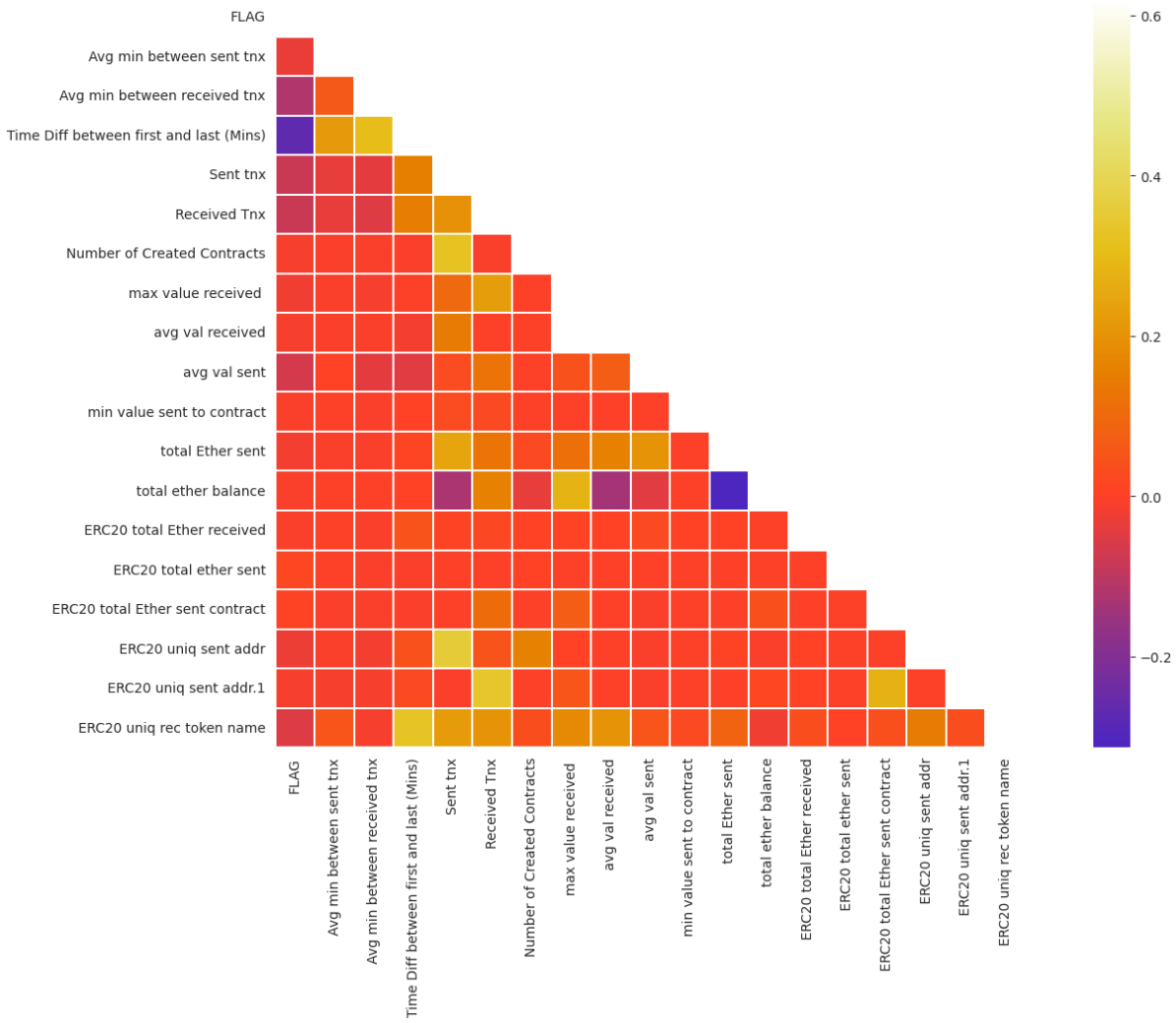
Remove the features that have high correlations:

```
drop = ['total transactions (including tnx to create contract', 'total ether sent contracts', 'max val sent to contract', ' ERC20 avg val rec',
    ' ERC20 avg val rec',' ERC20 max val rec', ' ERC20 min val rec', ' ERC20 uniq rec contract addr', 'max val sent', ' ERC20 avg val sent',
    ' ERC20 min val sent', ' ERC20 max val sent', ' Total ERC20 tnxs', 'avg value sent to contract', 'Unique Sent To Addresses',
    'Unique Received From Addresses', 'total ether received', ' ERC20 uniq sent token name', 'min value received', 'min val sent', ' ERC20 uniq rec addr' ]
df.drop(drop, axis=1, inplace=True)
```

Then, check the Correlation matrix visualization again:

```
# Recheck the Correlation matrix
corr = df.corr()

mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)]=True
with sns.axes_style('white'):
  fig, ax = plt.subplots(figsize=(18,10))
```

```
sns.heatmap(corr, mask=mask, annot=False, cmap='CMRmap', center=0, linewidths=0.1, square=True)
```



Show the remaining columns:

```
columns = df.columns
columns
```

Output:

```
Index(['FLAG', 'Avg min between sent tnx', 'Avg min between received tnx',
       'Time Diff between first and last (Mins)', 'Sent tnx', 'Received Tnx',
       'Number of Created Contracts', 'max value received ',
       'avg val received', 'avg val sent', 'min value sent to contract',
       'total Ether sent', 'total ether balance',
```

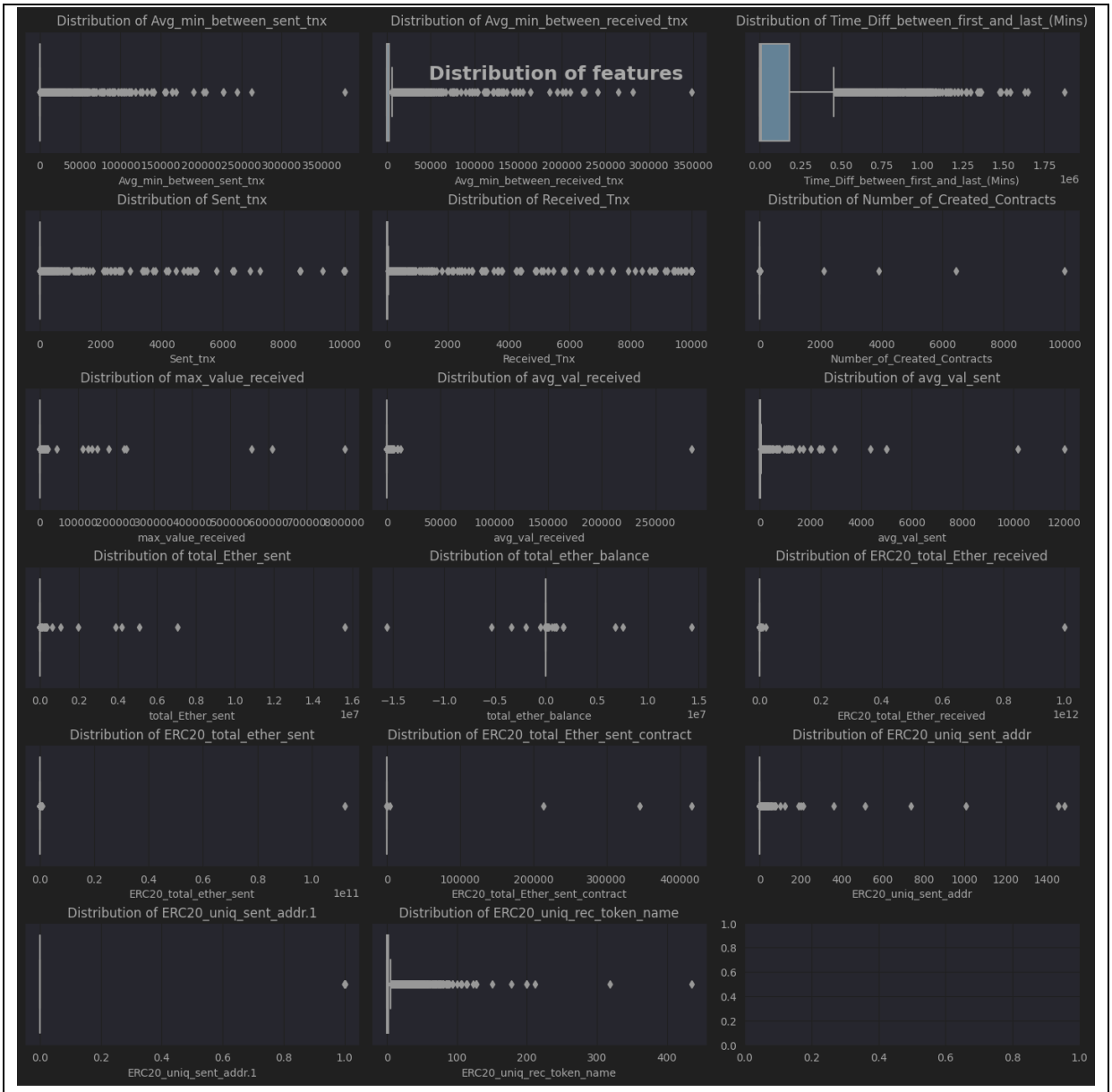
```
 ' ERC20 total Ether received', ' ERC20 total ether sent',
 ' ERC20 total Ether sent contract', ' ERC20 uniq sent addr',
 ' ERC20 uniq sent addr.1', ' ERC20 uniq rec token name'],
dtype='object')
```

4.4.1 DATA DISTRIBUTION

The data distribution of the features is heavily skewed with multiple outliers. Hence, data transformation is needed to normalize the values using the following code:

```
# Investigate the distribution of our features using box plots
fig, axes = plt.subplots(6, 3, figsize=(14, 14), constrained_layout=True)
plt.subplots_adjust(wspace=0.7, hspace=0.8)
plt.suptitle("Distribution of features", y=0.95, size=18, weight='bold')

for i, column in enumerate(columns[1:]):
    row = i // 3
    col = i % 3
    ax = sns.boxplot(ax=axes[row, col], data=df, x=column)
    ax.set_title(f'Distribution of {column}')

plt.show()
```

Output:




```
# Some features present a small distribution
for i in df.columns[1:]:
    if len(df[i].value_counts()) < 10:
        print(f'The column {i} has the following distribution: \n{df[i].value_counts()}')
        print('=====================================')
```

Output:

```
The column  ERC20 uniq sent addr.1 has the following distribution:
0.0    9813
```

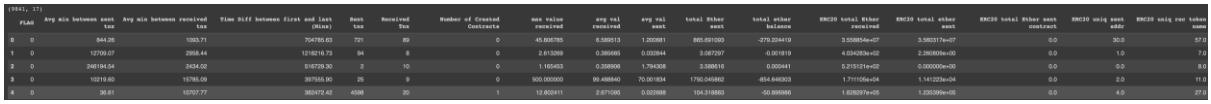
```
1.0     26
3.0      1
2.0      1
Name:  ERC20 uniq sent addr.1, dtype: int64
========================================
```

Based on the output, it can be observed that the values of these two features are mostly 0s. Thus, both features will be discarded since they will not be helpful for our model.

```
drops = ['ERC20_uniq_sent_addr']
df.drop(drops, axis=1, inplace=True)
print(df.shape)
df.head()
```

Output:

```
(4681, 17)
```



	FLAG	Avg min between sent tnx	Avg min between received tnx	Time Diff between first and last (Mins)	Sent tnx	Received Tnx	Number of Created Contracts	max value received	avg val received	avg val sent	total Ether sent	total ether balance	ERC20 total Ether received	ERC20 total ether sent	ERC20 total Ether sent contract	ERC20 uniq sent addr	ERC20 uniq rec token name
0	0	844.26	1093.71	704785.63	721	89	0	45.806785	6.589513	1.200681	865.691093	-279.224419	3.558854e+07	3.560317e+07	0.0	30.0	57.0
1	0	12709.07	2958.44	1218216.73	94	8	0	2.613269	0.389585	0.032844	3.087297	-0.001819	4.034283e+02	2.280809e+05	0.0	1.0	7.0
2	0	246194.54	2434.02	516729.30	2	10	0	1.165453	0.358906	1.794308	3.589616	0.000441	5.215121e+02	0.000000e+00	0.0	0.0	9.0
3	0	10219.80	15785.09	397555.90	25	9	0	500.000000	99.488840	70.001834	1750.045862	-854.640303	1.711105e+04	1.141223e+04	0.0	2.0	11.0
4	0	36.61	10707.77	382472.42	4598	20	1	12.800411	2.671095	0.020988	104.318883	-60.898886	1.628297e+05	1.235399e+05	0.0	4.0	27.0

4.5 MODEL PREPARATION

Next, the data scientist prepares the dataset into training and validation datasets to be used for modelling of different algorithms.

```
y = df.iloc[:, 0]
X = df.iloc[:, 1:]
print(X.shape, y.shape)
```

Output:

```
(4681, 16) (4681,)
```

4.5.1 DATA SPLITTING

Then, split the dataset into 80:20 distribution.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PowerTransformer
from imblearn.over_sampling import SMOTE
```

```
# Split into training (80%) and testing set (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 123)
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

Output:

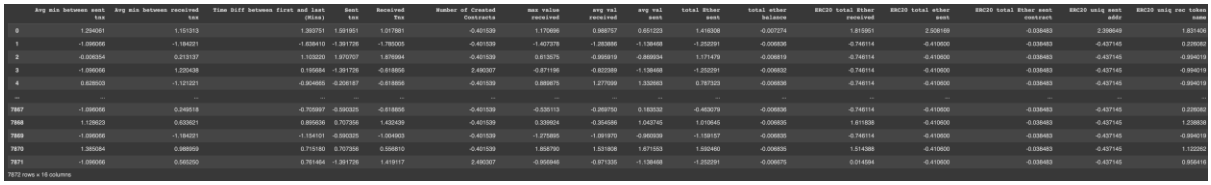
```
(3744, 16) (3744,)
(937, 16) (937,)
```

4.5.2 DATA NORMALIZATION

PowerTransformer is used for data distribution that has heavy skewed or tails.

```
# Normalize the training features
norm = PowerTransformer()
norm_train_f = norm.fit_transform(X_train)

# Transform test features
norm_test_f = norm.transform(X_test)

norm_df = pd.DataFrame(norm_train_f, columns=X_train.columns)
norm_df
```

Output:



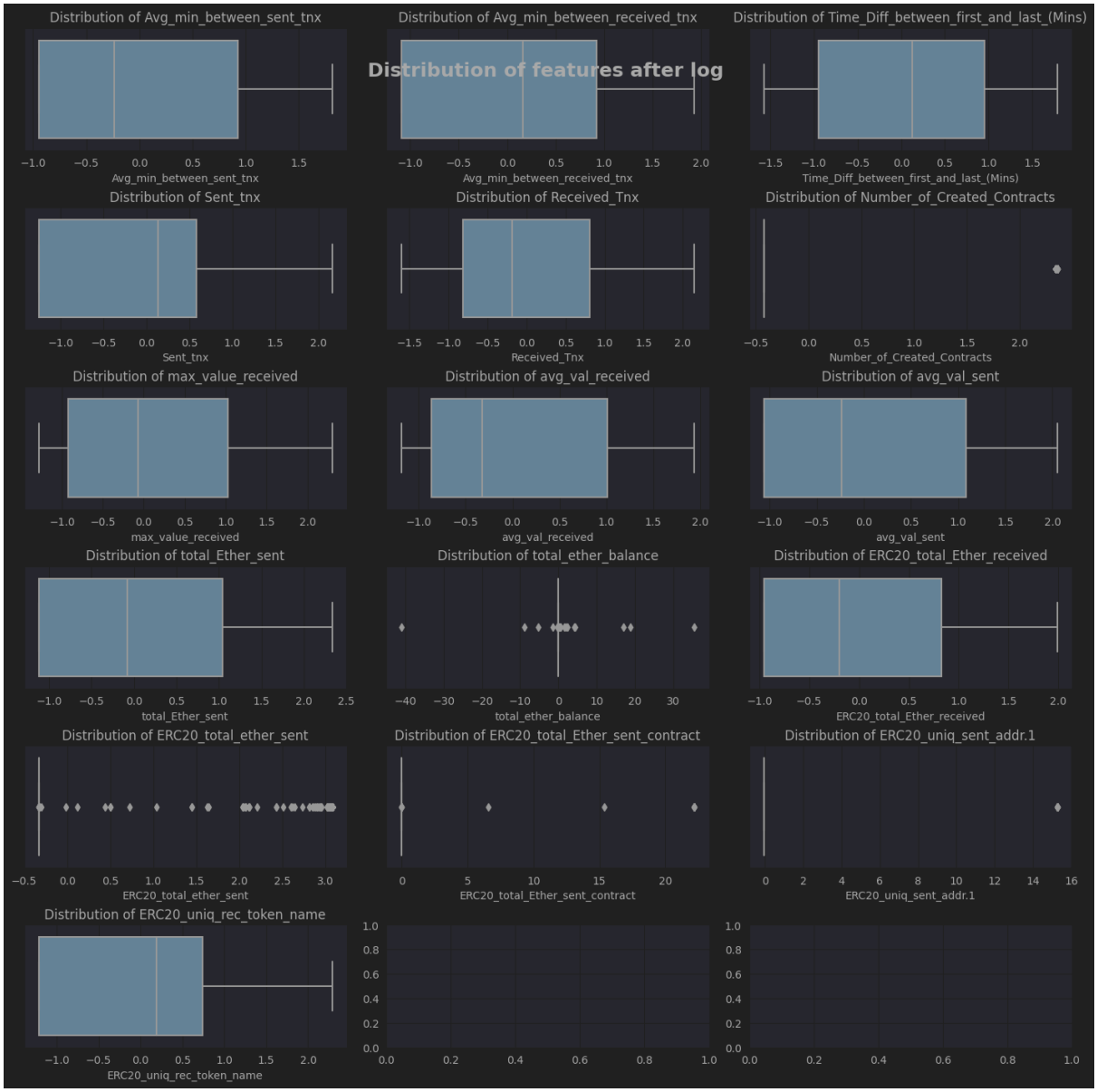
	Avg min between sent tnx	Avg min between received tnx	Time Diff between first and last (Mins)	Sent tnx	Received Tnx	Number of Created Contracts	max value received	avg val received	avg val sent	total Ether sent	total ether balance	ERC20 total Ether received	ERC20 total ether sent	ERC20 total Ether sent contract	ERC20 uniq sent addr	ERC20 uniq rec token name
0	1.294061	1.151313	1.393751	1.591951	1.017881	-0.401539	1.170696	0.988757	0.651223	1.416308	-0.007274	1.815951	2.506169	-0.038483	2.398649	1.831406
1	-1.099056	-1.184221	-1.639410	-1.391726	-1.785005	-0.401539	-1.407378	-1.283886	-1.136468	-1.252291	-0.006836	-0.746114	-0.410600	-0.038483	-0.437145	0.239082
2	-0.008354	0.213137	1.103220	1.970707	1.876994	-0.401539	0.813575	-0.995919	-0.869934	1.171479	-0.006819	-0.746114	-0.410600	-0.038483	-0.437145	-0.994019
3	-1.099056	1.230438	0.199584	-1.391726	-0.818856	2.490307	-0.871196	-0.822389	-1.136468	-1.252291	-0.006832	-0.746114	-0.410600	-0.038483	-0.437145	-0.994019
4	0.628503	-1.121221	-0.904665	-0.206187	-0.818856	-0.401539	0.889675	1.277099	1.332983	0.787323	-0.006836	-0.746114	-0.410600	-0.038483	-0.437145	-0.994019
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7867	-1.099056	0.249518	-0.705897	-0.590325	-0.818856	-0.401539	-0.535113	-0.269750	0.183532	-0.463079	-0.006836	-0.746114	-0.410600	-0.038483	-0.437145	0.239082
7868	1.128623	0.633621	0.895836	0.707356	1.432439	-0.401539	0.339924	-0.354586	1.043745	1.010645	-0.006835	1.611838	-0.410600	-0.038483	-0.437145	1.238838
7869	-1.099056	-1.184221	-1.154101	-0.590325	-1.004903	-0.401539	-1.275895	-1.091970	-0.969939	-1.159157	-0.006835	-0.746114	-0.410600	-0.038483	-0.437145	-0.994019
7870	1.385084	0.966959	0.715180	0.707356	0.556810	-0.401539	1.656790	1.531808	1.671553	1.552460	-0.006835	1.514368	-0.410600	-0.038483	-0.437145	1.122252
7871	-1.099056	0.565250	0.761464	-1.391726	1.419117	2.490307	-0.956946	-0.971335	-1.136468	-1.252291	-0.006675	0.014594	-0.410600	-0.038483	-0.437145	0.956416

7872 rows × 16 columns

Check the data distribution after the log transformation:

```
# Investigate the distribution of our features using boxplots
fig, axes = plt.subplots(6, 3, figsize=(14, 14), constrained_layout=True)
plt.subplots_adjust(wspace=0.7, hspace=0.8)
plt.suptitle("Distribution of features after log",y=0.95, size=18, weight='bold')

for i, column in enumerate(norm_df[1:]):
    row = i // 3
    col = i % 3
    ax = sns.boxplot(ax=axes[row, col], data=norm_df, x=column)
    ax.set_title(f'Distribution of {column}')

plt.show()
```

Output:



4.5.3 DATA BALANCING

The distribution of categorical value of the feature may be imbalanced, that it affects the model performance. The minority class is majorly overwhelmed by majority class, and the model may be biased towards with majority class, leads to poor performance on the minority class, means the model learned nothing about the minority class. Hence, it's important to fix the issue by either perform oversampling or under sampling.

```
oversample = SMOTE()
print(f'Shape of the training before SMOTE: {norm_train_f.shape, y_train.shape}')
```

```
x_tr_resample, y_tr_resample = oversample.fit_resample(norm_train_f, y_train)
print(f'Shape of the training after SMOTE: {x_tr_resample.shape, y_tr_resample.shape}')
```

Output:

```
Shape of the training before SMOTE: ((3744, 16), (3744,))
Shape of the training after SMOTE: ((4018, 16), (4018,))
```

Check the distribution of the two classes:

```
# Target distribution before SMOTE
non_fraud = 0
fraud = 0

for i in y_train:
    if i == 0:
        non_fraud +=1
    else:
        fraud +=1

# Target distribution after SMOTE
no = 0
yes = 1

for j in y_tr_resample:
    if j == 0:
        no +=1
    else:
        yes +=1



print(f'BEFORE OVERSAMPLING \n \tNon-frauds: {non_fraud} \n \tFrauds: {fraud}')
print(f'AFTER OVERSAMPLING \n \tNon-frauds: {no} \n \tFrauds: {yes}')
```

Output:

BEFORE OVERSAMPLING

Non-frauds: 2009

Frauds: 1735

AFTER OVERSAMPLING

Non-frauds: 2009

Frauds: 2010

4.6 MODELLING

This stage involves generate a model. There are a lot of existing techniques used by the researchers based on the literature reviews in the previous sections.

The researcher introduced the Grid Search Hyperparameter for the selected models with multiple cross-validation folds.
There are several reasons why Grid Search Hyperparameters are introduced:

- **Automation**: Manually tuning hyperparameters can be a time-consuming and error-prone process. Grid Search Hyperparameters automates this process and finds the best combination of hyperparameters to improve the model's performance.
- **Performance improvement**: Hyperparameters have a significant impact on a model's performance. Grid Search Hyperparameters can help in selecting the best combination of hyperparameters that can lead to improved performance.
- **Generalization**: Hyperparameters that are optimized for a specific dataset may not work well on other datasets. Grid Search Hyperparameters can help in finding the optimal hyperparameters that work well on different datasets.
- **Efficiency**: Grid Search Hyperparameters is an efficient approach to finding the best hyperparameters, as it systematically explores all possible combinations of hyperparameters.

This technique requires using GridSearchCV function that tries to speed up the hyperparameter search process by removing the other possibilities when tuning hyperparameters. It uses a technique called "grid search" to exhaustively search through all the possible combinations of hyperparameters, but it uses a smart approach to search only through the combinations that are likely to be useful.

It also performs cross-validations on each combination of hyperparameters and only selecting those combinations that perform the best. It proceeds the search with these combinations until it finds the best hyperparameters.

However, despite its intelligent search approach, GridSearchCV can still be computationally expensive, especially when the search space is large. The data scientist has to choose the hyperparameters to search and limit the search space to avoid unnecessary computations.

For DNN, the architecture contains dropout layers and early stopping to disable dropout layer and prevent overfitting.

4.6.1 VISUALIZATIONS & PERFORMANCE METRICS

First, import the libraries that are required for perform model building, along with visualizations (plots) and performance metrics:

```
from sklearn.model_selection import cross_val_score, GridSearchCV,
RepeatedStratifiedKFold
from sklearn.metrics import confusion_matrix, roc_auc_score, roc_curve, auc,
classification_report, ConfusionMatrixDisplay, average_precision_score,
precision_recall_curve, accuracy_score, precision_score, recall_score, f1_score
```

```
def evaluate_model(name, model, norm_test_f, y, cv):

    print("Model name: ", name)

    # Perform cross-validation
    if name != "DNN":
        cv_scores = cross_val_score(model, norm_test_f, y, cv=cv, scoring='accuracy', n_jobs=-
1)
        # Print cross-validation scores
        print("Cross-Validation Scores:", cv_scores)
        print("Mean Cross-Validation Score:", cv_scores.mean())
    else:
        print('Skipping cross validation')

    # Predict using the model
    preds = model.predict(norm_test_f)
    # Make predictions using the final model
    # final_y_pred = final_model.predict(norm_test_f)

    if name == "DNN":
```

```
    show_model_performance(norm_test_f, y_test, model)
else:
    print('Skipping model performance, loss and accuracy plots')

# Convert predicted probabilities to binary predictions
final_y_pred_binary = np.where(preds > 0.5, 1, 0)

if name == "DNN":
    preds = final_y_pred_binary

# Calculate accuracy
accuracy = accuracy_score(y_test, preds)
print("Accuracy:", accuracy)

# Calculate precision
precision = precision_score(y_test, preds)
print("Precision:", precision)

# Calculate recall
recall = recall_score(y_test, preds)
print("Recall:", recall)

# Calculate F1-score
f1 = f1_score(y_test, preds)
print("F1-score:", f1)

if name == "DNN":
    show_confusion_matrix(preds)
    show_roc_curve_plot(preds)
    show_precision_recall_curve_plot(preds)
else:
    show_confusion_matrix(final_y_pred_binary)
    show_roc_curve_plot(final_y_pred_binary)
    show_precision_recall_curve_plot(final_y_pred_binary)
```

```
def show_model_performance(norm_test_f, y_test, model):
  # Evaluate the model
  lost, accuracy = model.evaluate(norm_test_f, y_test, verbose=0)
  print('Loss:', lost)
  print('Accuracy:', accuracy)

def show_loss_and_accuracy_line_plot(history):
  # Plot the loss and accuracy over epochs
  plt.plot(history.history['loss'])
  plt.plot(history.history['val_loss'])
  plt.title('Model loss')
  plt.ylabel('Loss')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Validation'], loc='upper left')
  plt.show()

  plt.plot(history.history['accuracy'])
  plt.plot(history.history['val_accuracy'])
  plt.title('Model accuracy')
  plt.ylabel('Accuracy')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Validation'], loc='upper left')
  plt.show()

def show_confusion_matrix(preds):
    # Print classification report
    print(classification_report(y_test, preds))

    # Create the confusion matrix
    cm = confusion_matrix(y_test, preds)
    print(cm)

    # Create the display object
```

```
    # disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=np.unique(y_test))
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)

    # Plot the confusion matrix
    disp.plot()

def show_roc_curve_plot(y_pred):

    # Calculate AUC-ROC score
    auc_roc = roc_auc_score(y_test, y_pred)
    print("AUC-ROC Score:", auc_roc)

    # Calculate false positive rate (FPR), true positive rate (TPR), and thresholds
    fpr, tpr, thresholds = roc_curve(y_test, y_pred)

    roc_auc = auc(fpr, tpr)

    # Plot ROC curve
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = %0.2f)' % auc_roc)
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc="lower right")
    plt.show()


def show_precision_recall_curve_plot(y_pred):
  # Compute Precision-Recall curve and average precision for each class
  precision, recall, _ = precision_recall_curve(y_test, y_pred)
```

```
average_precision = average_precision_score(y_test, y_pred)

# Plot Precision-Recall curve
plt.figure(figsize=(8, 6))
plt.step(recall, precision, color='b', alpha=0.2, where='post')
plt.fill_between(recall, precision, alpha=0.2, color='b')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('Precision-Recall curve: AP={0:0.2f}'.format(average_precision))
plt.show()
```

From the above functions, there are multiple performance metrics that are calculated with the existing libraries to ensure the model results are consistent and accurate. In the main function, they will check for the model name, and calculate the following:

- Cross validation scores
- Loss and Accuracy plot
- Accuracy
- Precision
- Recall
- F1 score
- Confusion Matrix
- AUC-ROC plot
- Precision-Recall Curve

With these metrics, the data scientist can produce results that can be referenced by future researchers with similar domain of research.

4.6.2 LOGISTIC REGRESSION (LR)

```
from sklearn.linear_model import LogisticRegression
```

```
# Define hyperparameters grid for tuning
lr_param_grid = {
    'penalty': ['l1', 'l2'],  # Regularization penalty
    'C': [0.001, 0.01, 0.1, 1, 10, 100],  # Regularization parameter
    'solver': ['liblinear'],  # Solver for optimization problem
    'class_weight': [None, 'balanced'],  # Class weights to handle imbalance
    'max_iter': [100, 200, 300],  # Maximum number of iterations
    'tol': [1e-4, 1e-3, 1e-2]  # Tolerance for stopping criteria
}
```

```
# Define different numbers of folds for cross-validation
num_folds = [3, 4, 5, 10]

for folds in num_folds:
    # Create RepeatedStratifiedKFold with the specified number of folds
    cv = RepeatedStratifiedKFold(n_splits=folds, n_repeats=3, random_state=42)



    # Define the logistic regression model
    LR = LogisticRegression(random_state=42)

    # Perform grid search with repeated cross-validation
    grid_search = GridSearchCV(estimator=LR, param_grid=lr_param_grid, cv=cv, scoring='accuracy', n_jobs=-1)
    grid_search.fit(x_tr_resample, y_tr_resample)

    # Get the best model from grid search
    best_LR = grid_search.best_estimator_

    # Print the best parameters found
```

```
print(f"\nNumber of Folds: {folds}")
print("Best Parameters:", grid_search.best_params_)

# Use the best parameters to build the model
best_params = grid_search.best_params_
best_lr_model = LogisticRegression(random_state=42, **best_params)

# Fit the model on the training data with the best parameters
best_lr_model.fit(x_tr_resample, y_tr_resample)

# Evaluate the best model
evaluate_model("Logistic Regression", best_LR, norm_test_f, y_test, cv)
```

From the above, a comprehensive script has been developed to tune the model of Logistic Regression with multiple cross validation in mind. The hyperparameters are produced with the limitation of the current hardware resource to find out the best parameters of the model. After the best parameter of the model is acquired from each cross validation, the performance model will be evaluated and visualized.

Output:

```
Best Parameters: {'C': 10, 'class_weight': 'balanced', 'max_iter': 100, 'penalty': 'l1', 'solver': 'liblinear', 'tol': 0.001}
Model name:  Logistic Regression
Cross-Validation Scores: [0.91054313 0.8974359  0.89102564 0.90095847 0.8525641 0.92948718
 0.89776358 0.91666667 0.88782051]
Mean Cross-Validation Score: 0.8982516861909833
Skipping model performance, loss and accuracy plots
Accuracy: 0.9124866595517609
Precision: 0.8969298245614035
Recall: 0.9211711711711712
F1-score: 0.9088888888888889
              precision    recall  f1-score   support
```

0 0.93 0.90 0.92 493

1 0.90 0.92 0.91 444

accuracy 0.91 937

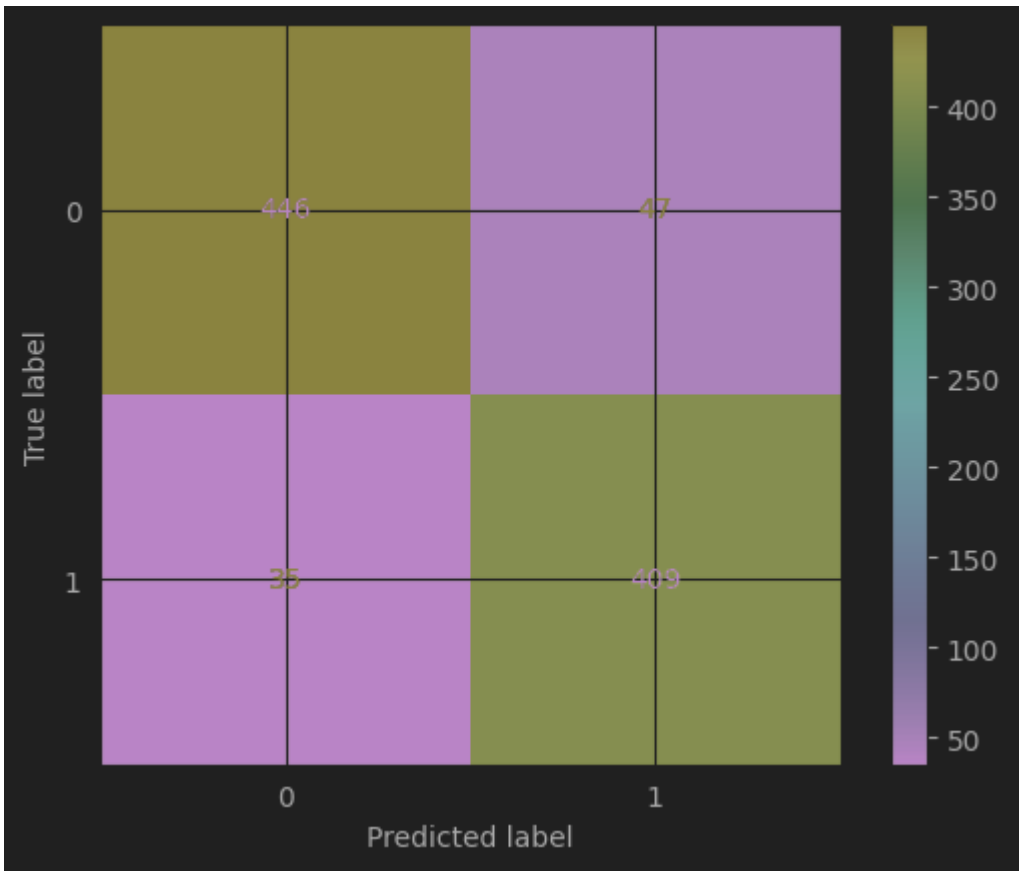
macro avg 0.91 0.91 0.91 937

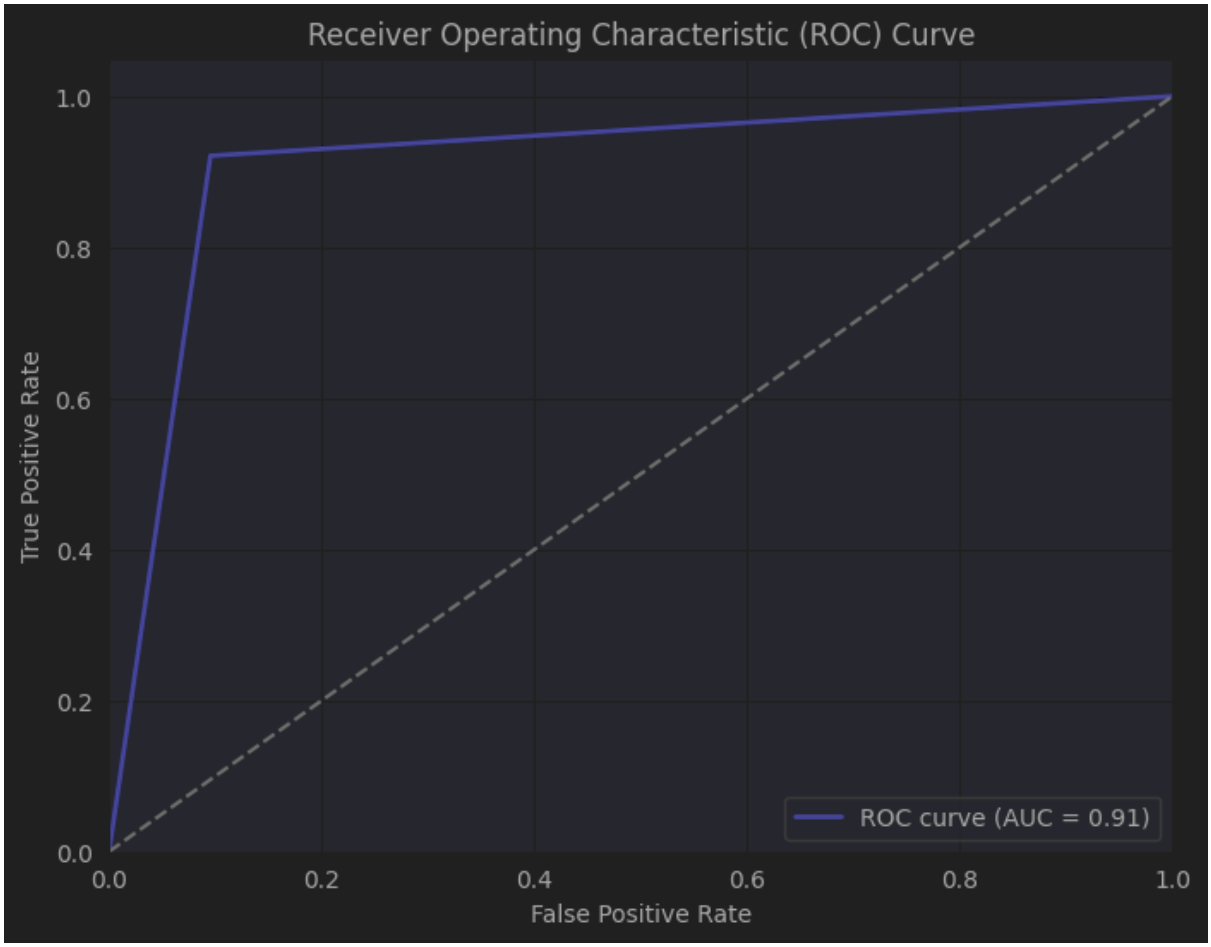
weighted avg 0.91 0.91 0.91 937

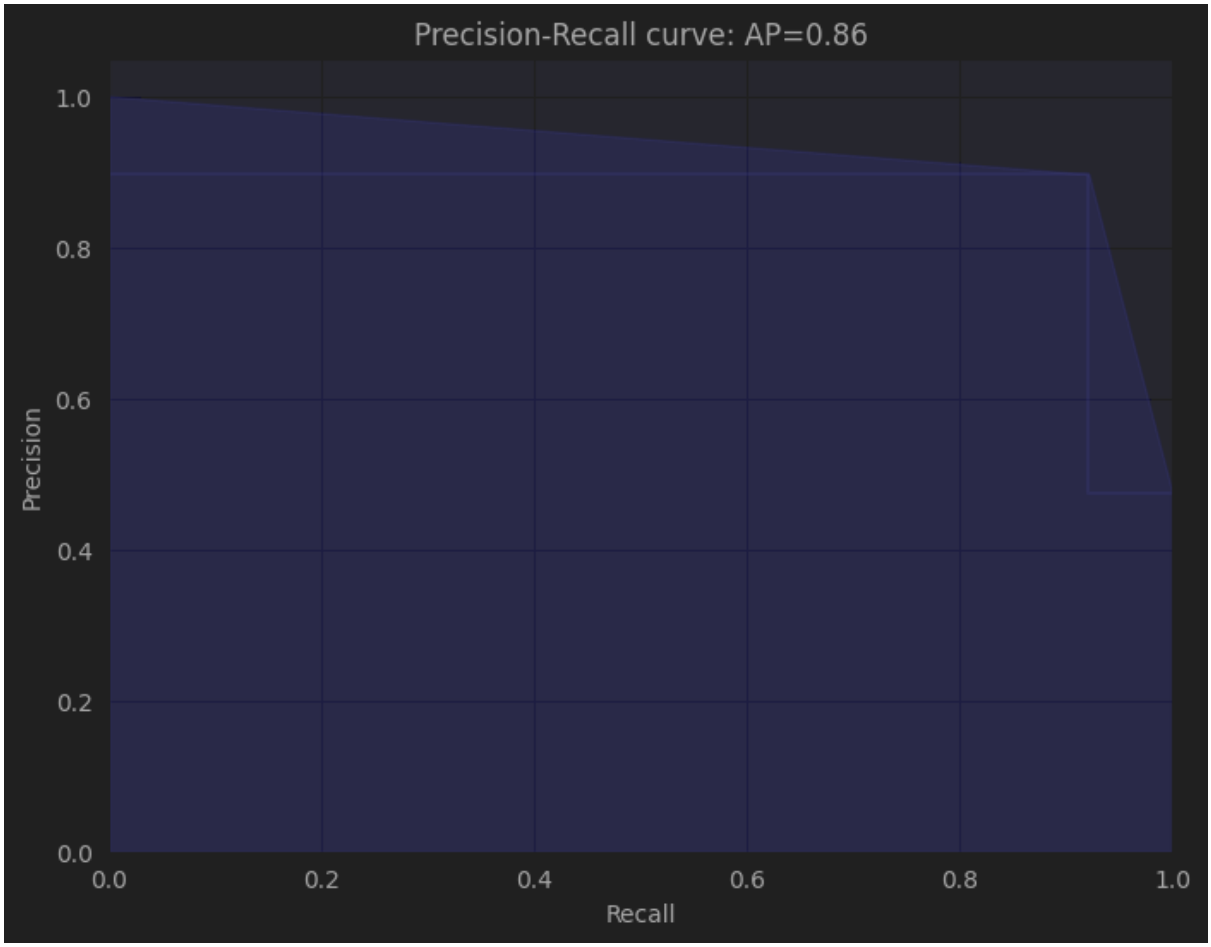
[[446 47]

[35 409]]

AUC-ROC Score: 0.9129182427863969




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.91)



Number of Folds: 4

Best Parameters: {'C': 100, 'class_weight': None, 'max_iter': 100, 'penalty': 'l2', 'solver': 'liblinear', 'tol': 0.01}

Model name: Logistic Regression

Cross-Validation Scores: [0.88510638 0.90598291 0.90598291 0.91025641 0.90638298 0.87606838

0.89316239 0.93162393 0.89787234 0.9017094 0.93589744 0.88034188]

Mean Cross-Validation Score: 0.9025322785961084

Skipping model performance, loss and accuracy plots

Accuracy: 0.9124866595517609

Precision: 0.8969298245614035

Recall: 0.9211711711711712

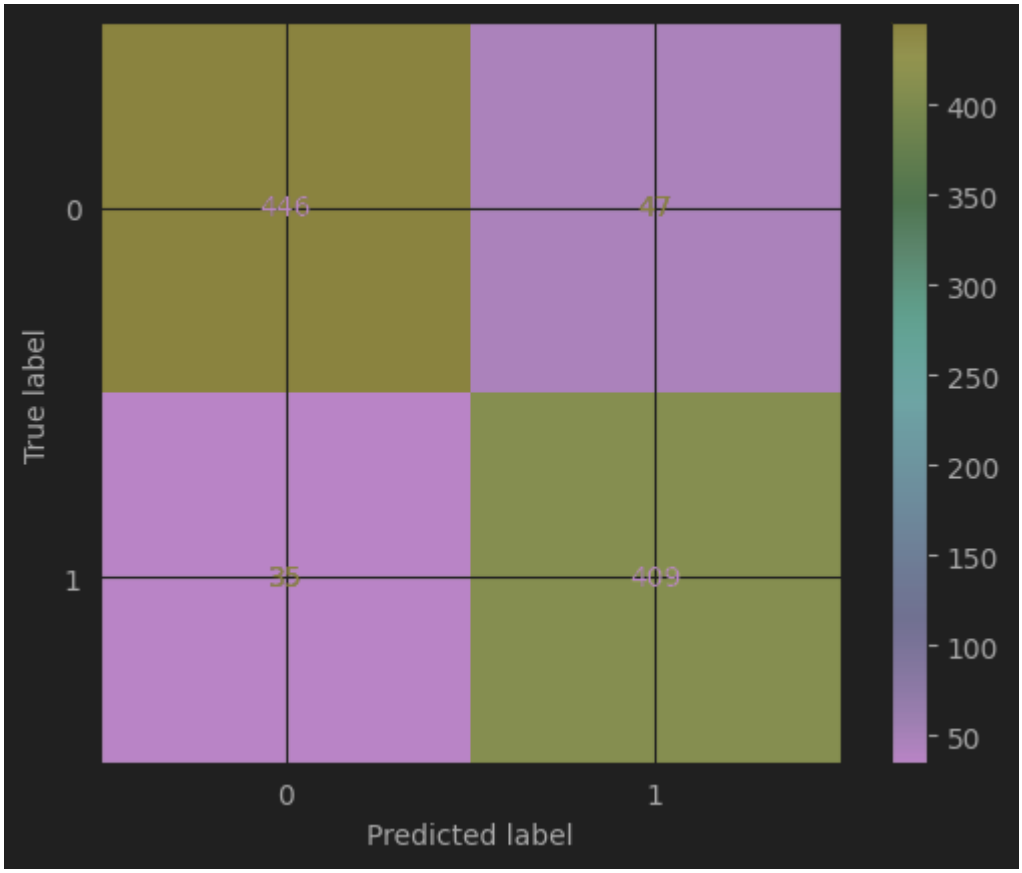
F1-score: 0.9088888888888889

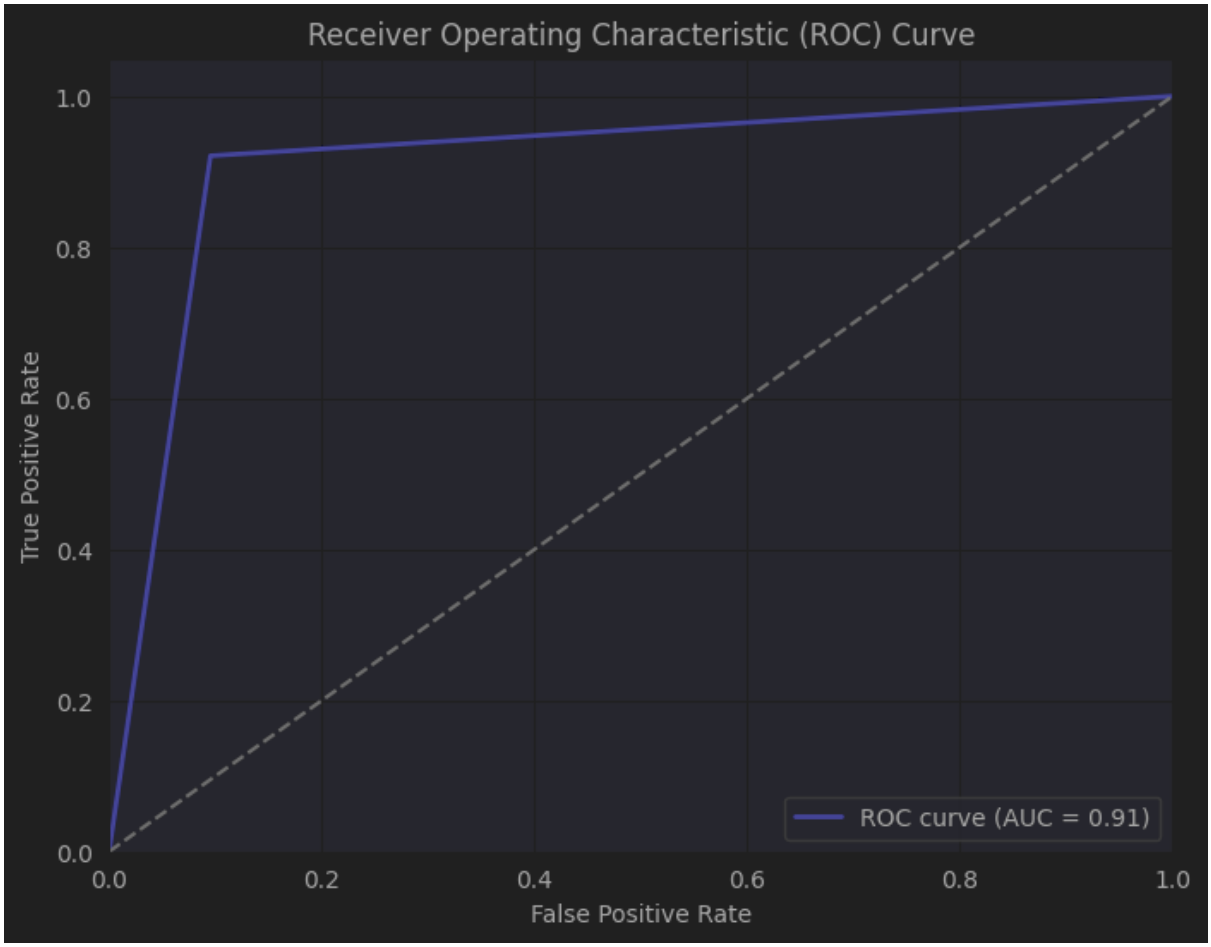
	precision	*recall*	*f1-score*	*support*
0	*0.93*	*0.90*	*0.92*	*493*

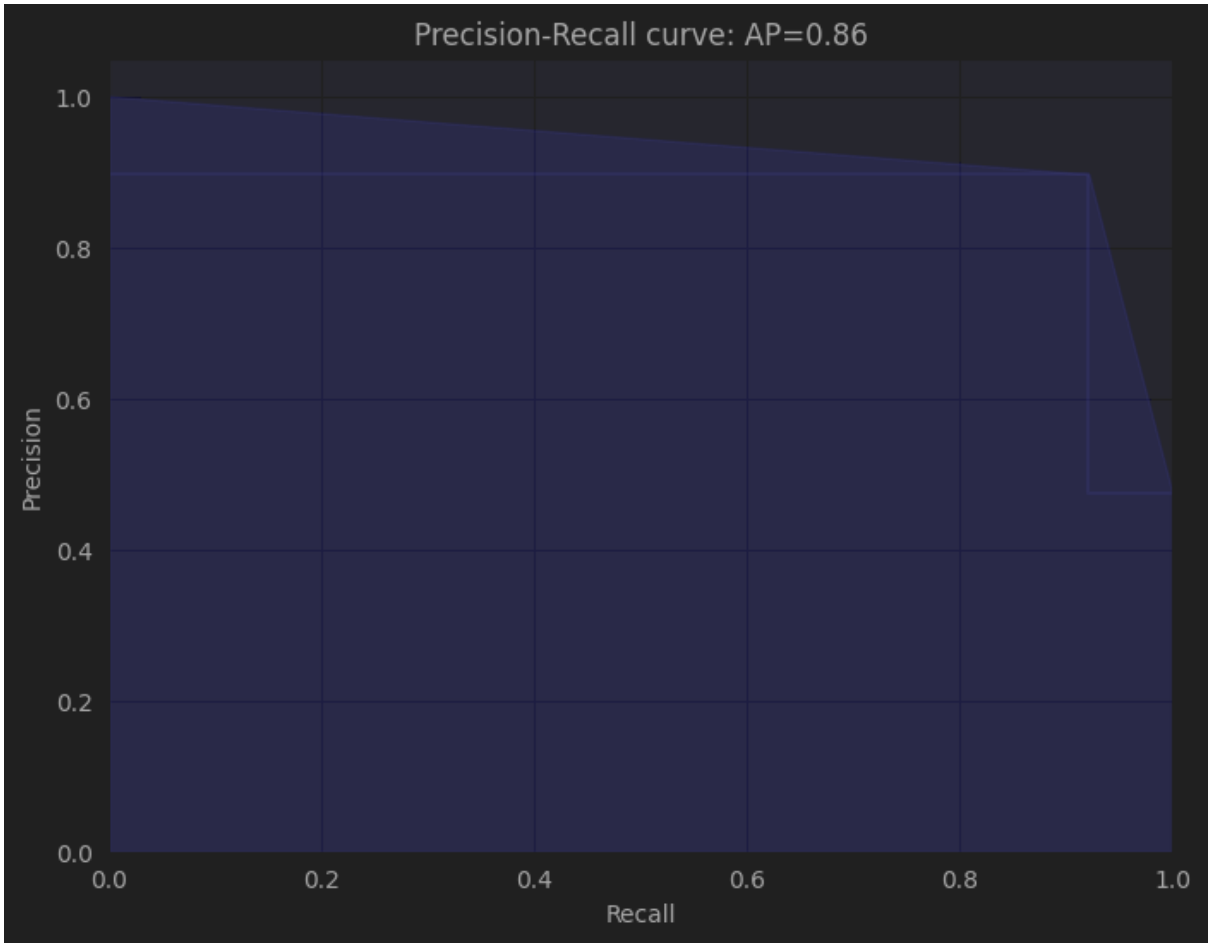
1 0.90 0.92 0.91 444

accuracy 0.91 937
macro avg 0.91 0.91 0.91 937
weighted avg 0.91 0.91 0.91 937

[[446 47]
[35 409]]
AUC-ROC Score: 0.9129182427863969




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.91)



Number of Folds: 5

Best Parameters: {'C': 10, 'class_weight': None, 'max_iter': 100, 'penalty': 'l2', 'solver': 'liblinear', 'tol': 0.01}

Model name: Logistic Regression

Cross-Validation Scores: [0.86702128 0.93085106 0.90374332 0.87700535 0.9144385 0.91489362

0.88829787 0.84491979 0.93582888 0.93048128 0.89893617 0.89893617

0.93048128 0.92513369 0.87700535]

Mean Cross-Validation Score: 0.9025315735578563

Skipping model performance, loss and accuracy plots

Accuracy: 0.9114194236926361

Precision: 0.8949671772428884

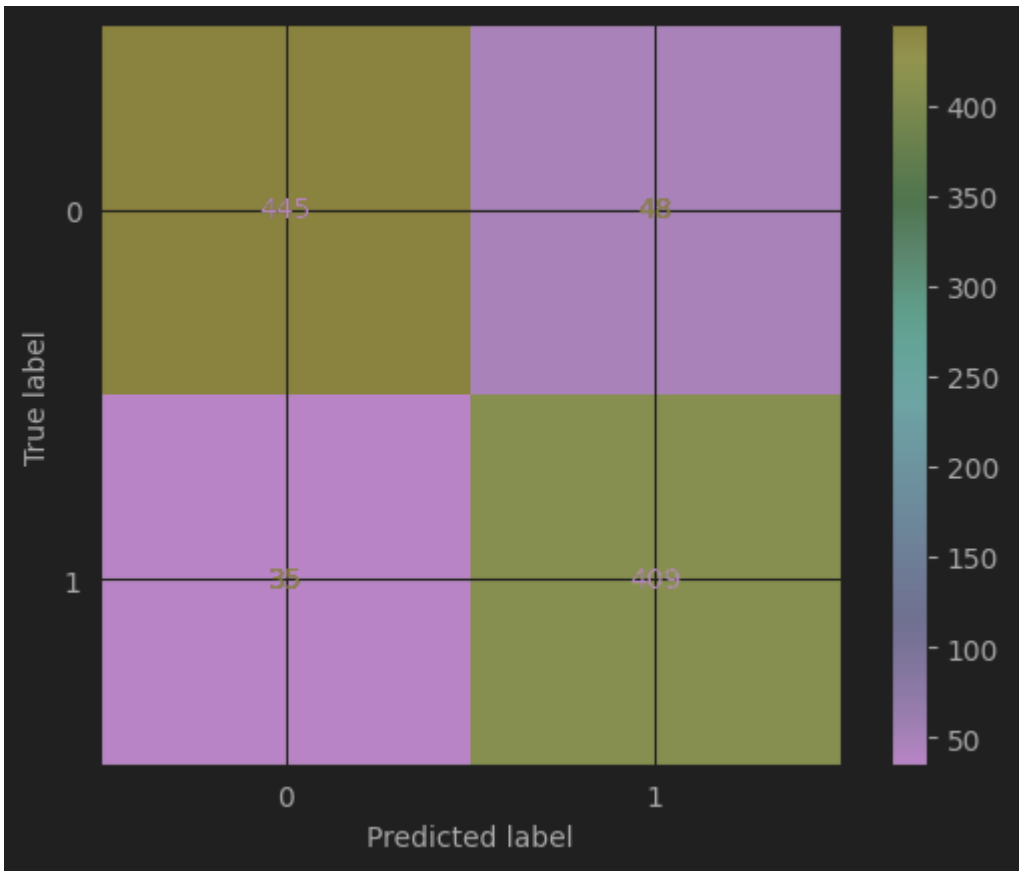
Recall: 0.9211711711711712

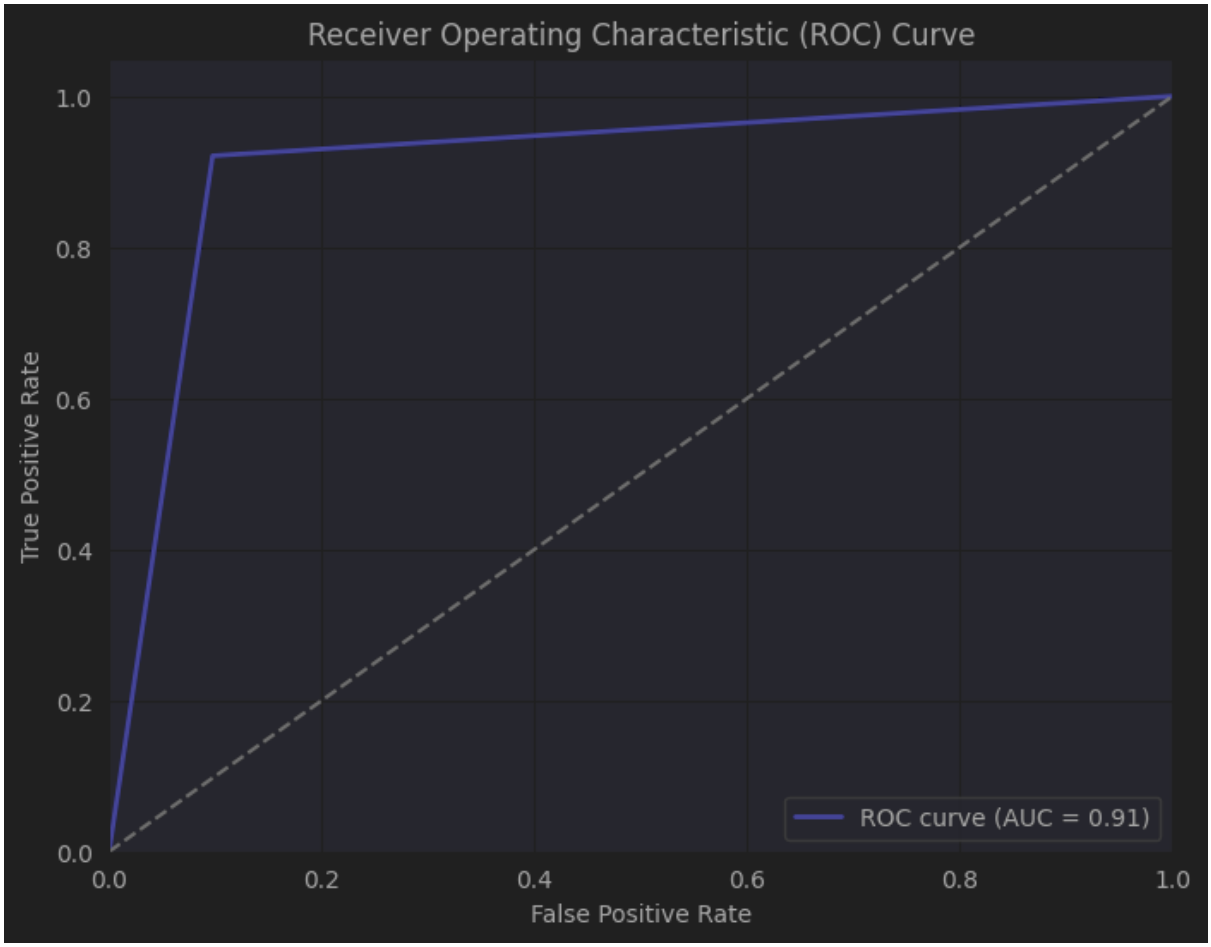
F1-score: 0.9078801331853497

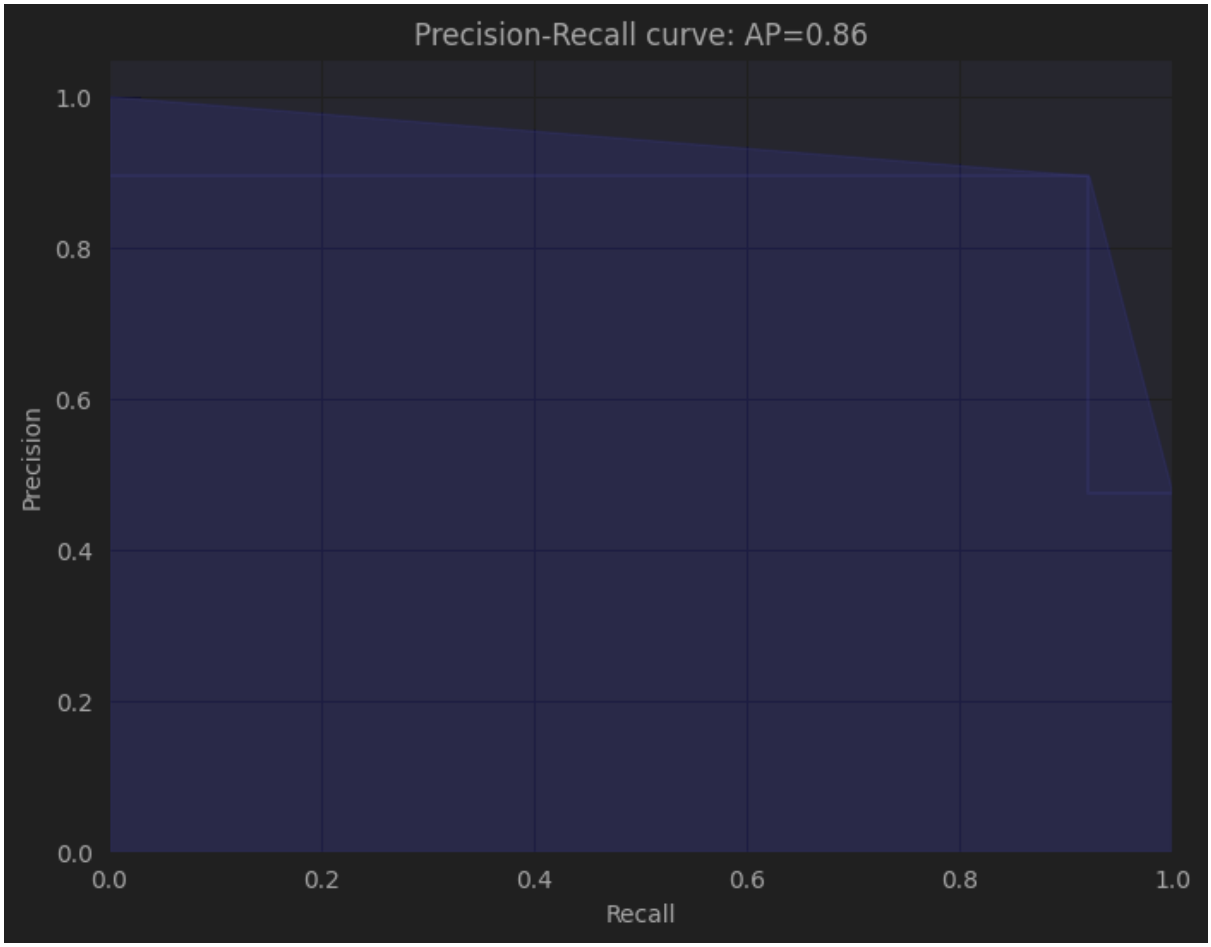
	precision	*recall*	*f1-score*	*support*

0	*0.93*	*0.90*	*0.91*	*493*
1	*0.89*	*0.92*	*0.91*	*444*
accuracy			*0.91*	*937*
macro avg	*0.91*	*0.91*	*0.91*	*937*
weighted avg	*0.91*	*0.91*	*0.91*	*937*

[[445 48]
[35 409]]
AUC-ROC Score: 0.9119040440034355




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.91)



Number of Folds: 10

Best Parameters: {'C': 1, 'class_weight': None, 'max_iter': 100, 'penalty': 'l1', 'solver': 'liblinear', 'tol': 0.0001}

Model name: Logistic Regression

Cross-Validation Scores: [0.84042553 0.89361702 0.96808511 0.92553191 0.89361702 0.92553191

0.90425532 0.87096774 0.89247312 0.94623656 0.89361702 0.93617021

0.92553191 0.86170213 0.88297872 0.81914894 0.93617021 0.94623656

0.95698925 0.91397849 0.86170213 0.91489362 0.88297872 0.92553191

0.92553191 0.95744681 0.93617021 0.90322581 0.86021505 0.90322581]

Mean Cross-Validation Score: 0.9068062228323038

Skipping model performance, loss and accuracy plots

Accuracy: 0.9114194236926361

Precision: 0.8949671772428884

Recall: 0.9211711711711712

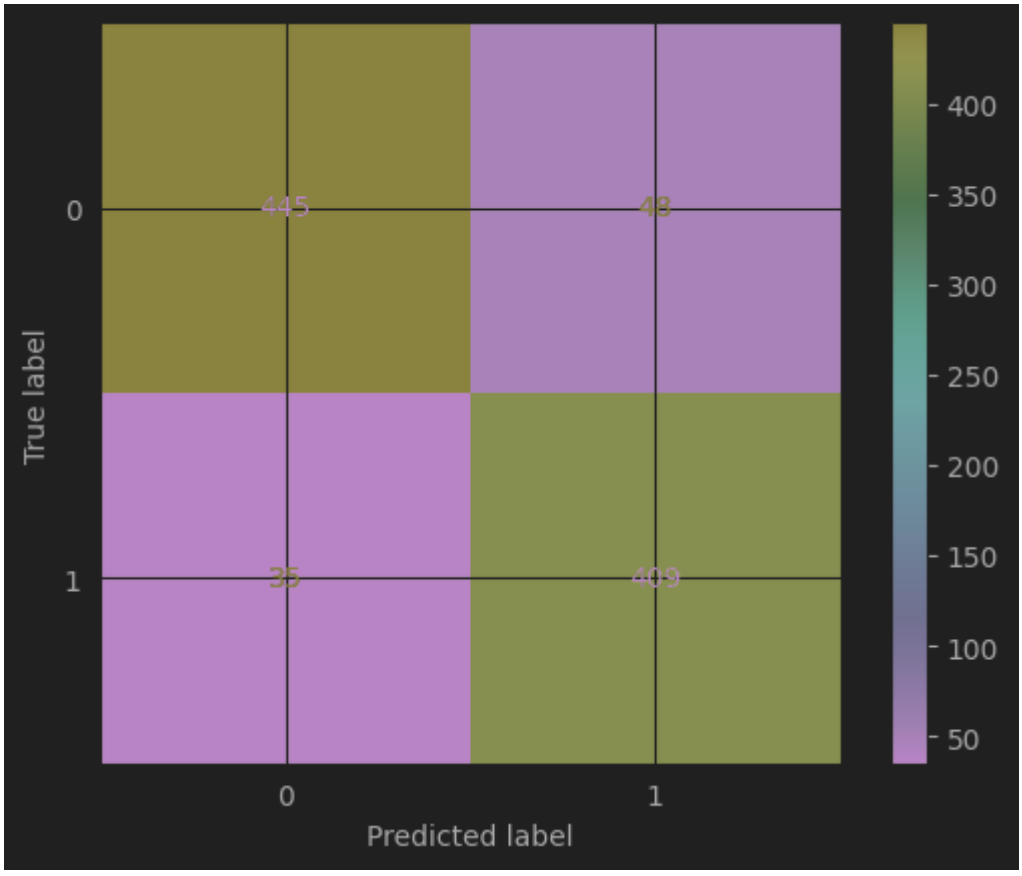
F1-score: 0.9078801331853497

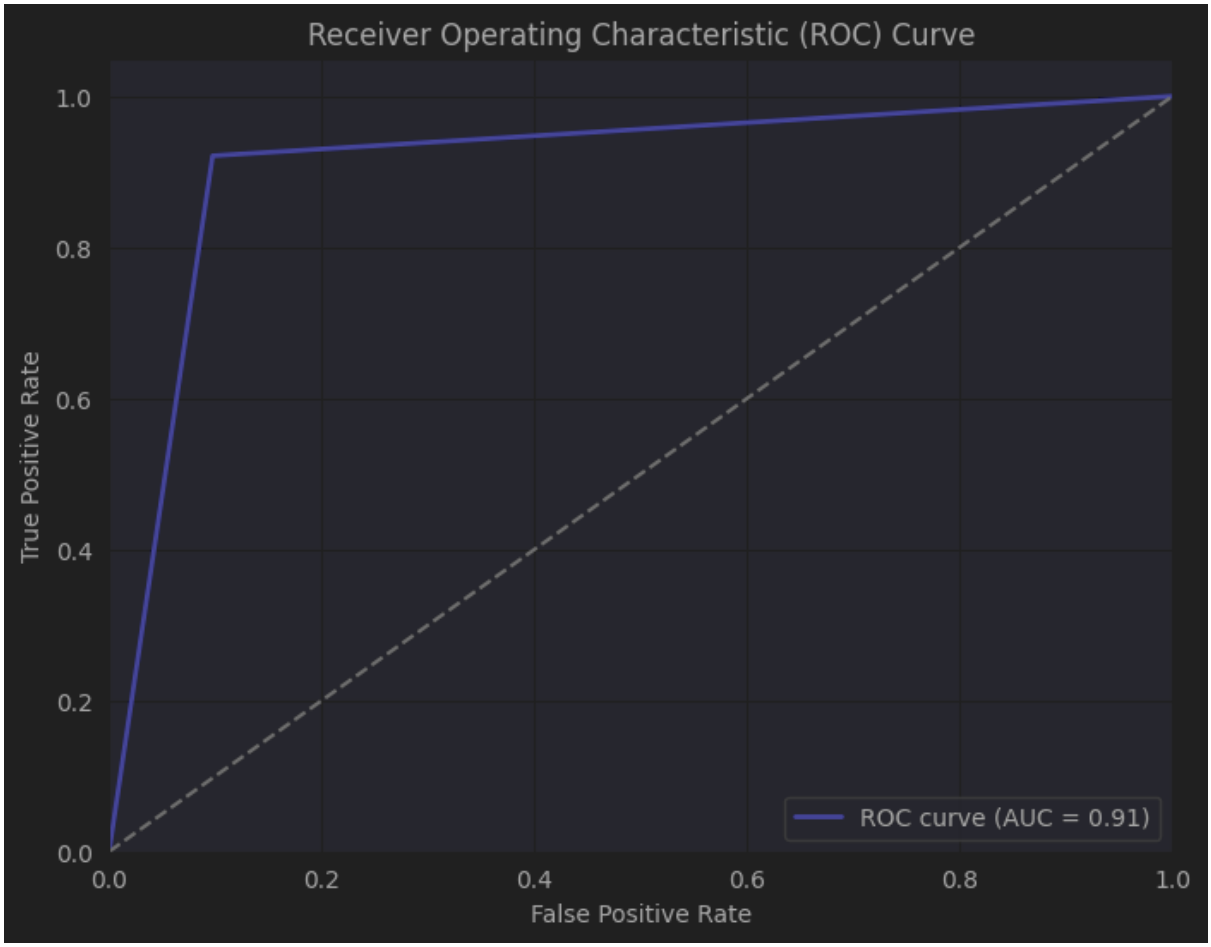
	precision	*recall*	*f1-score*	*support*
0	*0.93*	*0.90*	*0.91*	*493*
1	*0.89*	*0.92*	*0.91*	*444*
accuracy			*0.91*	*937*
macro avg	*0.91*	*0.91*	*0.91*	*937*
weighted avg	*0.91*	*0.91*	*0.91*	*937*

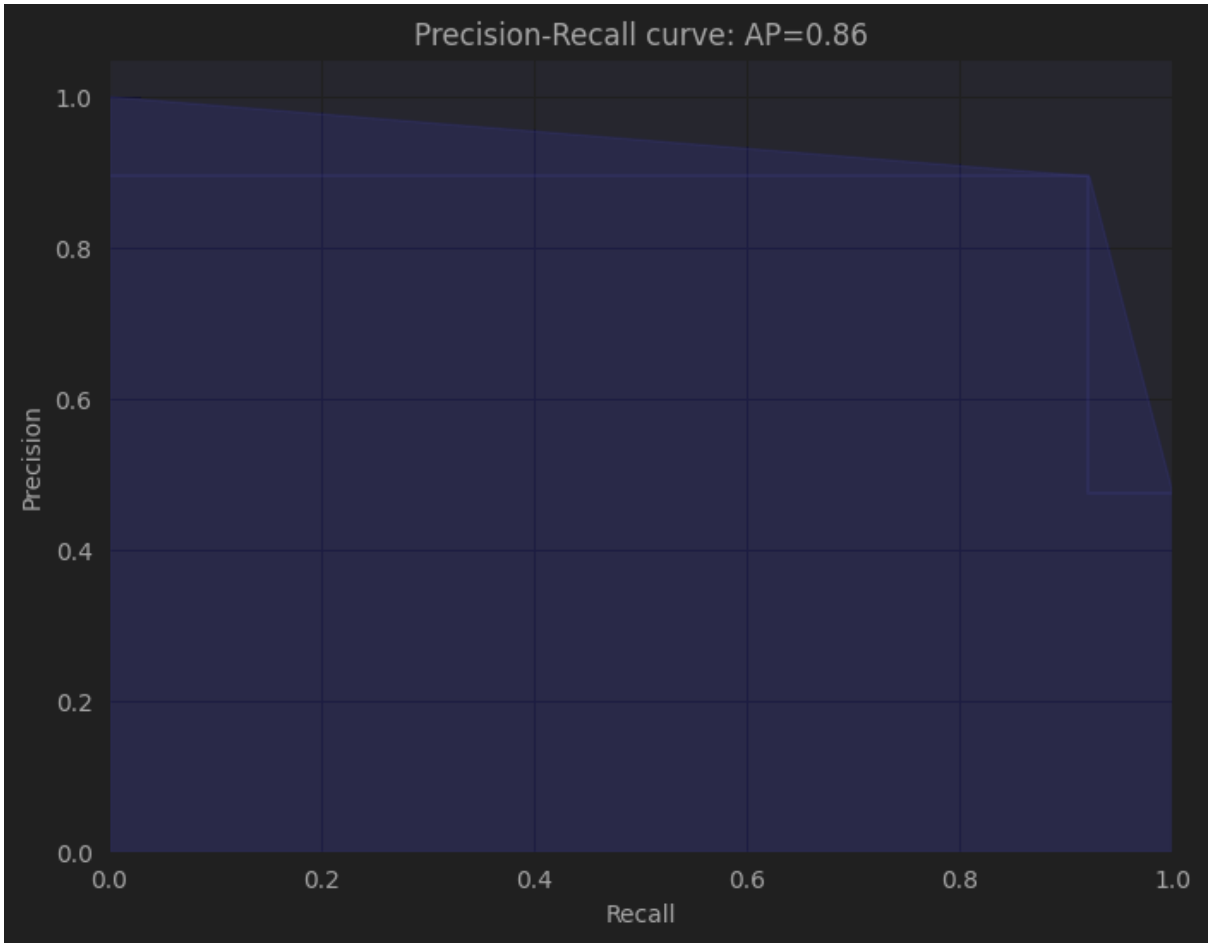
[[445 48]

[35 409]]

AUC-ROC Score: 0.9119040440034355




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.91)


Precision-Recall curve: AP=0.86
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

4.6.3 RANDOM FOREST (RF)

```
# Define hyperparameters grid for tuning
rf_param_grid = {
    'n_estimators': [100, 200, 300],  # Number of trees in the forest
    'max_depth': [None, 10, 20, 30],    # Maximum depth of the tree
    'min_samples_split': [2, 5, 10],    # Minimum number of samples required to split an internal node
    'min_samples_leaf': [1, 2, 4]      # Minimum number of samples required to be at a leaf node
}

# Define different numbers of folds for cross-validation
num_folds = [3, 4, 5, 10]
# num_folds = [10]

for folds in num_folds:
    # Create RepeatedStratifiedKFold with the specified number of folds
    cv = RepeatedStratifiedKFold(n_splits=folds, n_repeats=3, random_state=42)
    # cv = folds

    # Define the logistic regression model
    RF = RandomForestClassifier(random_state=42)

    # Perform grid search with repeated cross-validation
    grid_search = GridSearchCV(estimator=RF, param_grid=rf_param_grid, cv=cv, scoring='accuracy', n_jobs=-1)
    grid_search.fit(x_tr_resample, y_tr_resample)

    # Get the best model from grid search
    best_RF = grid_search.best_estimator_

    # Print the best parameters found
    print(f"\nNumber of Folds: {folds}")
```

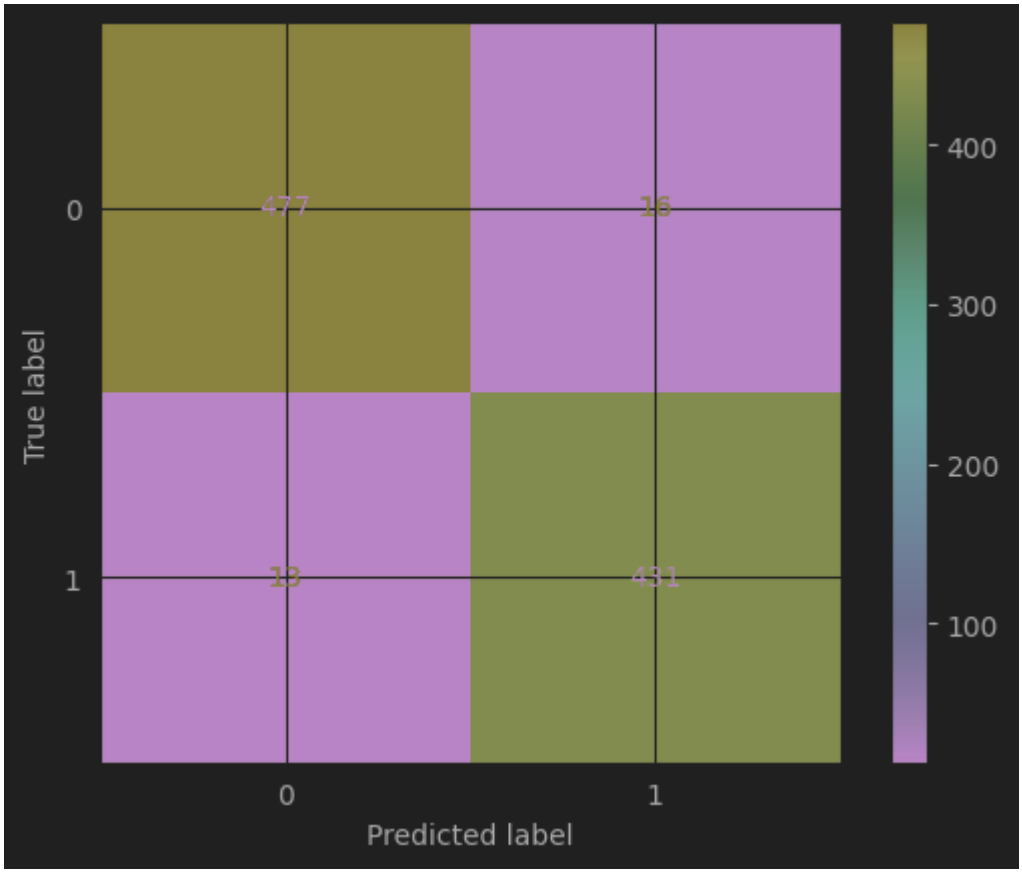
```
print("Best Parameters:", grid_search.best_params_)

# Use the best parameters to build the model
best_params = grid_search.best_params_
best_rf_model = RandomForestClassifier(random_state=42, **best_params)

# Fit the model on the training data with the best parameters
best_rf_model.fit(x_tr_resample, y_tr_resample)

# Evaluate the best model
evaluate_model("Random Forest", best_RF, norm_test_f, y_test, cv)
```

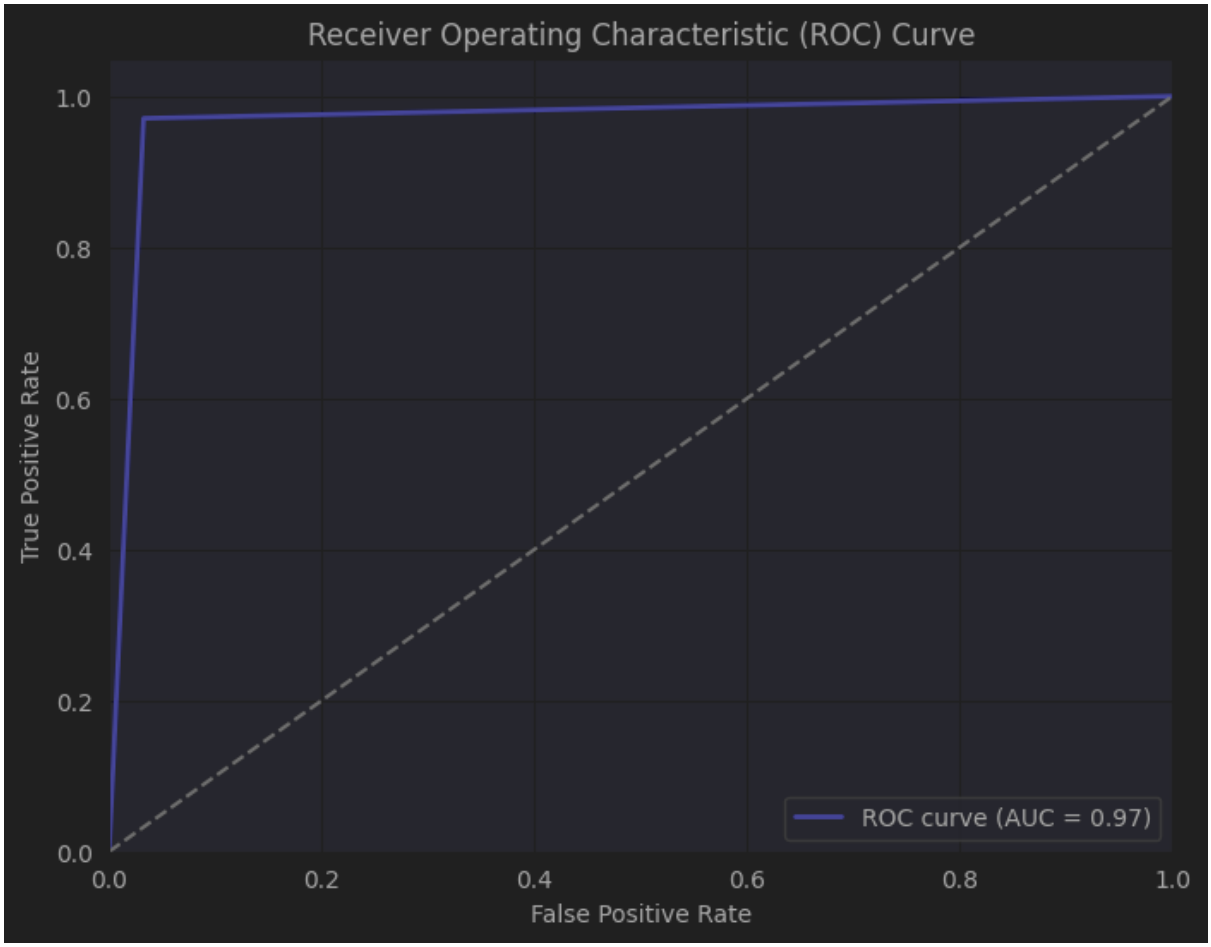
Output:

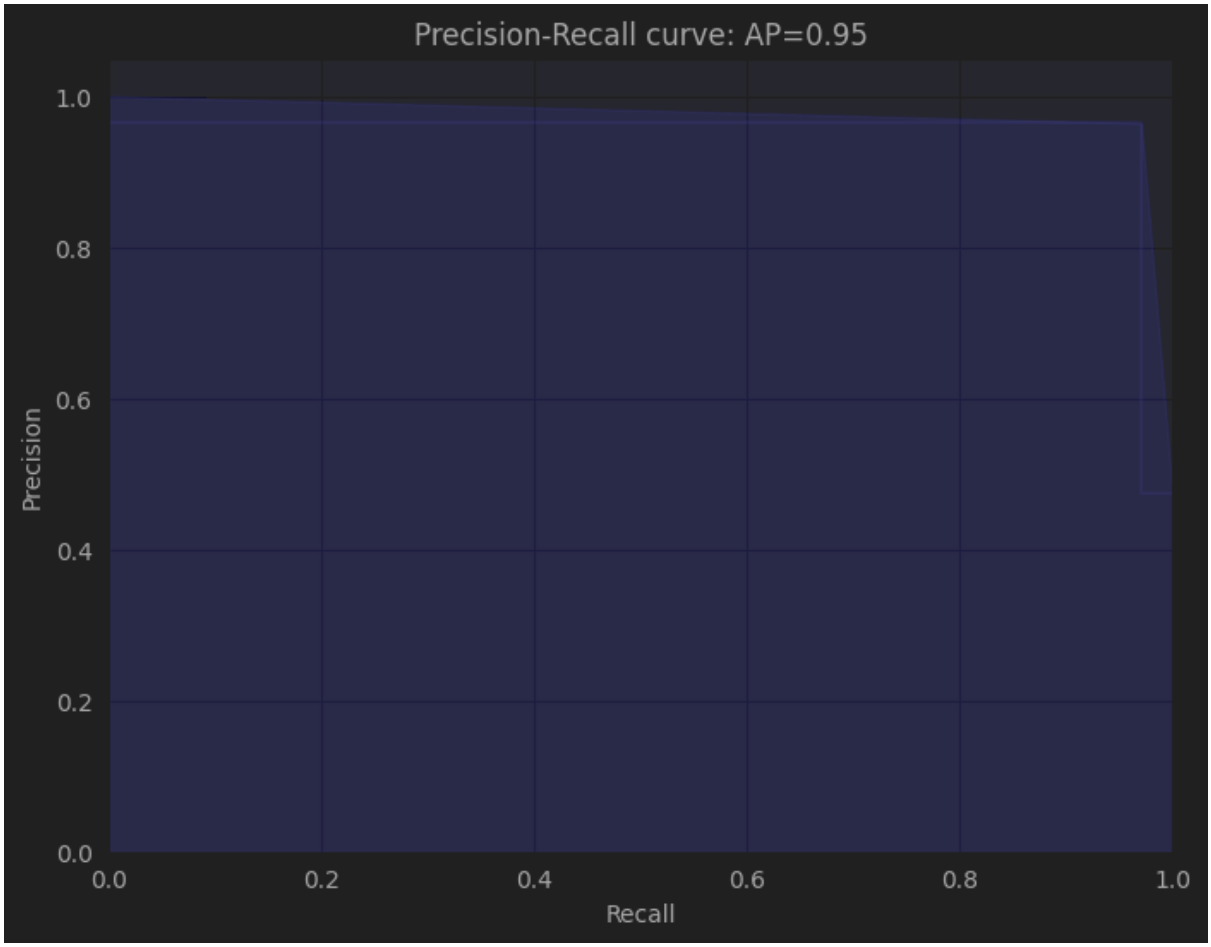
```
Number of Folds: 3
Best Parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}
Model name:  Random Forest
Cross-Validation Scores: [0.95207668 0.93910256 0.93589744 0.93929712 0.94551282 0.96474359
 0.94249201 0.95833333 0.92628205]
Mean Cross-Validation Score: 0.9448597343964755
Skipping model performance, loss and accuracy plots
Accuracy: 0.9690501600853789
Precision: 0.9642058165548099
Recall: 0.9707207207207207
F1-score: 0.9674523007856342
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       493
           1       0.96      0.97      0.97       444

    accuracy                           0.97       937
```

macro avg 0.97 0.97 0.97 937
weighted avg 0.97 0.97 0.97 937

[[477 16]
[13 431]]
AUC-ROC Score: 0.969133179832977




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.97)



Number of Folds: 4

Best Parameters: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}

Model name: Random Forest

Cross-Validation Scores: [0.94468085 0.95726496 0.96581197 0.94017094 0.94042553 0.93162393

0.94871795 0.97008547 0.95319149 0.96581197 0.94871795 0.93589744]

Mean Cross-Validation Score: 0.9502000363702492

Skipping model performance, loss and accuracy plots

Accuracy: 0.9690501600853789

Precision: 0.9642058165548099

Recall: 0.9707207207207207

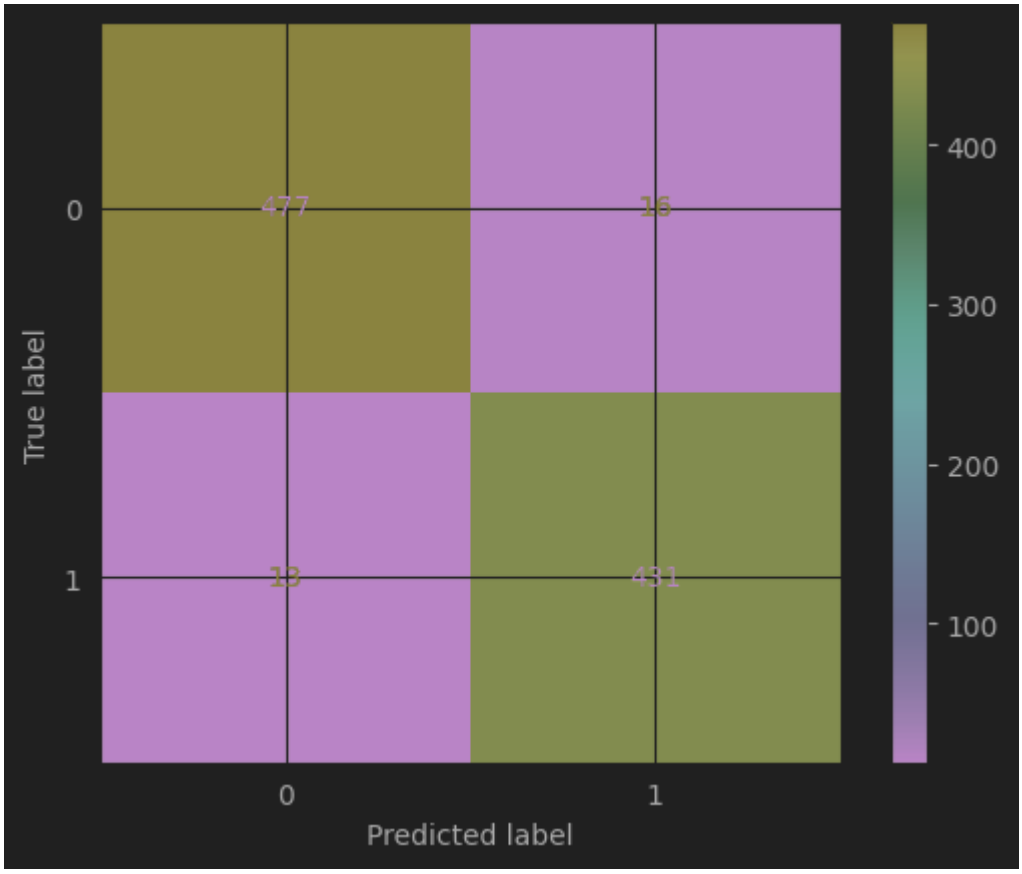
F1-score: 0.9674523007856342

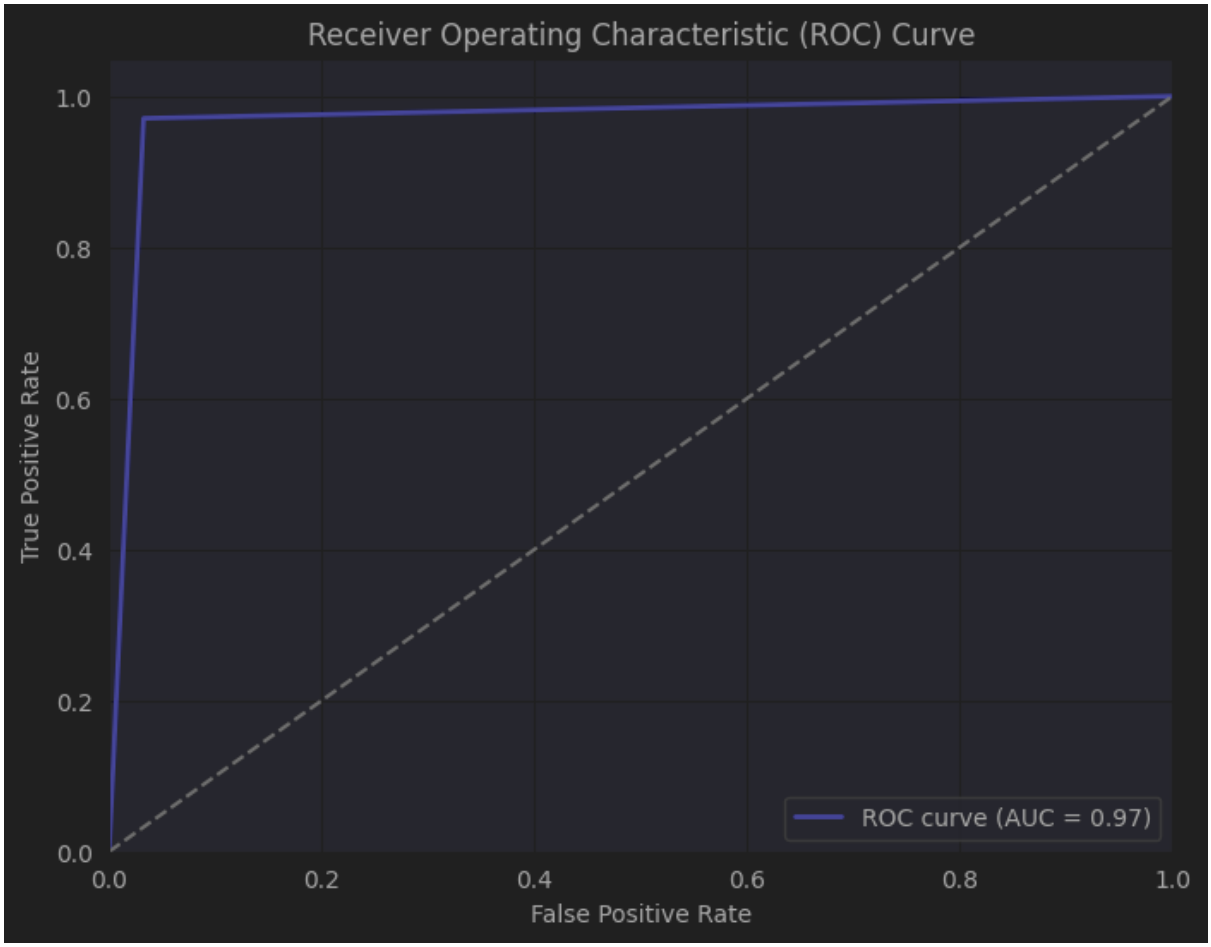
	precision	*recall*	*f1-score*	*support*
0	*0.97*	*0.97*	*0.97*	*493*

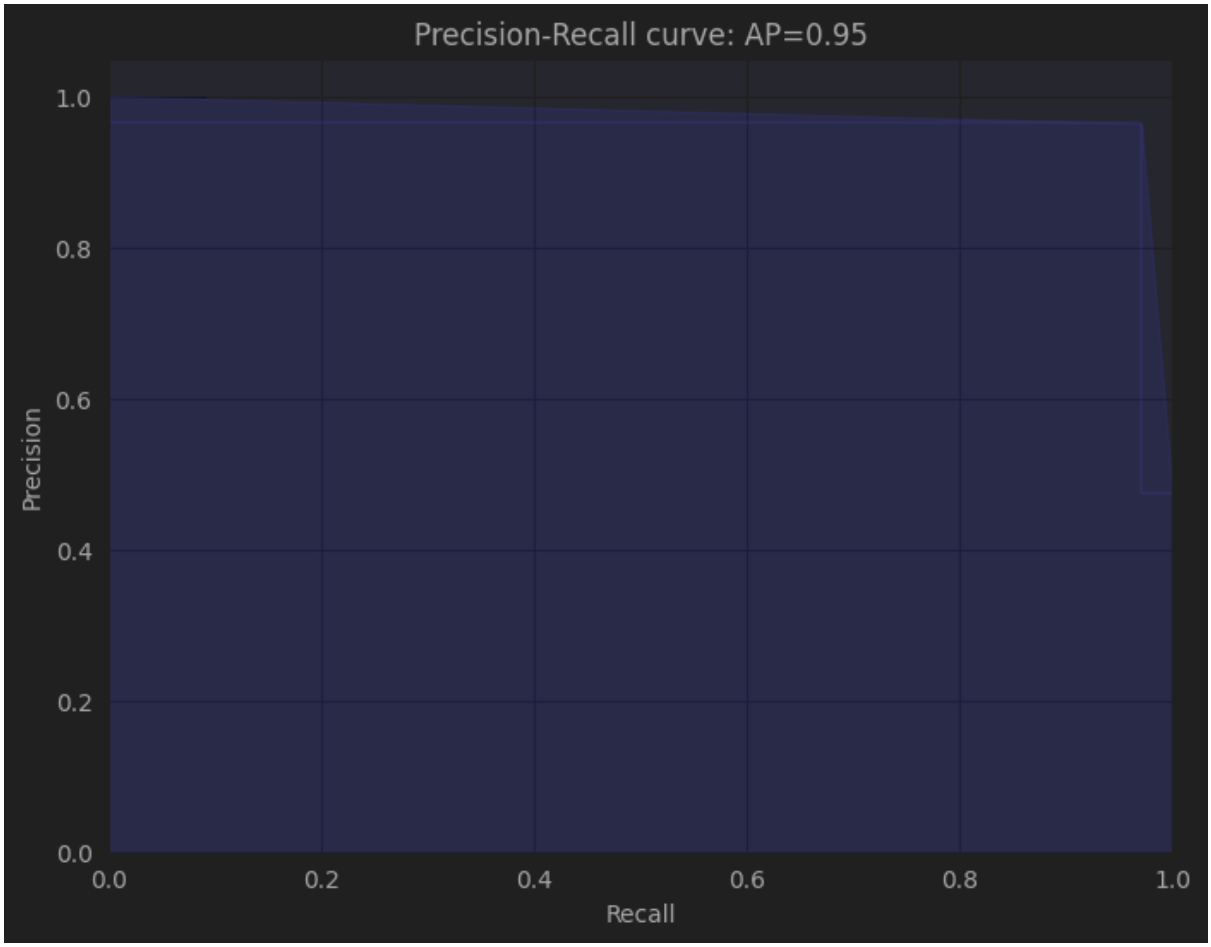
1 0.96 0.97 0.97 444

accuracy 0.97 937
macro avg 0.97 0.97 0.97 937
weighted avg 0.97 0.97 0.97 937

[[477 16]
[13 431]]
AUC-ROC Score: 0.969133179832977




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.97)



Number of Folds: 5

Best Parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}

Model name: Random Forest

Cross-Validation Scores: [0.95744681 0.95744681 0.94117647 0.96256684 0.92513369 0.93617021

0.94680851 0.93582888 0.97860963 0.96256684 0.95744681 0.95744681

0.96791444 0.92513369 0.93582888]

Mean Cross-Validation Score: 0.9498350210490387

Skipping model performance, loss and accuracy plots

Accuracy: 0.967982924226254

Precision: 0.96

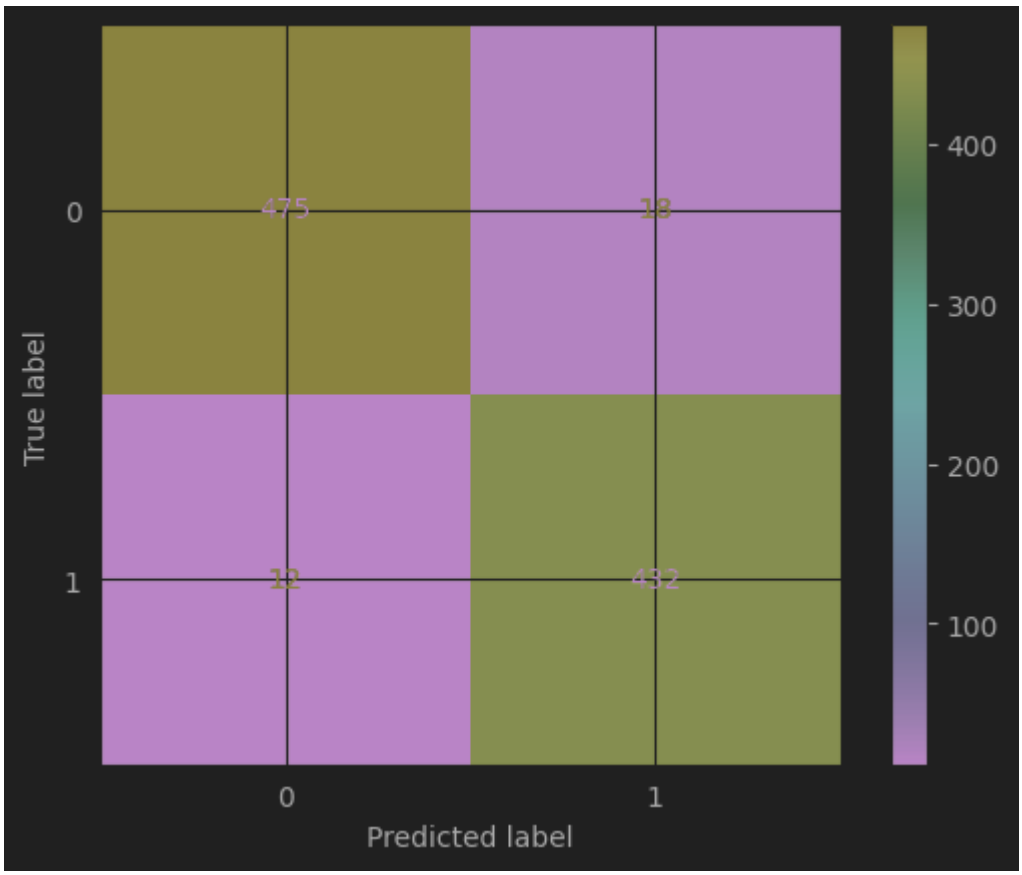
Recall: 0.972972972972973

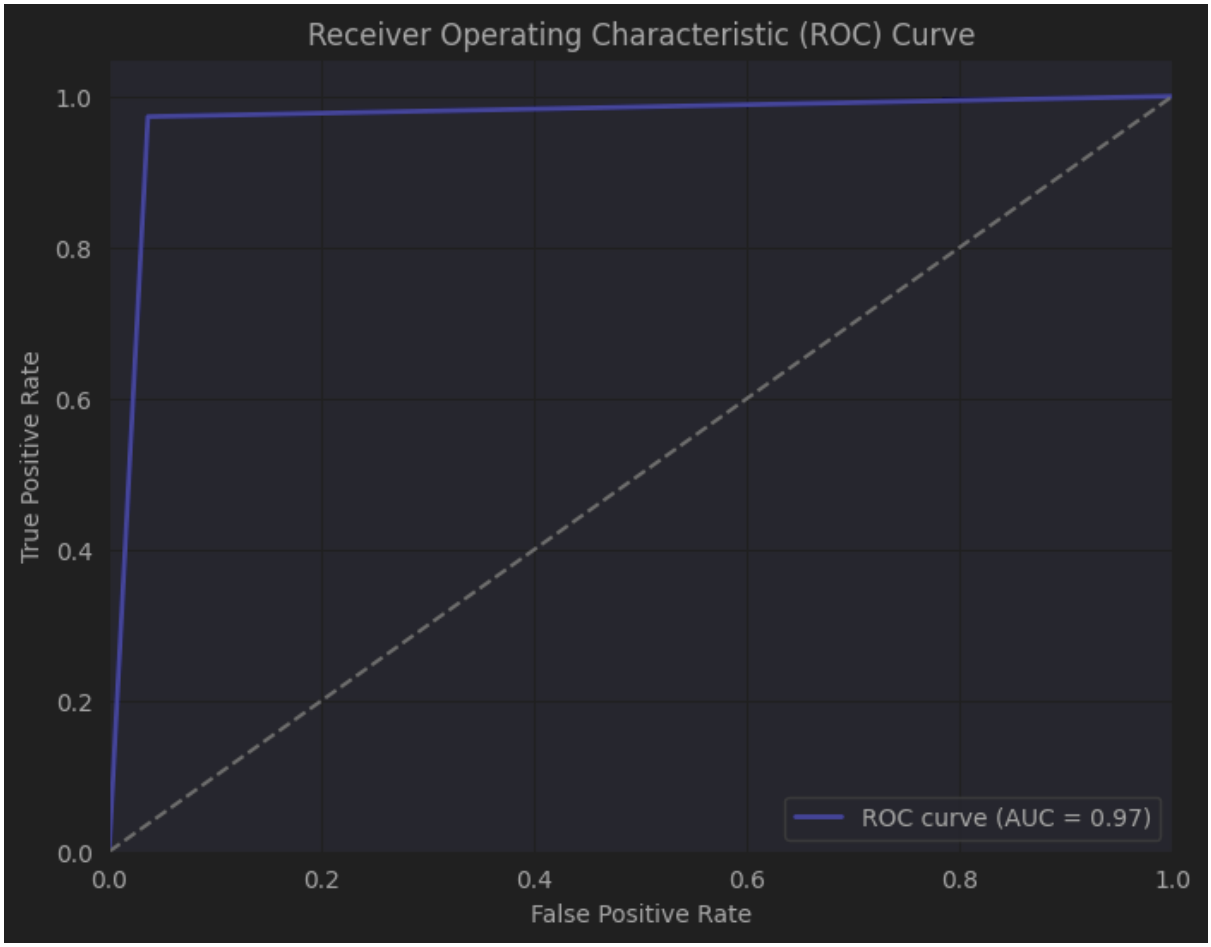
F1-score: 0.9664429530201343

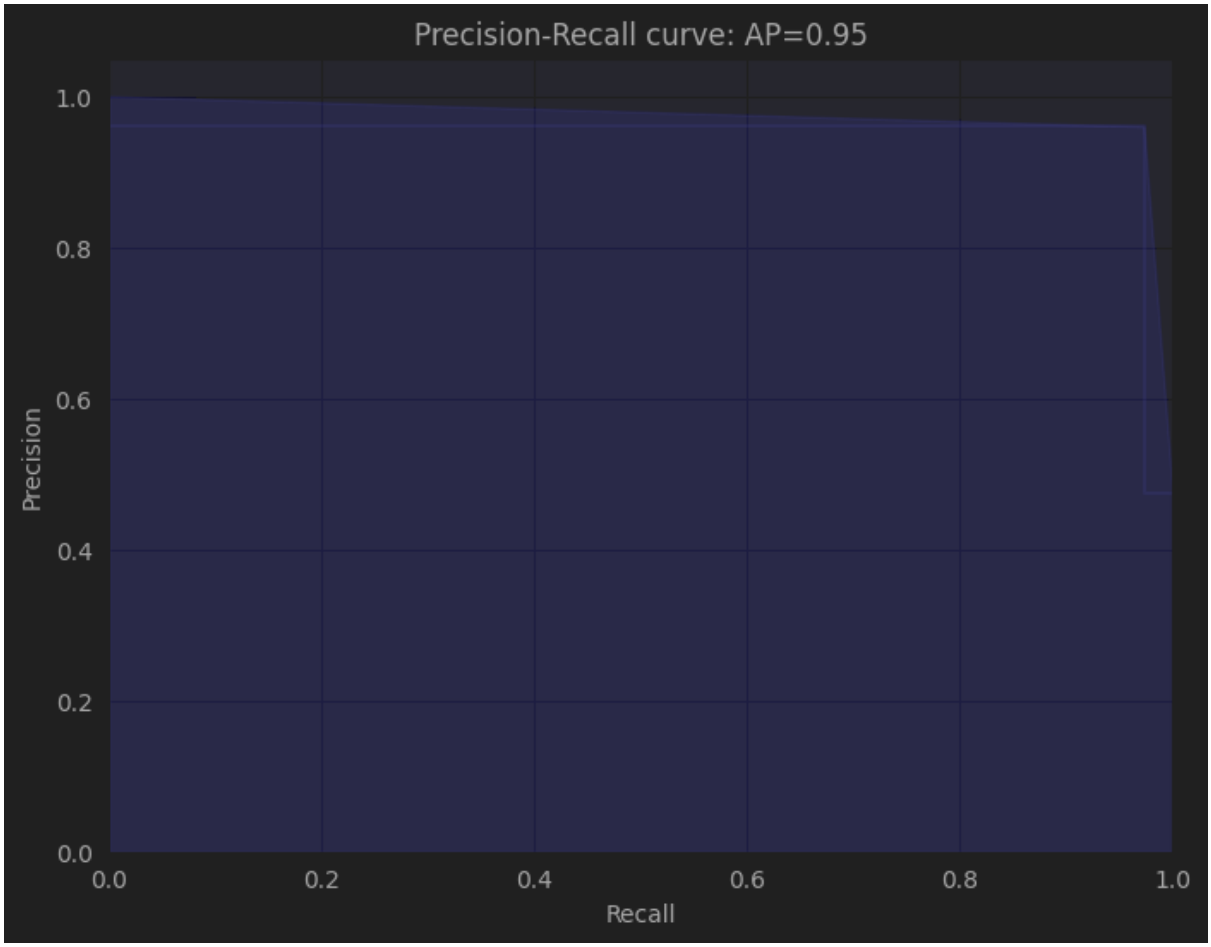
precision recall f1-score support

0	*0.98*	*0.96*	*0.97*	*493*
1	*0.96*	*0.97*	*0.97*	*444*
accuracy			*0.97*	*937*
macro avg	*0.97*	*0.97*	*0.97*	*937*
weighted avg	*0.97*	*0.97*	*0.97*	*937*

[[475 18]
[12 432]]
AUC-ROC Score: 0.9682309083931802




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.97)



Number of Folds: 10

Best Parameters: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}

Model name: Random Forest

Cross-Validation Scores: [0.92553191 0.96808511 0.95744681 0.94680851 0.93617021 0.95744681

0.9787234 0.93548387 0.92473118 0.94623656 0.92553191 0.94680851

0.94680851 0.93617021 0.93617021 0.93617021 0.9787234 0.96774194

0.96774194 0.97849462 0.93617021 0.96808511 0.96808511 0.96808511

0.96808511 0.95744681 0.93617021 0.94623656 0.92473118 0.94623656]

Mean Cross-Validation Score: 0.9505452604285822

Skipping model performance, loss and accuracy plots

Accuracy: 0.967982924226254

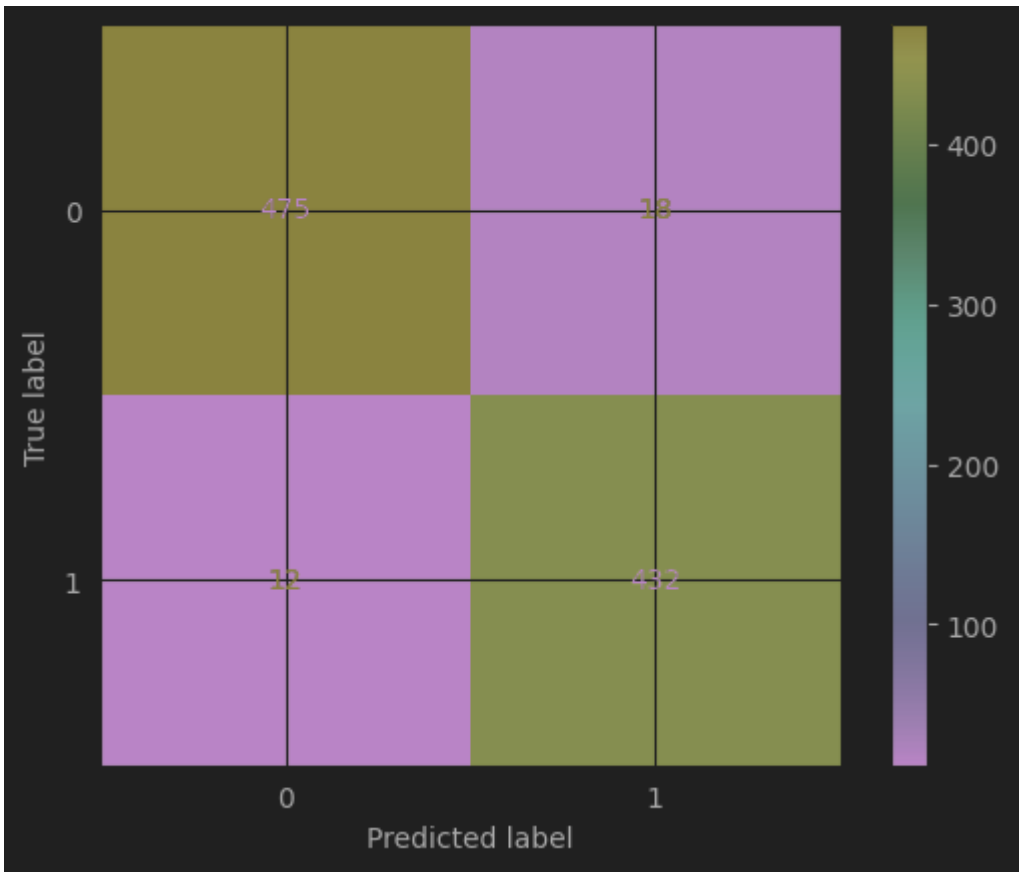
Precision: 0.96

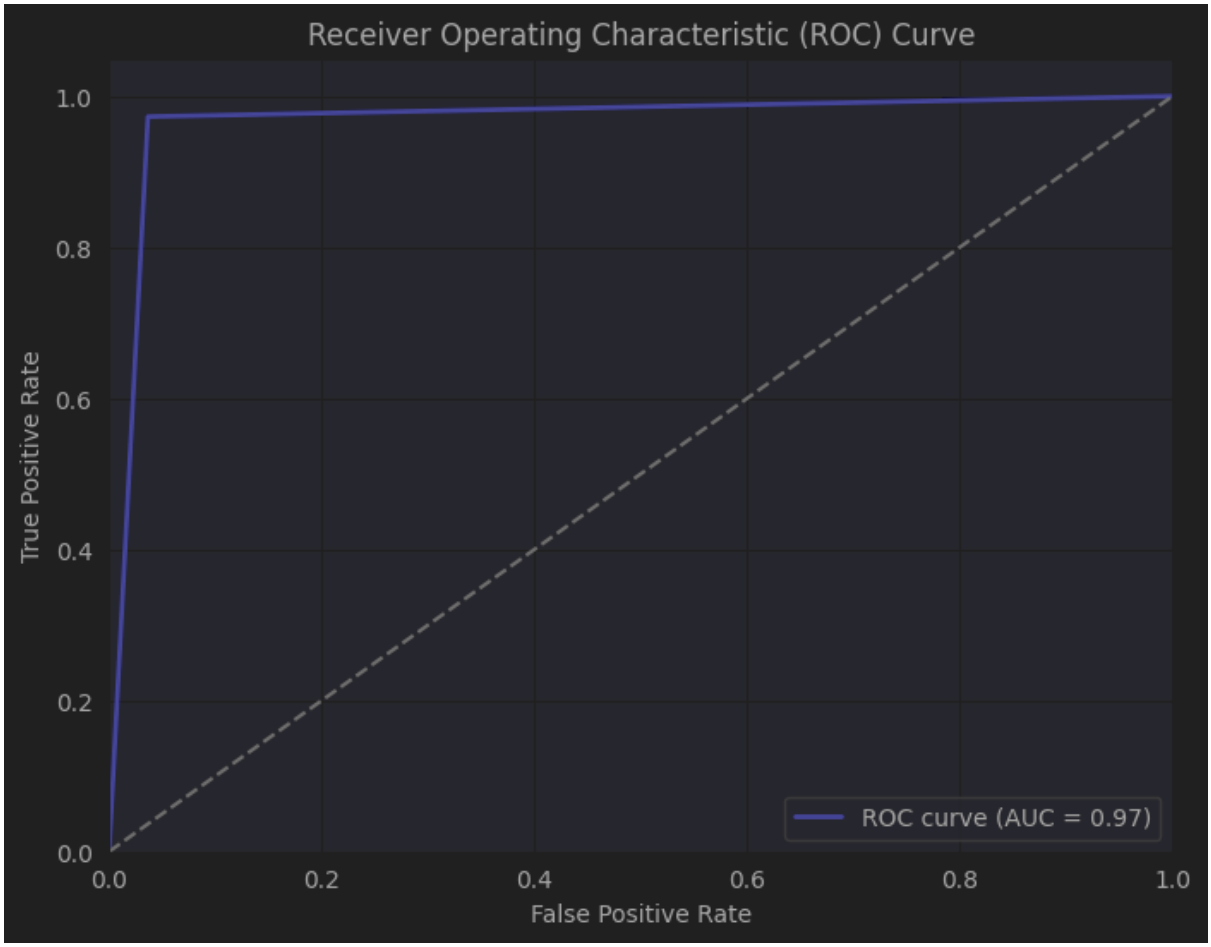
Recall: 0.972972972972973

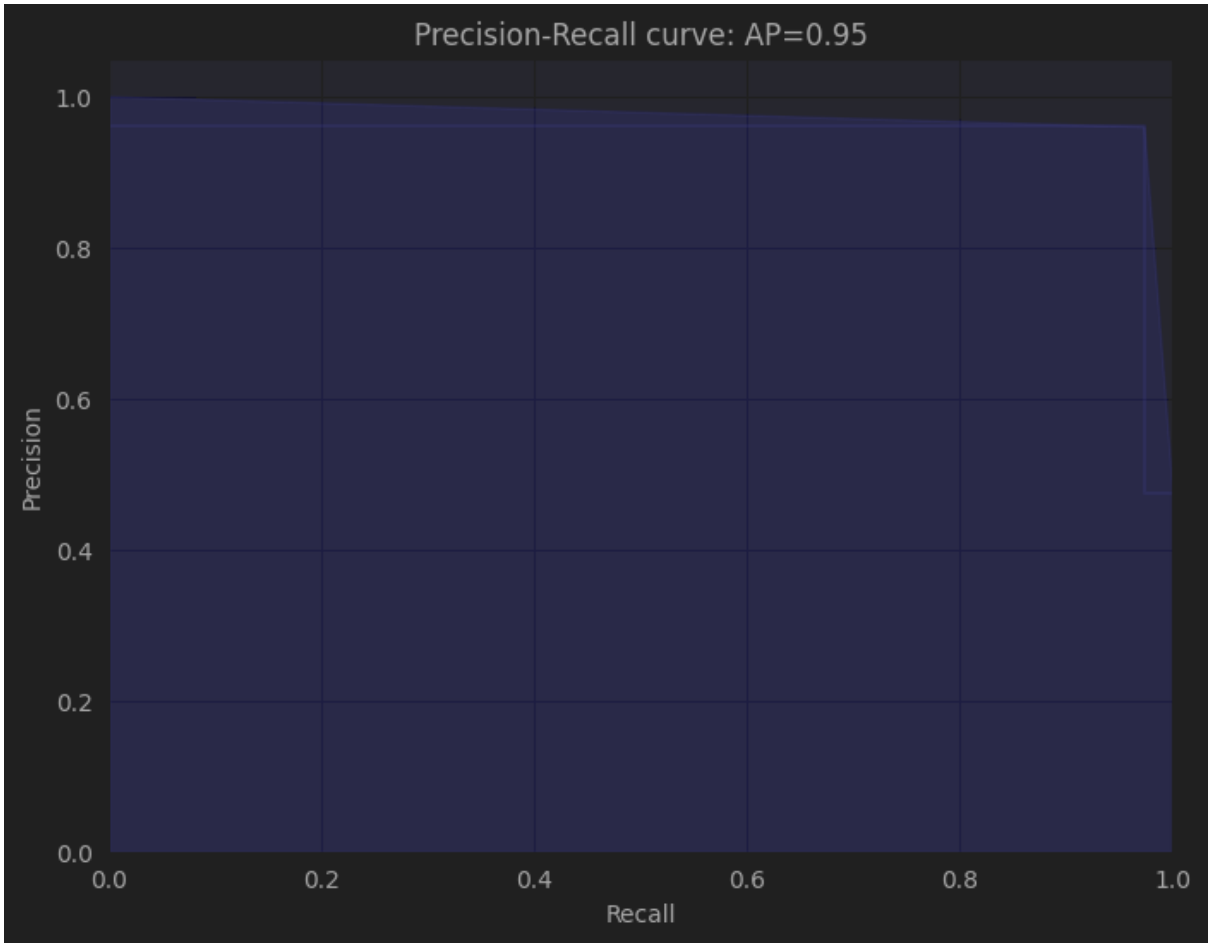
F1-score: 0.9664429530201343

	precision	recall	f1-score	support
0	0.98	0.96	0.97	493
1	0.96	0.97	0.97	444
accuracy			0.97	937
macro avg	0.97	0.97	0.97	937
weighted avg	0.97	0.97	0.97	937

[[475 18]
[12 432]]
AUC-ROC Score: 0.9682309083931802




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.97)


Precision-Recall curve: AP=0.95
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

4.6.4 SUPPORT VECTOR MACHINES (SVM)

```
from sklearn.svm import SVC

# Define hyperparameters grid for tuning
svm_param_grid = {
    'C': [0.1, 1, 10, 100], # Regularization parameter
    'gamma': ['scale', 'auto', 0.1, 1, 10, 100], # Kernel coefficient for 'rbf' kernel
    'kernel': ['linear', 'rbf'] # Kernel type
}

for folds in num_folds:
    # Create RepeatedStratifiedKFold with the specified number of folds
    cv = RepeatedStratifiedKFold(n_splits=folds, n_repeats=3, random_state=42)
    # cv = folds

    # Perform grid search with repeated cross-validation
    grid_search = GridSearchCV(estimator=SVC(random_state=42),
param_grid=svm_param_grid, cv=cv, scoring='accuracy',  n_jobs=5)
    grid_search.fit(x_tr_resample, y_tr_resample)

    # Get the best model from grid search
    best_svm = grid_search.best_estimator_

    # Print the best parameters found
    print(f"\nNumber of Folds: {folds}")
    print("Best Parameters:", grid_search.best_params_)

    # Use the best parameters to build the model
    best_params = grid_search.best_params_
    best_svm_model = SVC(random_state=42, **best_params)

    # Fit the model on the training data with the best parameters
    best_svm_model.fit(x_tr_resample, y_tr_resample)
```

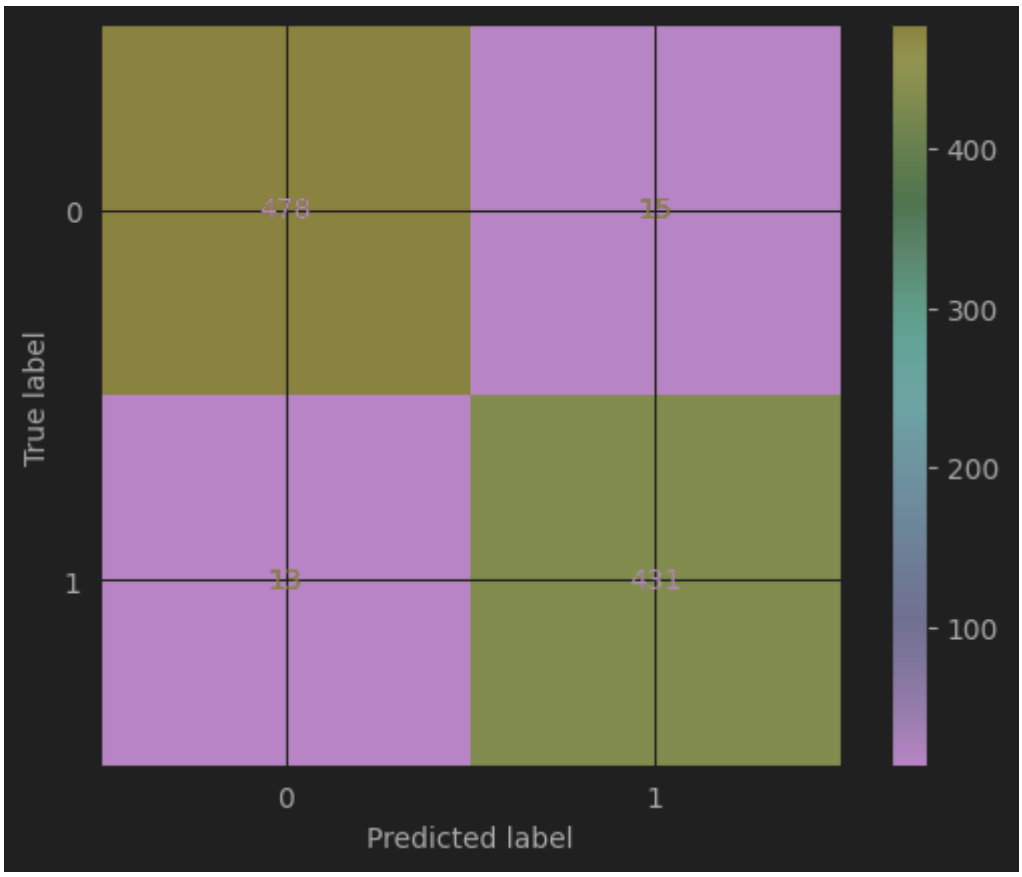
```
    # Save model results to array
    model_name = "SVM"

    # Get the best model from grid search
    best_dnn_estimator = grid_search.best_estimator_

    # Store the best parameters and performance metrics
    best_dnn_params = grid_search.best_params_
    best_dnn_accuracy = grid_search.best_score_
    dnn_cv_scores = grid_search.cv_results_['mean_test_score']

    # Save the results of the best model
    model_results[model_name].append({'name': model_name, 'cv-fold': folds, 'model':
best_svm_model, 'best_estimator': best_dnn_estimator, 'best_params': best_dnn_params,
'accuracy': best_dnn_accuracy, 'cv_scores': dnn_cv_scores})

    # Evaluate the best model
evaluate_model(model_name, best_svm_model, norm_test_f, y_test, cv)
```

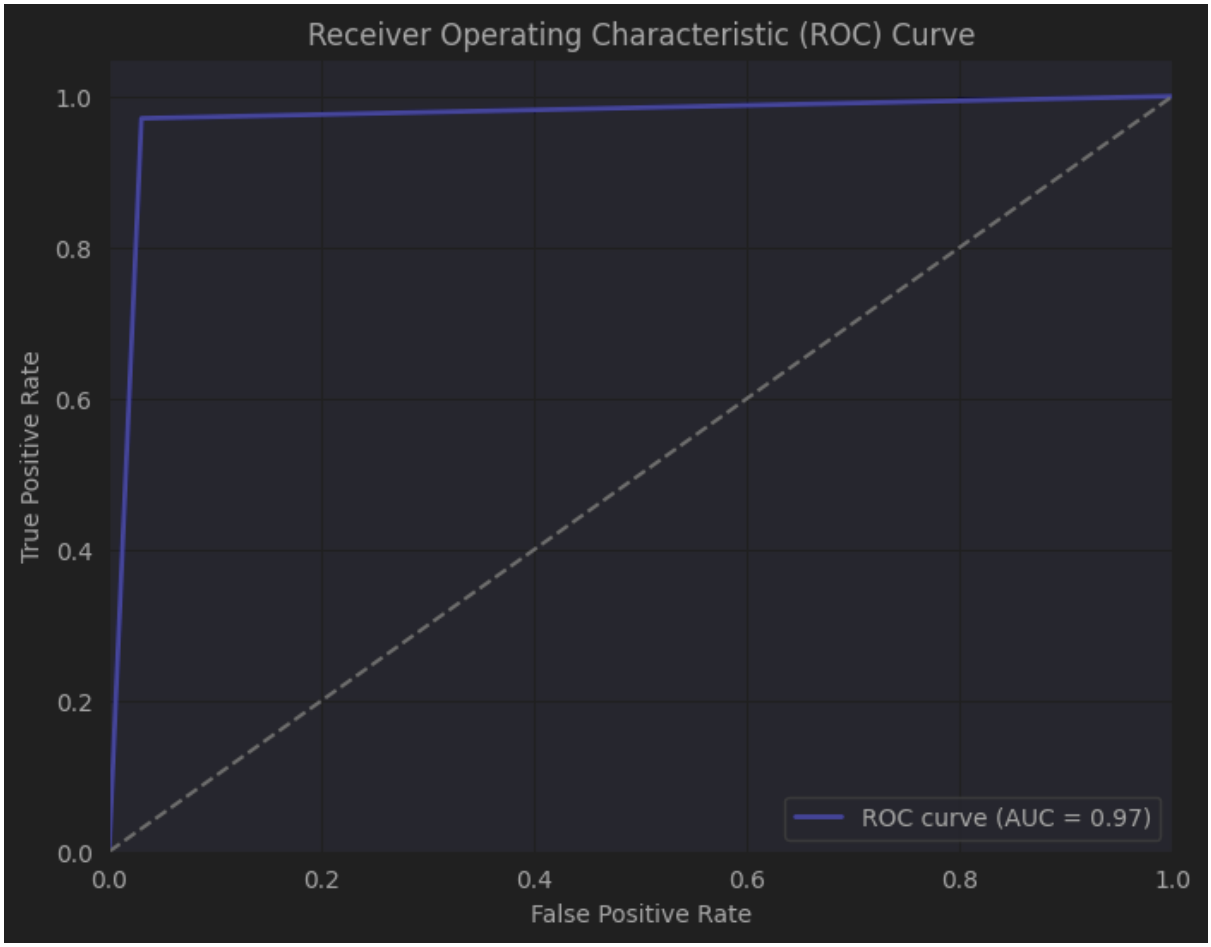
Output:

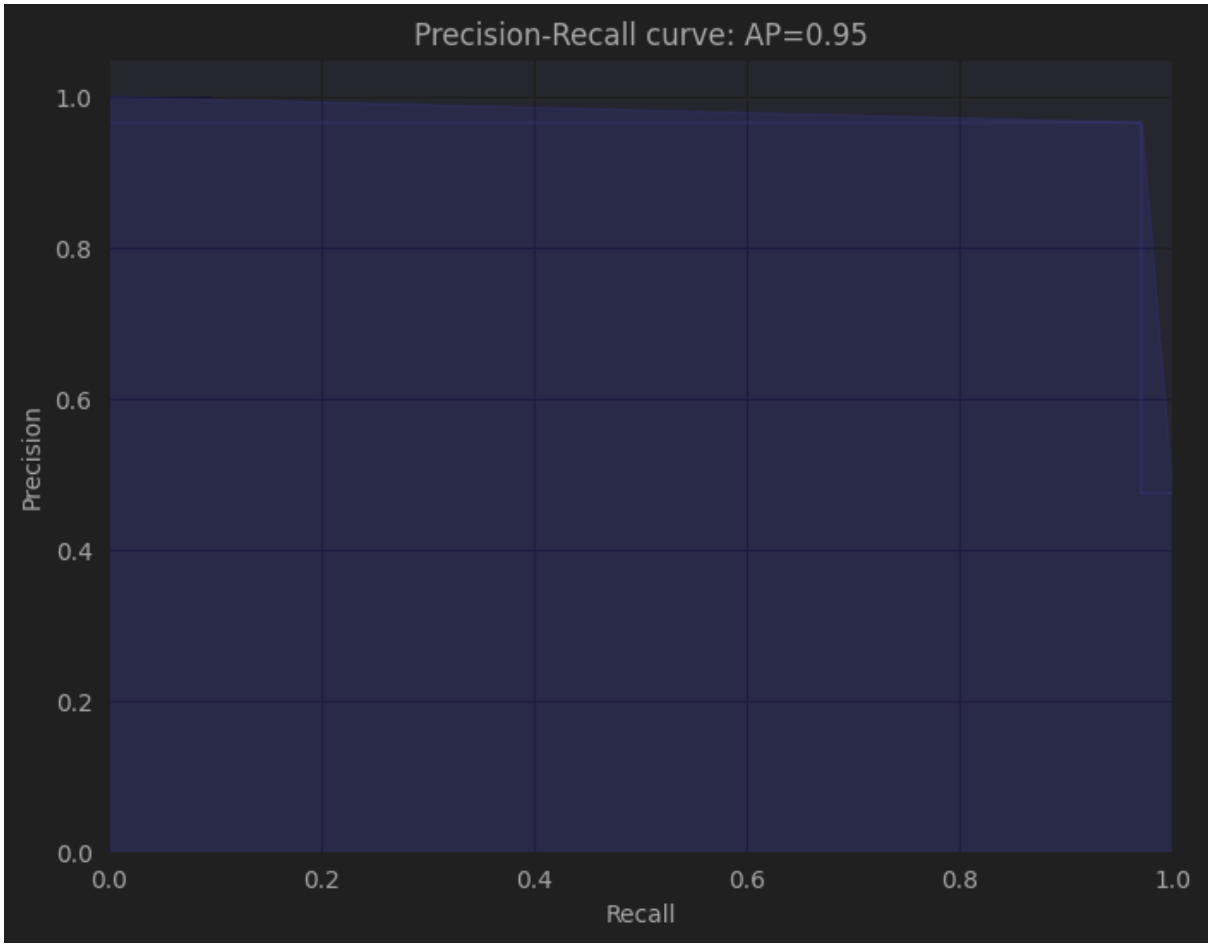
```
Number of Folds: 3
Best Parameters: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
Model name:  SVM
Cross-Validation Scores: [0.93929712 0.95192308 0.94871795 0.93610224 0.95833333
0.96474359
 0.95527157 0.95192308 0.94230769]
Mean Cross-Validation Score: 0.949846627162921
Skipping model performance, loss and accuracy plots
Accuracy: 0.9701173959445037
Precision: 0.9663677130044843
Recall: 0.9707207207207207
F1-score: 0.9685393258426966
```

	precision	*recall*	*f1-score*	*support*
0	*0.97*	*0.97*	*0.97*	*493*
1	*0.97*	*0.97*	*0.97*	*444*
accuracy			*0.97*	*937*
macro avg	*0.97*	*0.97*	*0.97*	*937*
weighted avg	*0.97*	*0.97*	*0.97*	*937*

[[478 15]
[13 431]]
AUC-ROC Score: 0.9701473786159384




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.97)



Number of Folds: 4

Best Parameters: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}

Model name: SVM

Cross-Validation Scores: [0.96170213 0.95726496 0.95726496 0.95299145 0.94893617 0.94444444

0.94017094 0.97863248 0.97021277 0.94871795 0.94017094 0.93589744]

Mean Cross-Validation Score: 0.9530338849487786

Skipping model performance, loss and accuracy plots

Accuracy: 0.9701173959445037

Precision: 0.9663677130044843

Recall: 0.9707207207207207

F1-score: 0.9685393258426966

	precision	*recall*	*f1-score*	*support*
0	*0.97*	*0.97*	*0.97*	*493*
1	*0.97*	*0.97*	*0.97*	*444*

accuracy 0.97 937

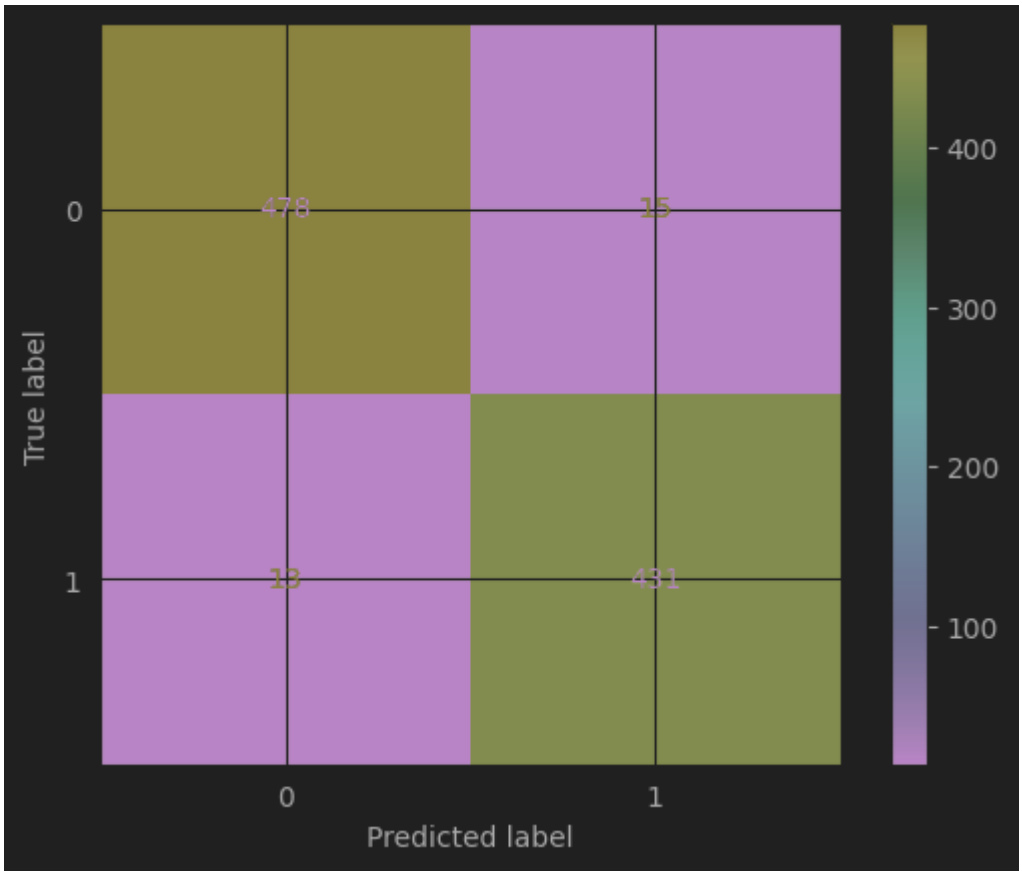
macro avg 0.97 0.97 0.97 937

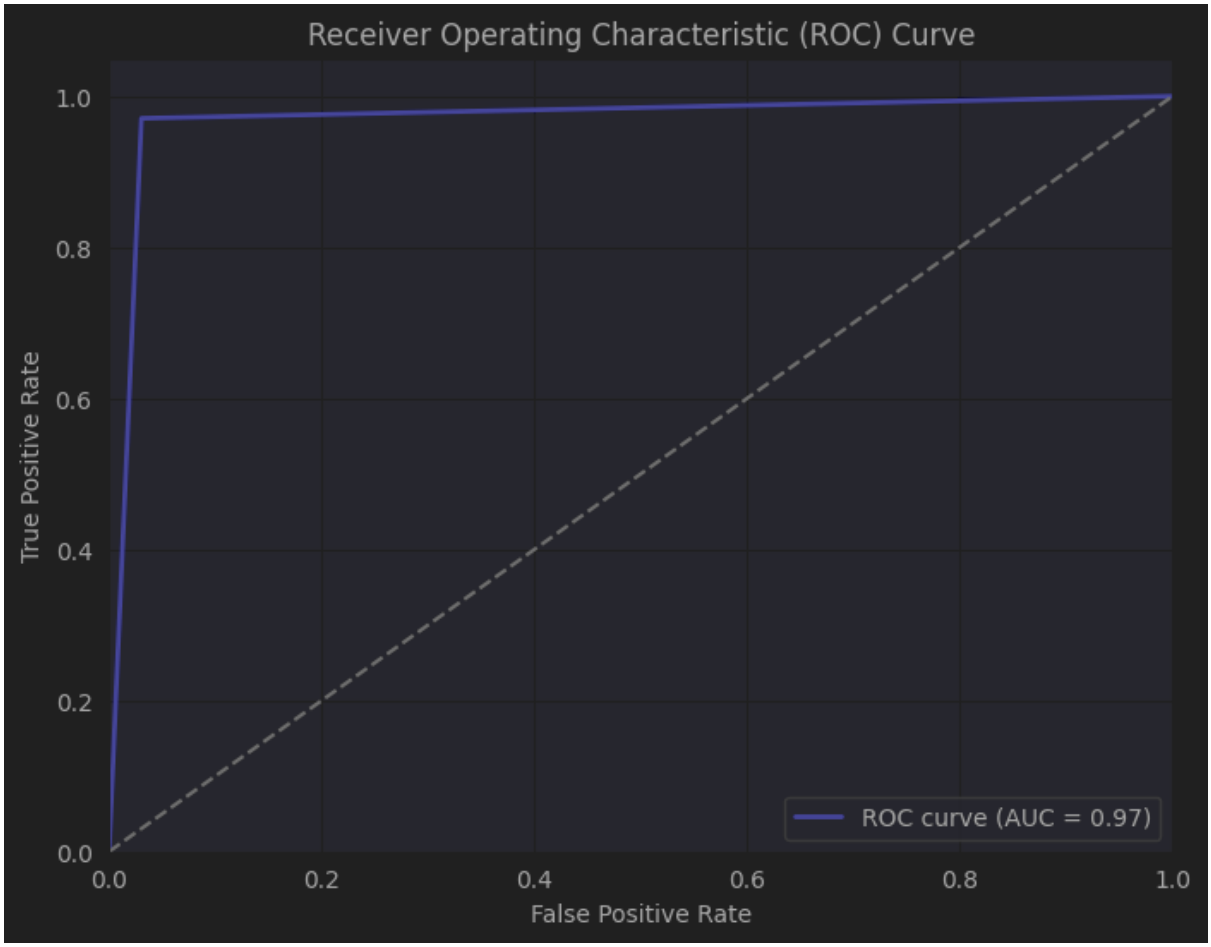
weighted avg 0.97 0.97 0.97 937

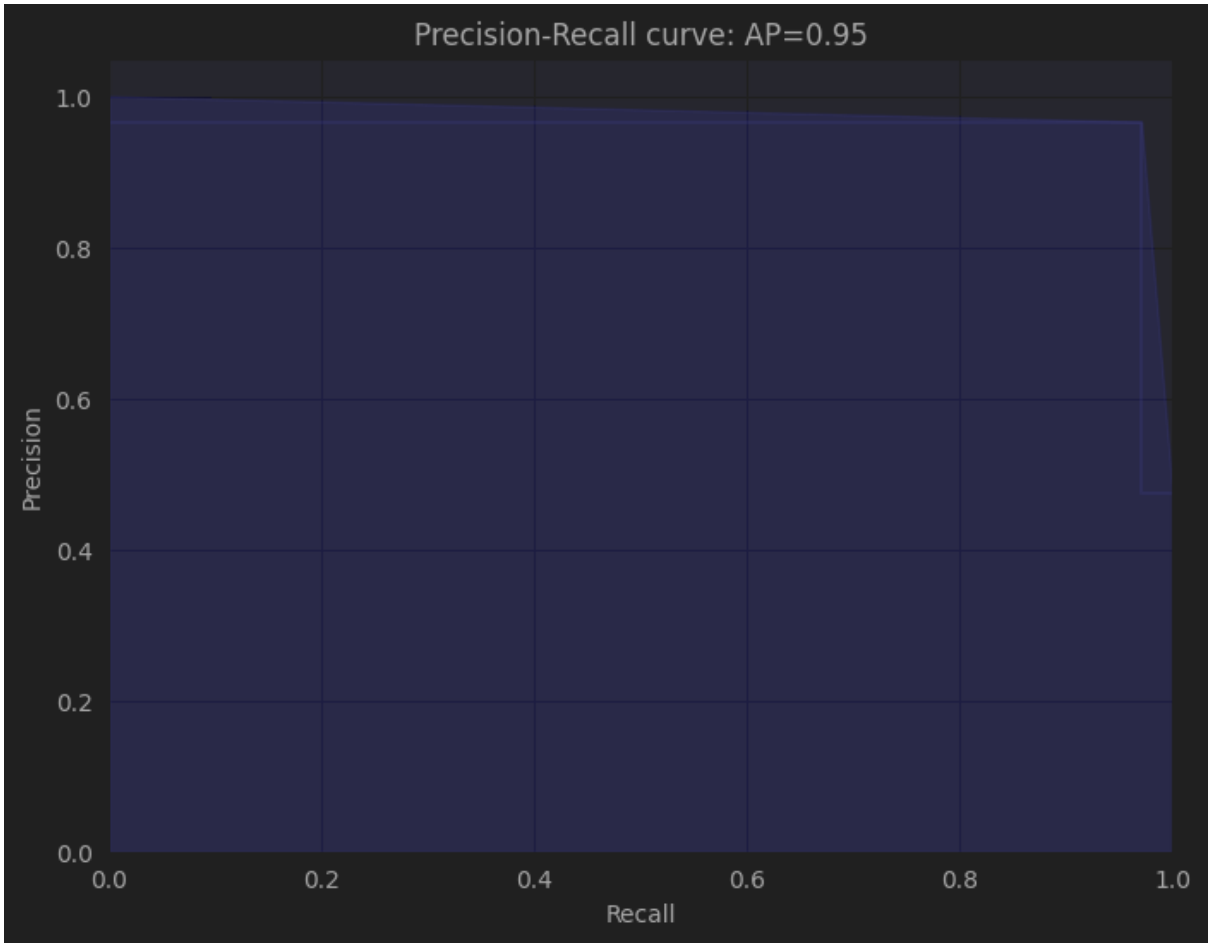
[[478 15]

[13 431]]

AUC-ROC Score: 0.9701473786159384




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.97)



Number of Folds: 5

Best Parameters: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}

Model name: SVM

Cross-Validation Scores: [0.95744681 0.95744681 0.94652406 0.94117647 0.95187166 0.94148936

0.95212766 0.93048128 0.95721925 0.96791444 0.97340426 0.96276596

0.96256684 0.93582888 0.92513369]

Mean Cross-Validation Score: 0.9508931619069293

Skipping model performance, loss and accuracy plots

Accuracy: 0.9711846318036286

Precision: 0.9664429530201343

Recall: 0.972972972972973

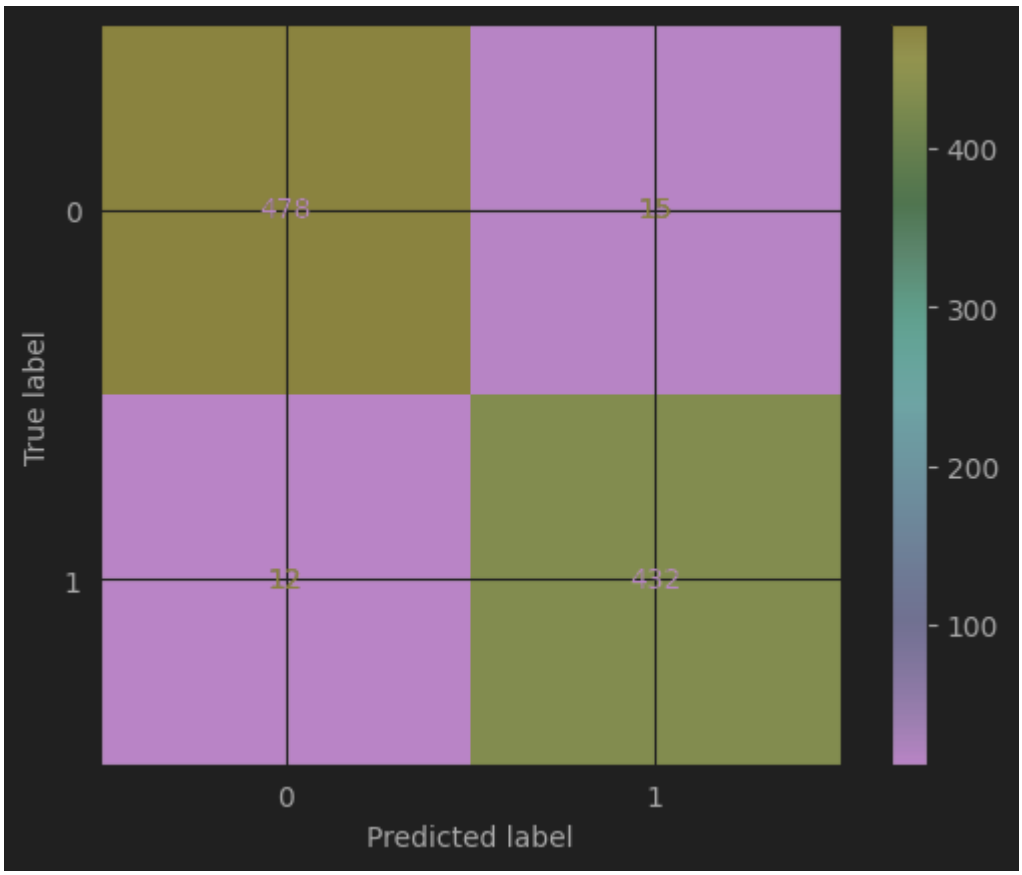
F1-score: 0.9696969696969697

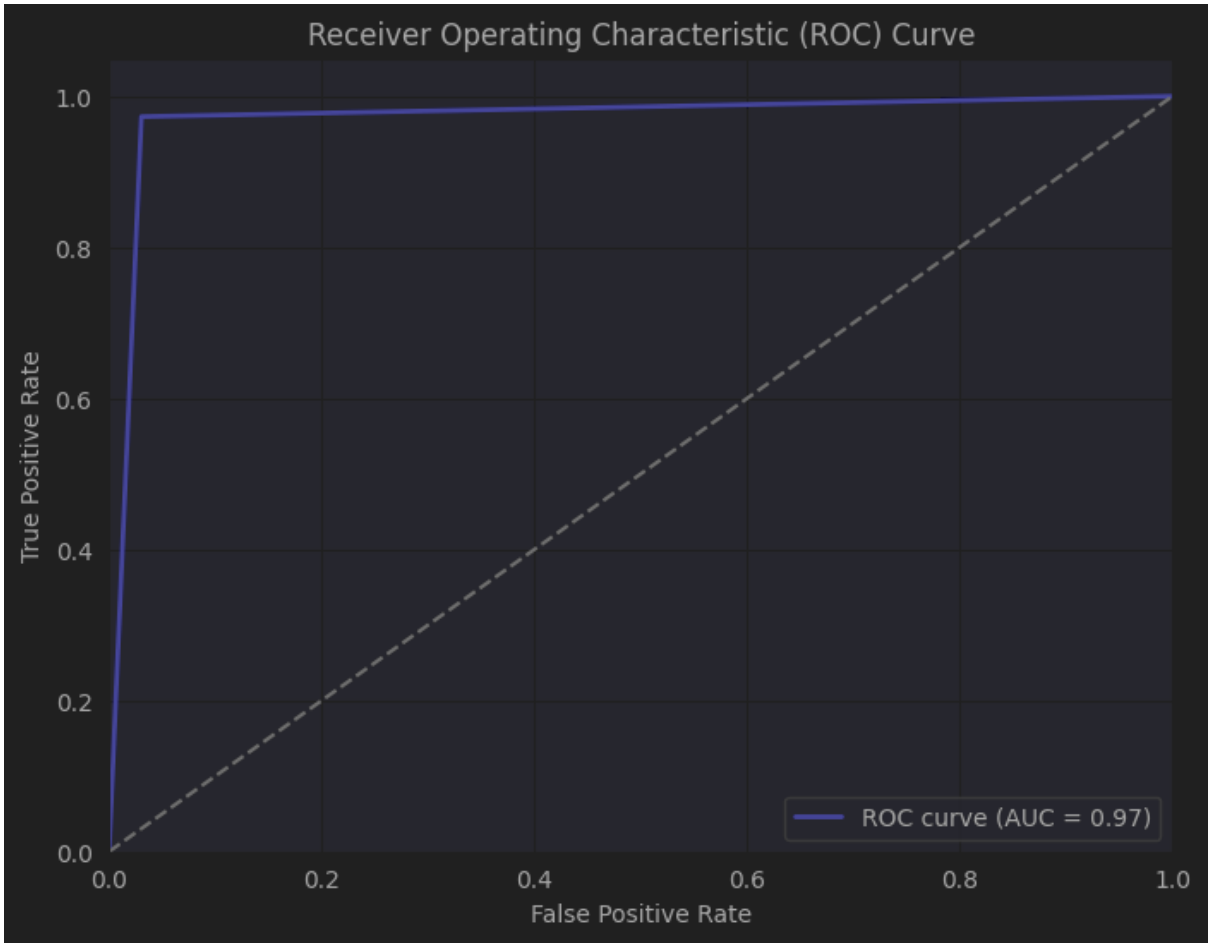
	precision	*recall*	*f1-score*	*support*
0	*0.98*	*0.97*	*0.97*	*493*

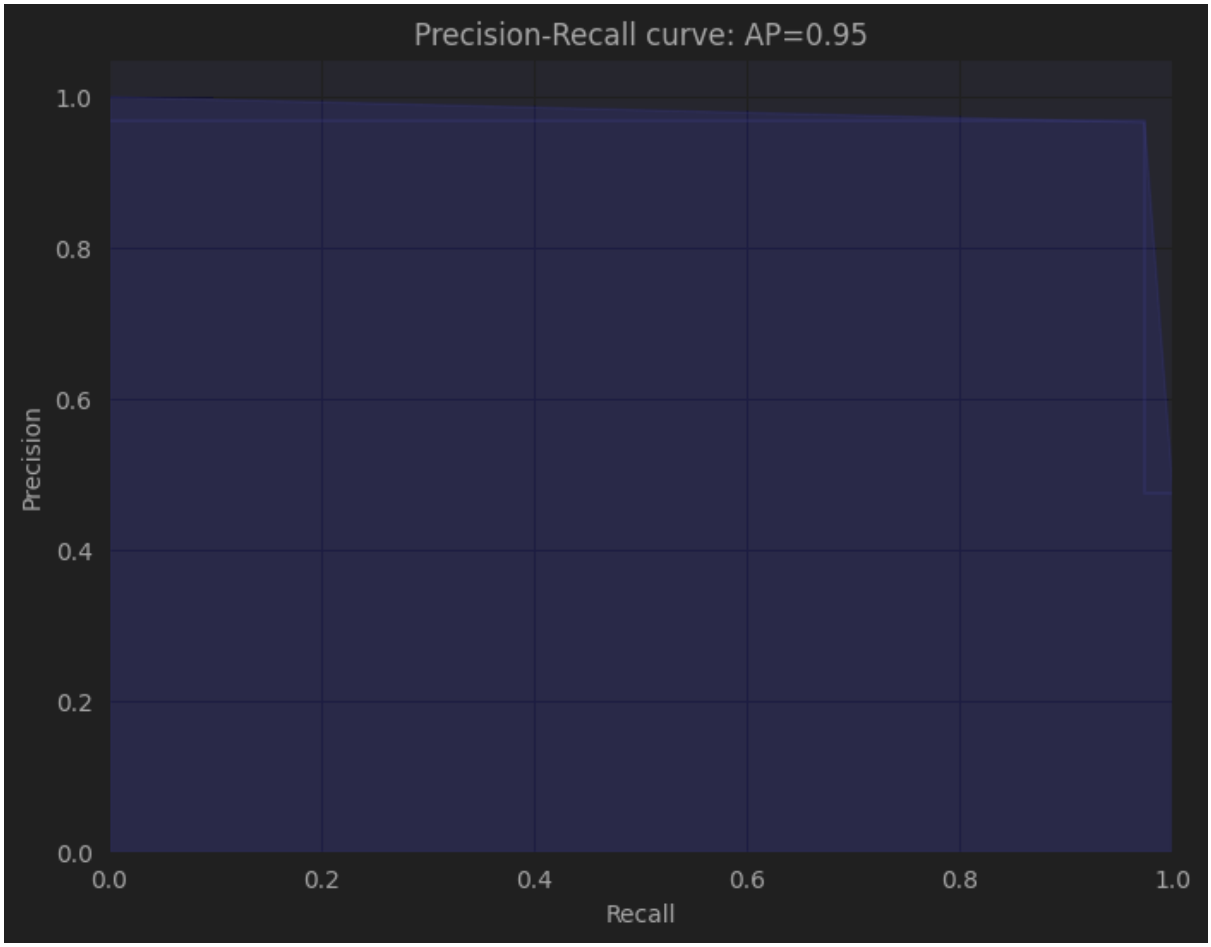
1 0.97 0.97 0.97 444

accuracy 0.97 937
macro avg 0.97 0.97 0.97 937
weighted avg 0.97 0.97 0.97 937

[[478 15]
[12 432]]
AUC-ROC Score: 0.9712735047420646




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.97)



Number of Folds: 10

Best Parameters: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}

Model name: SVM

Cross-Validation Scores: [0.95744681 0.96808511 0.94680851 0.94680851 0.95744681 0.95744681

0.95744681 0.91397849 0.94623656 0.94623656 0.93617021 0.94680851

0.95744681 0.94680851 0.93617021 0.92553191 0.95744681 0.94623656

0.97849462 0.97849462 0.9787234 0.96808511 0.94680851 0.94680851

0.95744681 0.95744681 0.91489362 0.95698925 0.92473118 0.91397849]

Mean Cross-Validation Score: 0.9491153816823001

Skipping model performance, loss and accuracy plots

Accuracy: 0.9701173959445037

Precision: 0.9663677130044843

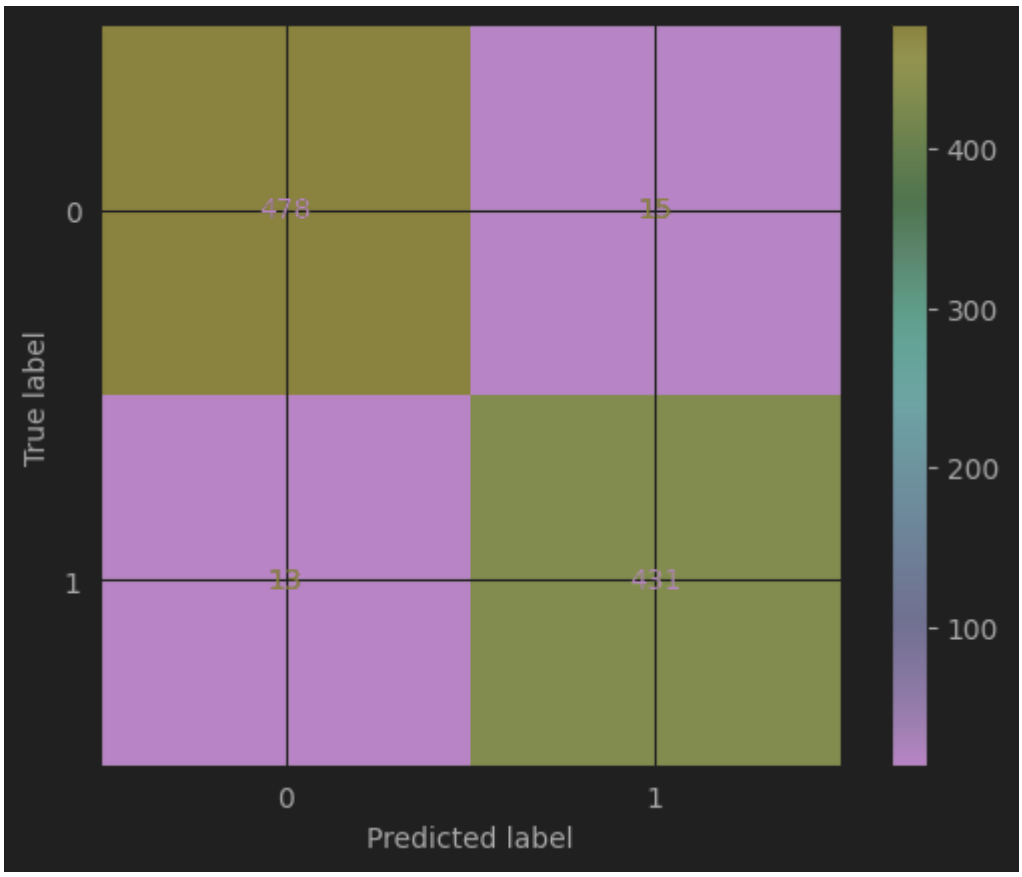
Recall: 0.9707207207207207

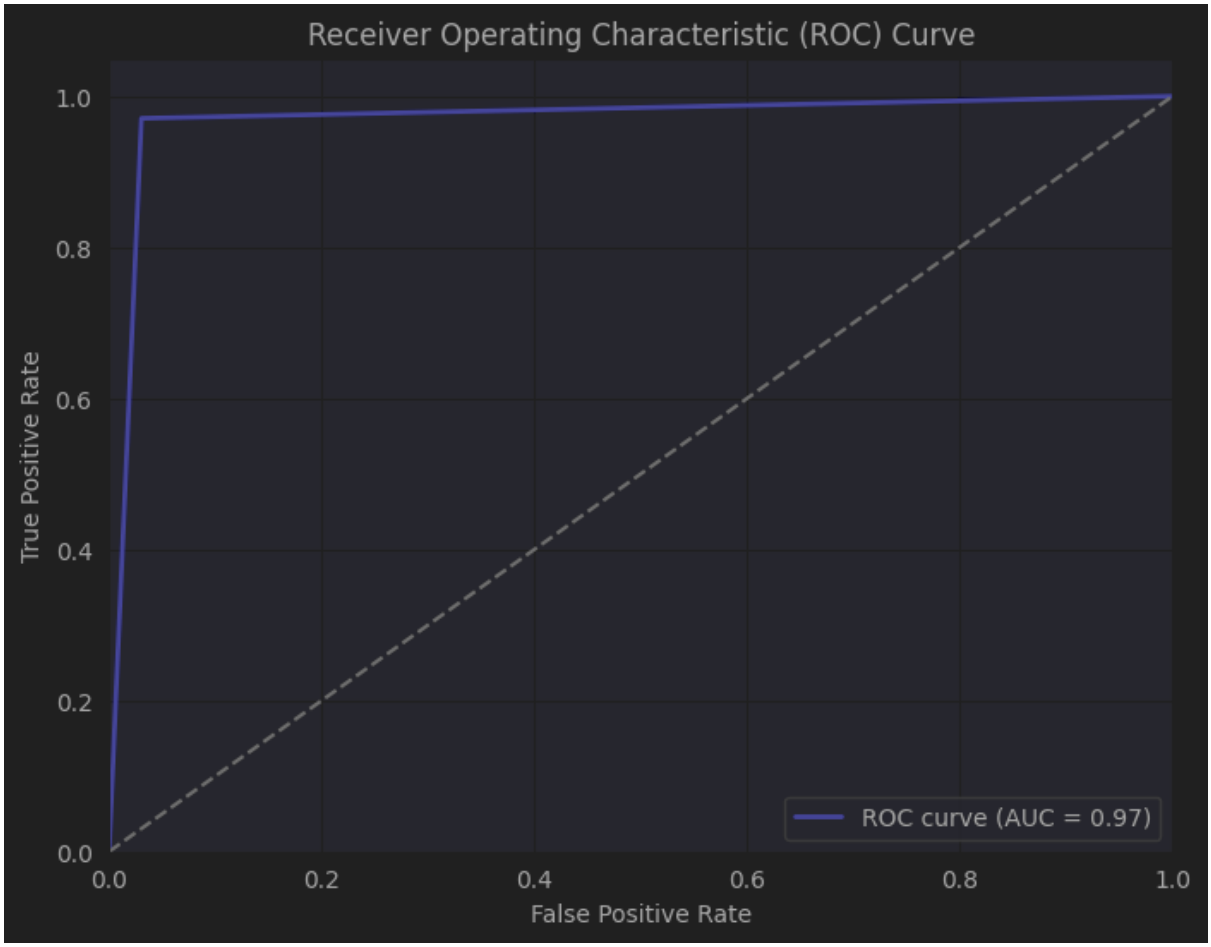
F1-score: 0.9685393258426966

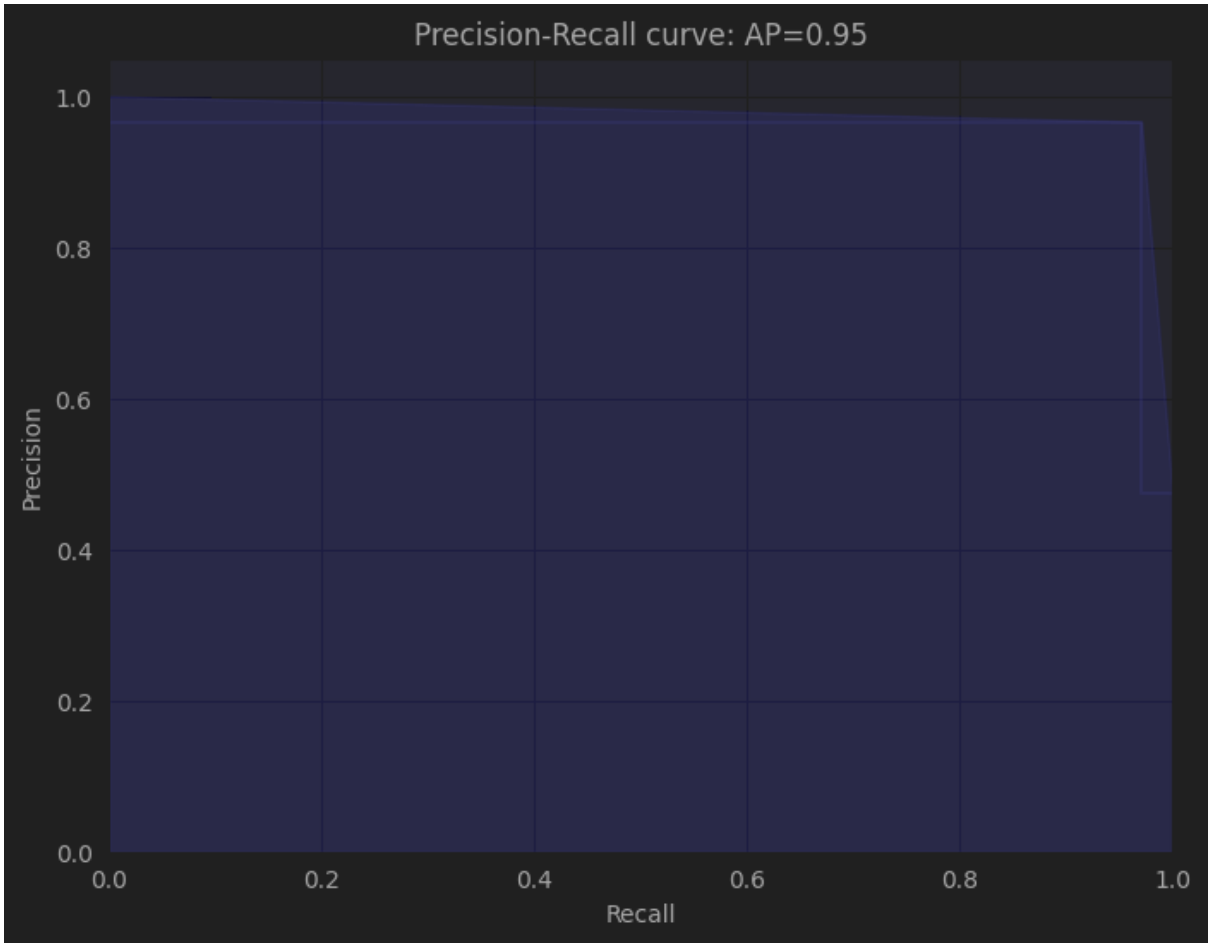
	precision	*recall*	*f1-score*	*support*

0	*0.97*	*0.97*	*0.97*	*493*
1	*0.97*	*0.97*	*0.97*	*444*
accuracy			*0.97*	*937*
macro avg	*0.97*	*0.97*	*0.97*	*937*
weighted avg	*0.97*	*0.97*	*0.97*	*937*

[[478 15]
[13 431]]
AUC-ROC Score: 0.9701473786159384




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.97)


Precision-Recall curve: AP=0.95
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

4.6.5 EXTREME GRADIENT BOOSTING (XGBOOST)

```
import xgboost as xgb

# Define hyperparameters grid for tuning
xgb_params_grid = {
    'learning_rate': [0.01, 0.1, 0.5],
    'n_estimators': [100, 200],
    'subsample': [0.3, 0.5, 0.9],
    'max_depth': [2, 3, 4],
    'colsample_bytree': [0.3, 0.5, 0.7]
}

# Define different numbers of folds for cross-validation
num_folds = [3, 4, 5, 10]

for folds in num_folds:
    # Create RepeatedStratifiedKFold with the specified number of folds
    cv = RepeatedStratifiedKFold(n_splits=folds, n_repeats=3, random_state=42)
    # cv = folds

    # Define the logistic regression model
    xgb_c = xgb.XGBClassifier(tree_method='hist', random_state=42)

    # Perform grid search with repeated cross-validation
    grid_search = GridSearchCV(estimator=xgb_c, param_grid=xgb_params_grid, cv=cv,
scoring='accuracy', n_jobs=5)
    grid_search.fit(x_tr_resample, y_tr_resample)

    # Get the best model from grid search
    best_xgb = grid_search.best_estimator_

    # Print the best parameters found
```

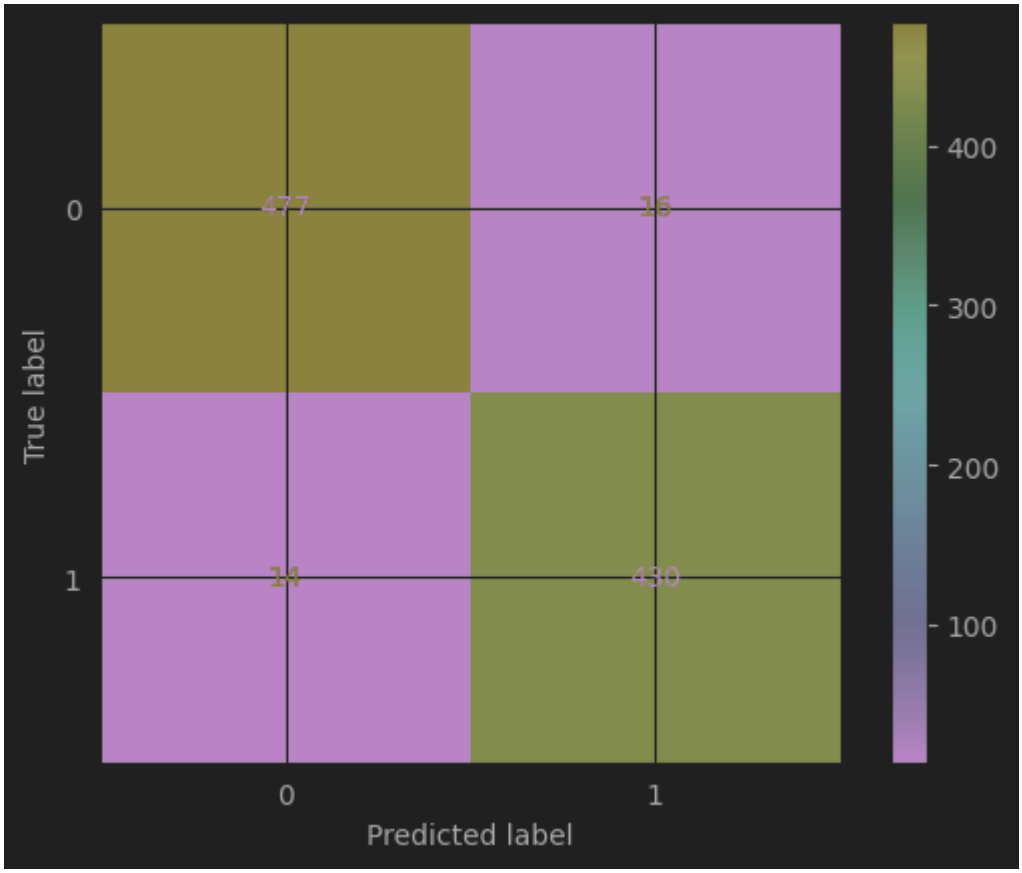
```
print(f"\nNumber of Folds: {folds}")
print("Best Parameters:", grid_search.best_params_)

# Use the best parameters to build the model
best_params = grid_search.best_params_
best_xgb_model = xgb.XGBClassifier(random_state=42, **best_params)

# Fit the model on the training data with the best parameters
best_xgb_model.fit(x_tr_resample, y_tr_resample)

# Evaluate the best model
evaluate_model("XGBoost", best_xgb, norm_test_f, y_test, cv)
```

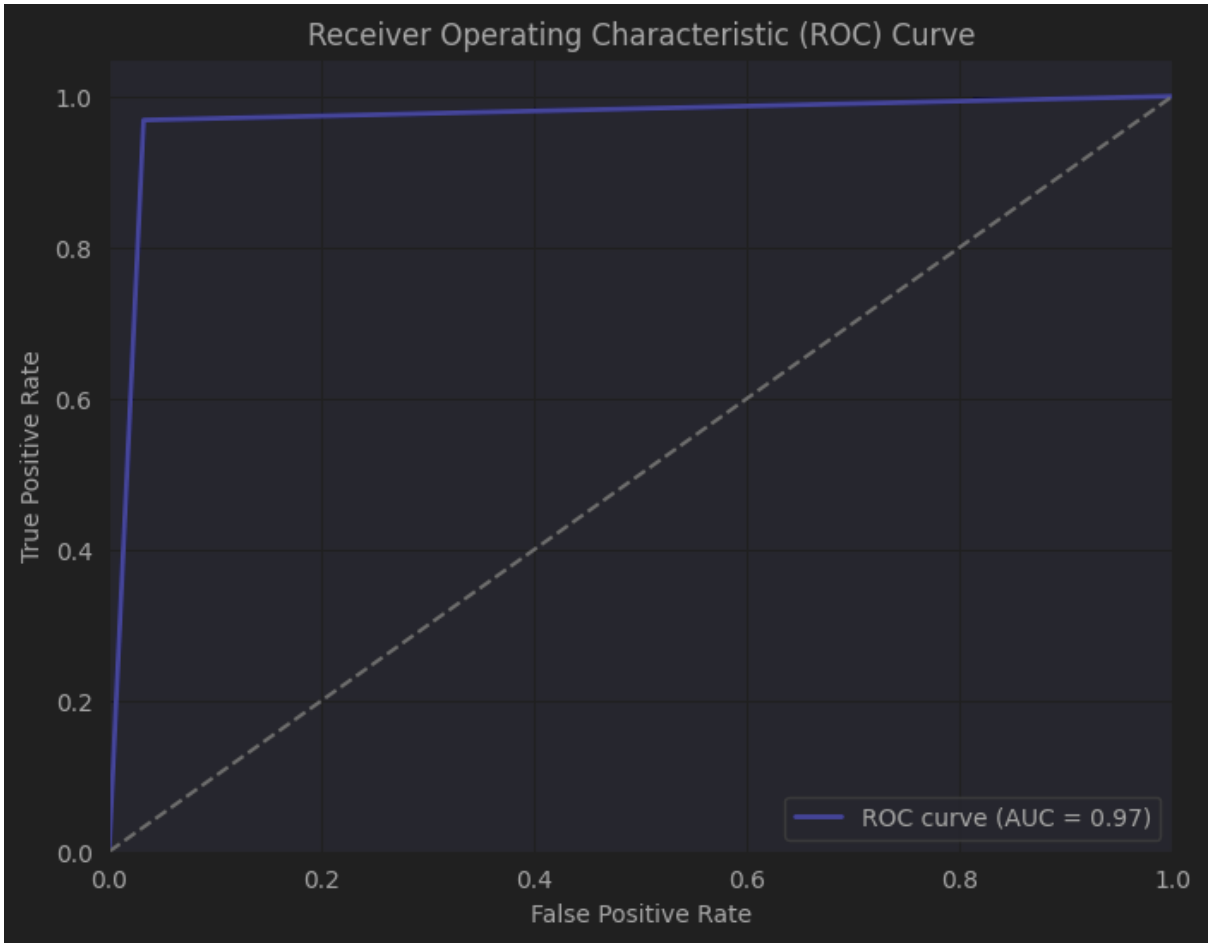
Output:

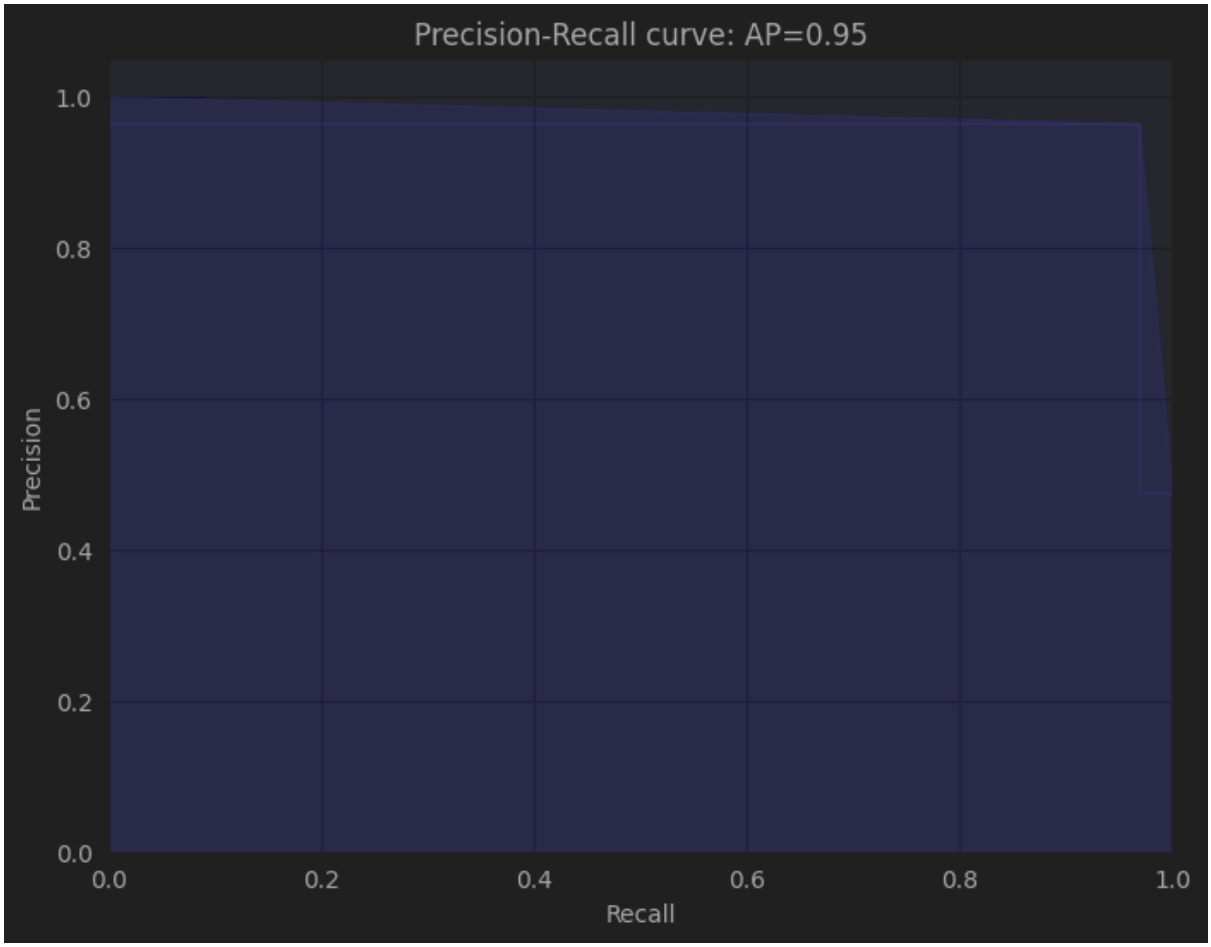
```
Number of Folds: 3
Best Parameters: {'colsample_bytree': 0.7, 'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 200, 'subsample': 0.9}
Model name:  XGBoost
Cross-Validation Scores: [0.95527157 0.95512821 0.93910256 0.93929712 0.95512821 0.96794872
 0.93929712 0.96153846 0.93589744]
Mean Cross-Validation Score: 0.9498454893822306
Skipping model performance, loss and accuracy plots
Accuracy: 0.967982924226254
Precision: 0.9641255605381166
Recall: 0.9684684684684685
F1-score: 0.9662921348314607
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       493
           1       0.96      0.97      0.97       444
```

accuracy			*0.97*	*937*
macro avg	*0.97*	*0.97*	*0.97*	*937*
weighted avg	*0.97*	*0.97*	*0.97*	*937*

[[477 16]
[14 430]]
AUC-ROC Score: 0.9680070537068509




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.97)



Number of Folds: 4

Best Parameters: {'colsample_bytree': 0.7, 'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 200, 'subsample': 0.5}

Model name: XGBoost

Cross-Validation Scores: [0.95319149 0.94871795 0.97008547 0.94017094 0.94468085 0.94444444

0.95299145 0.96153846 0.95319149 0.95726496 0.93589744 0.94871795]

Mean Cross-Validation Score: 0.9509077408013579

Skipping model performance, loss and accuracy plots

Accuracy: 0.9637139807897546

Precision: 0.9617117117117117

Recall: 0.9617117117117117

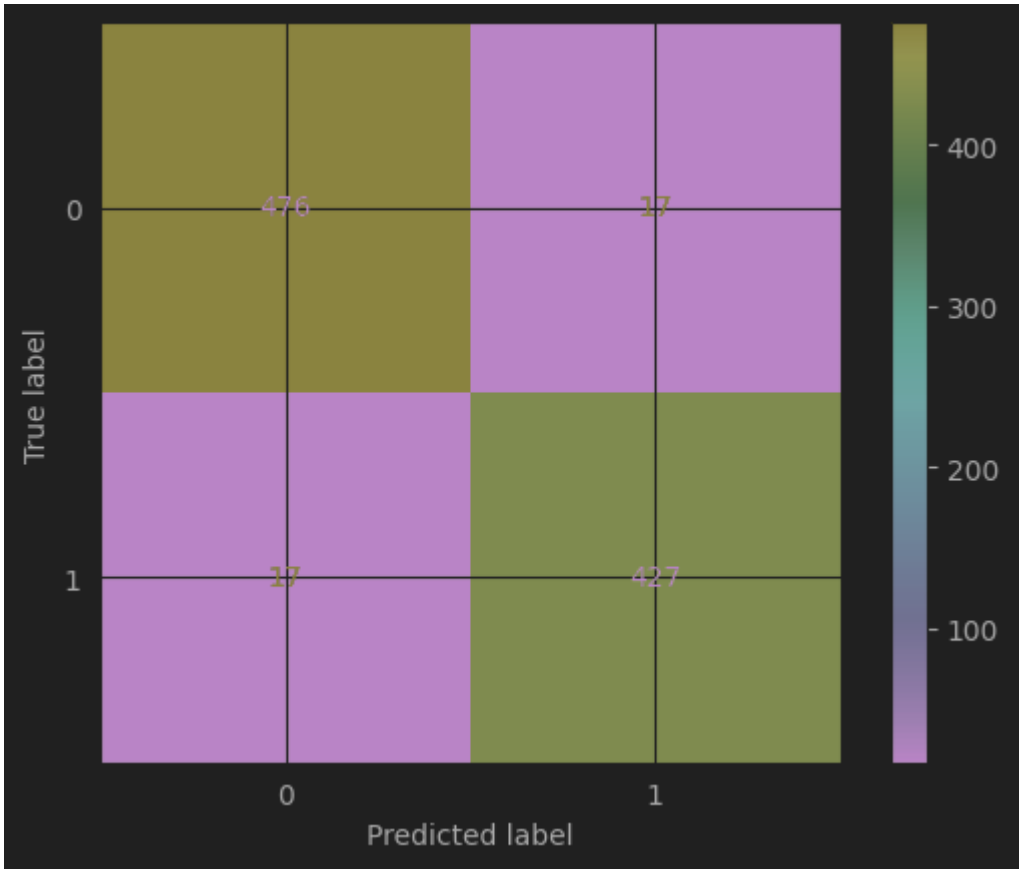
F1-score: 0.9617117117117117

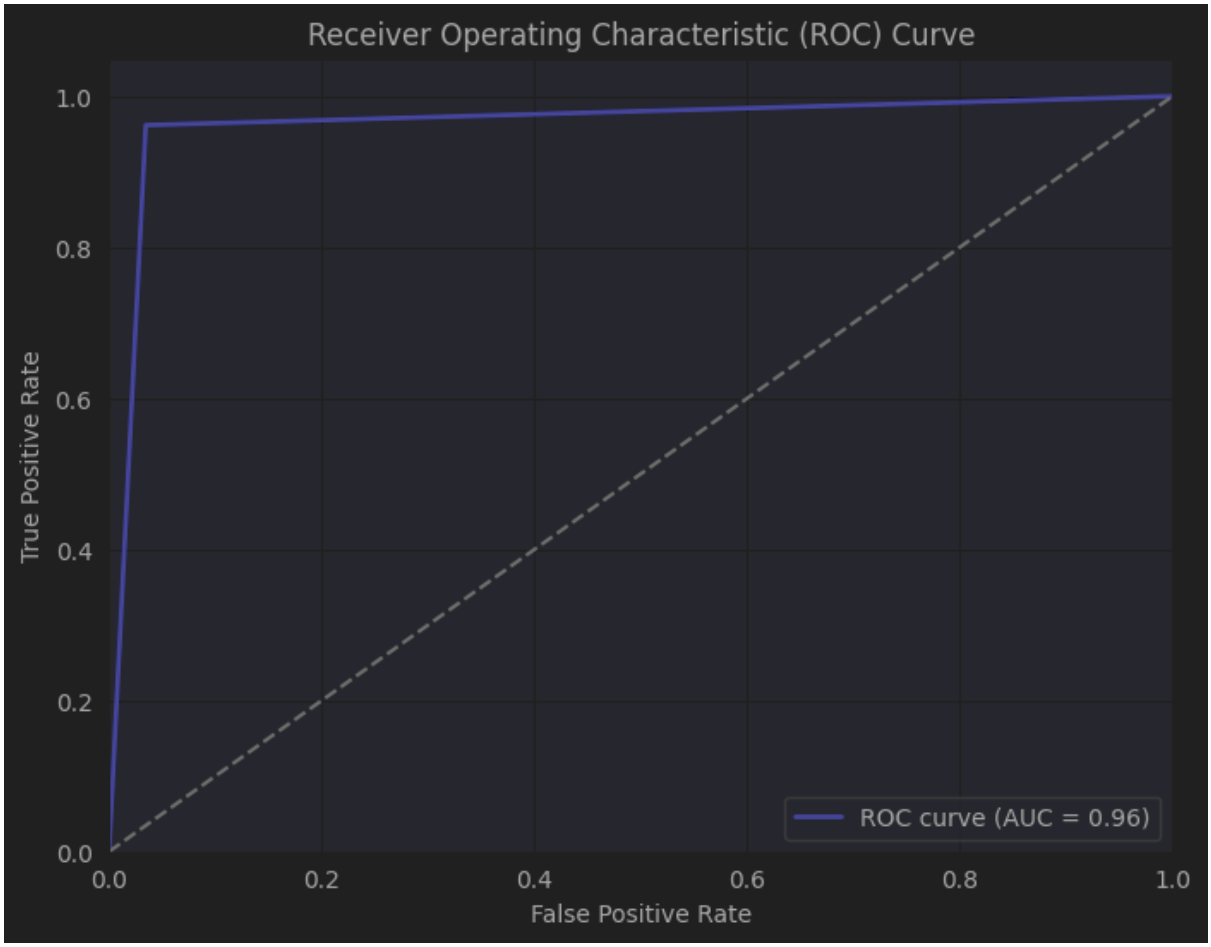
	precision	*recall*	*f1-score*	*support*
0	*0.97*	*0.97*	*0.97*	*493*

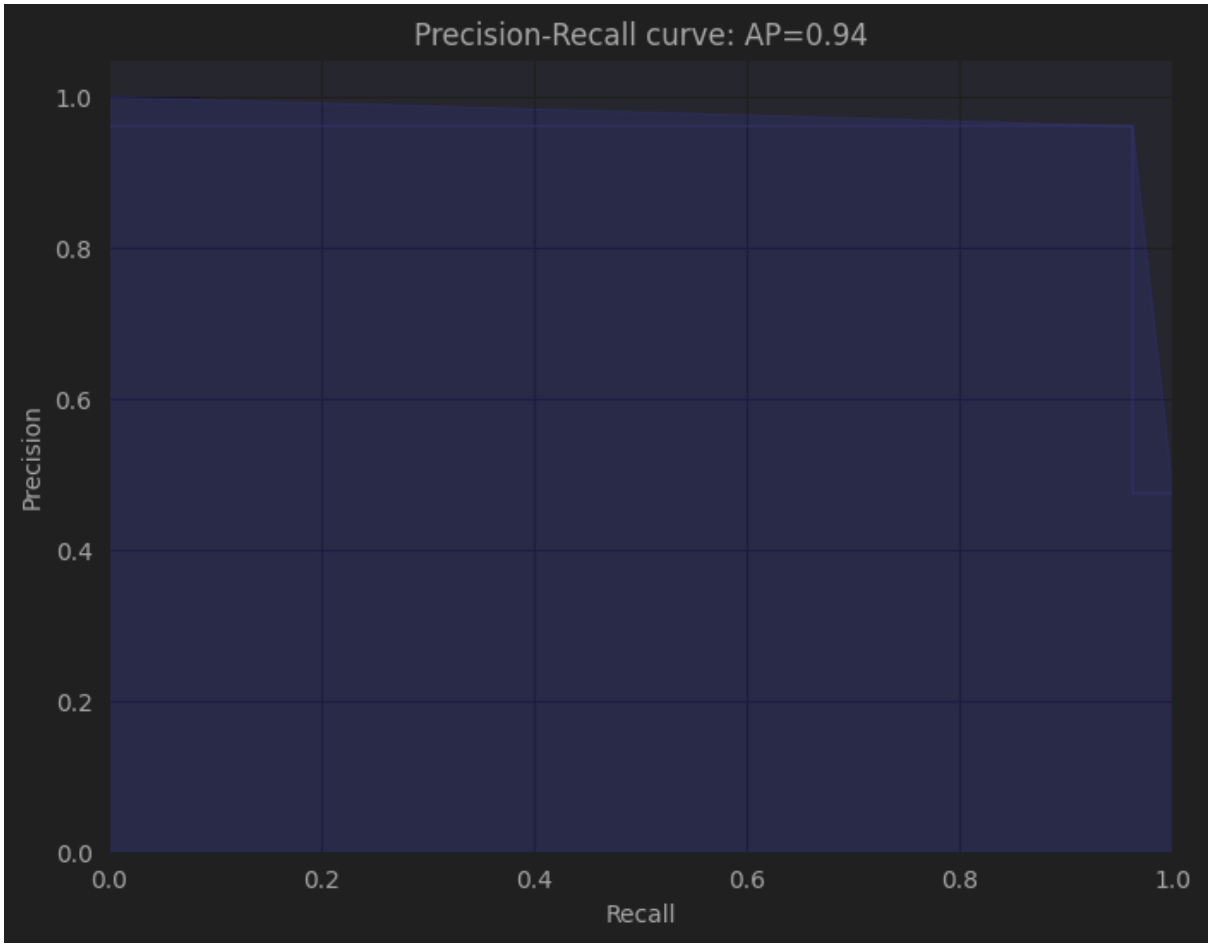
1 0.96 0.96 0.96 444

accuracy 0.96 937
macro avg 0.96 0.96 0.96 937
weighted avg 0.96 0.96 0.96 937

[[476 17]
[17 427]]
AUC-ROC Score: 0.9636144765455109




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.96)



Number of Folds: 5

Best Parameters: {'colsample_bytree': 0.7, 'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 200, 'subsample': 0.9}

Model name: XGBoost

Cross-Validation Scores: [0.96276596 0.94680851 0.94652406 0.97326203 0.94652406 0.94680851

0.94680851 0.94117647 0.97326203 0.96256684 0.95212766 0.96808511

0.96256684 0.93048128 0.95721925]

Mean Cross-Validation Score: 0.9544658095346457

Skipping model performance, loss and accuracy plots

Accuracy: 0.967982924226254

Precision: 0.9641255605381166

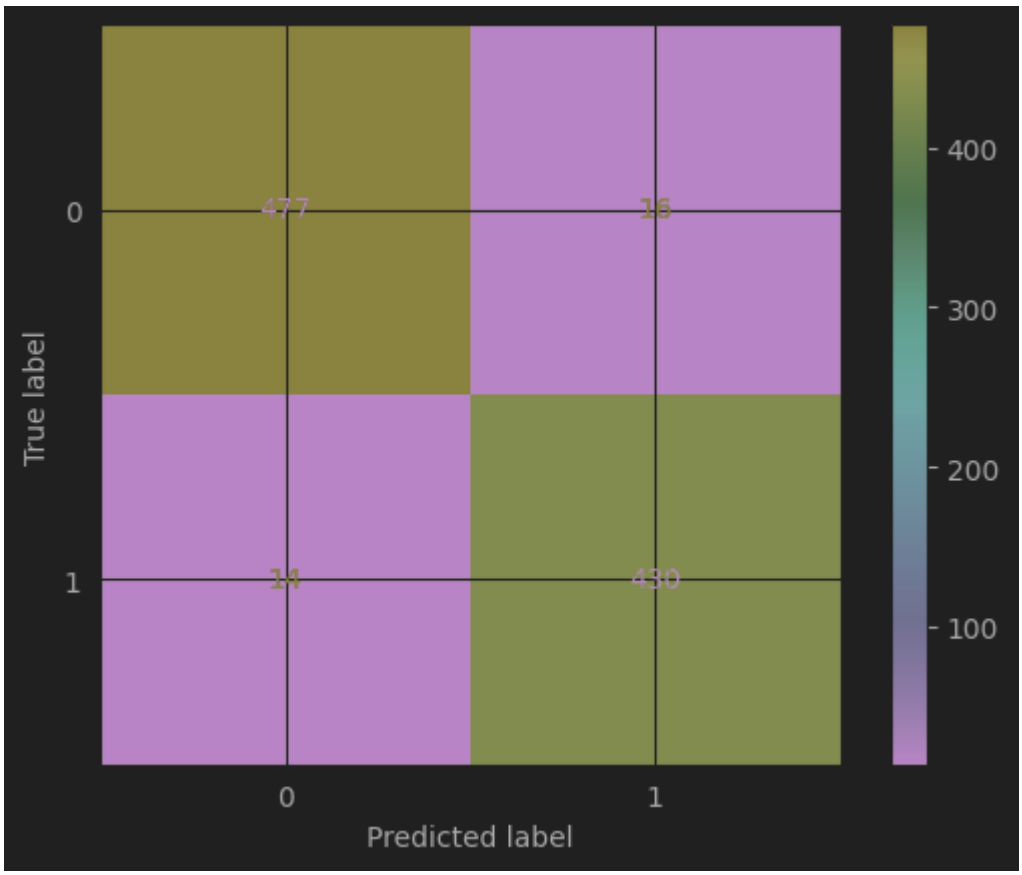
Recall: 0.9684684684684685

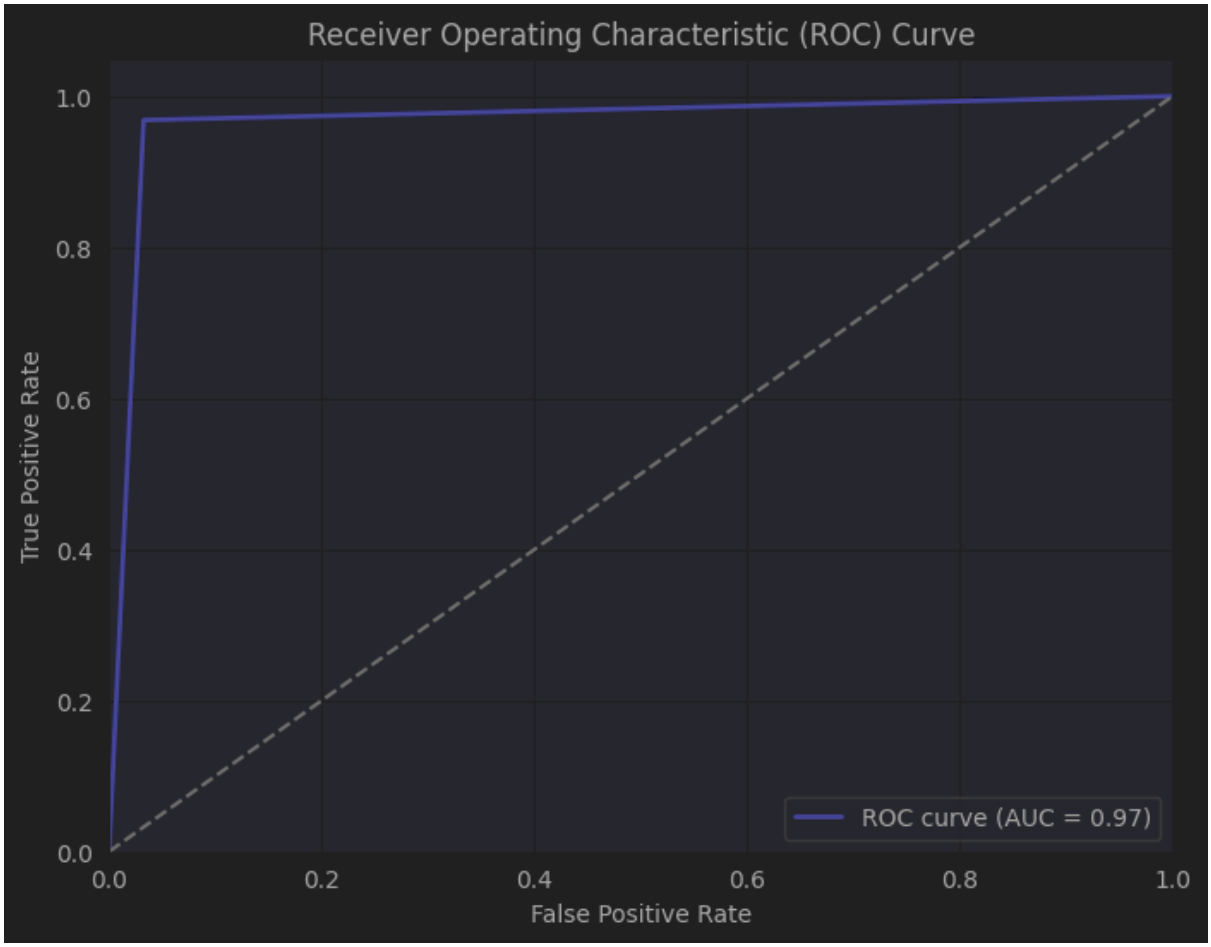
F1-score: 0.9662921348314607

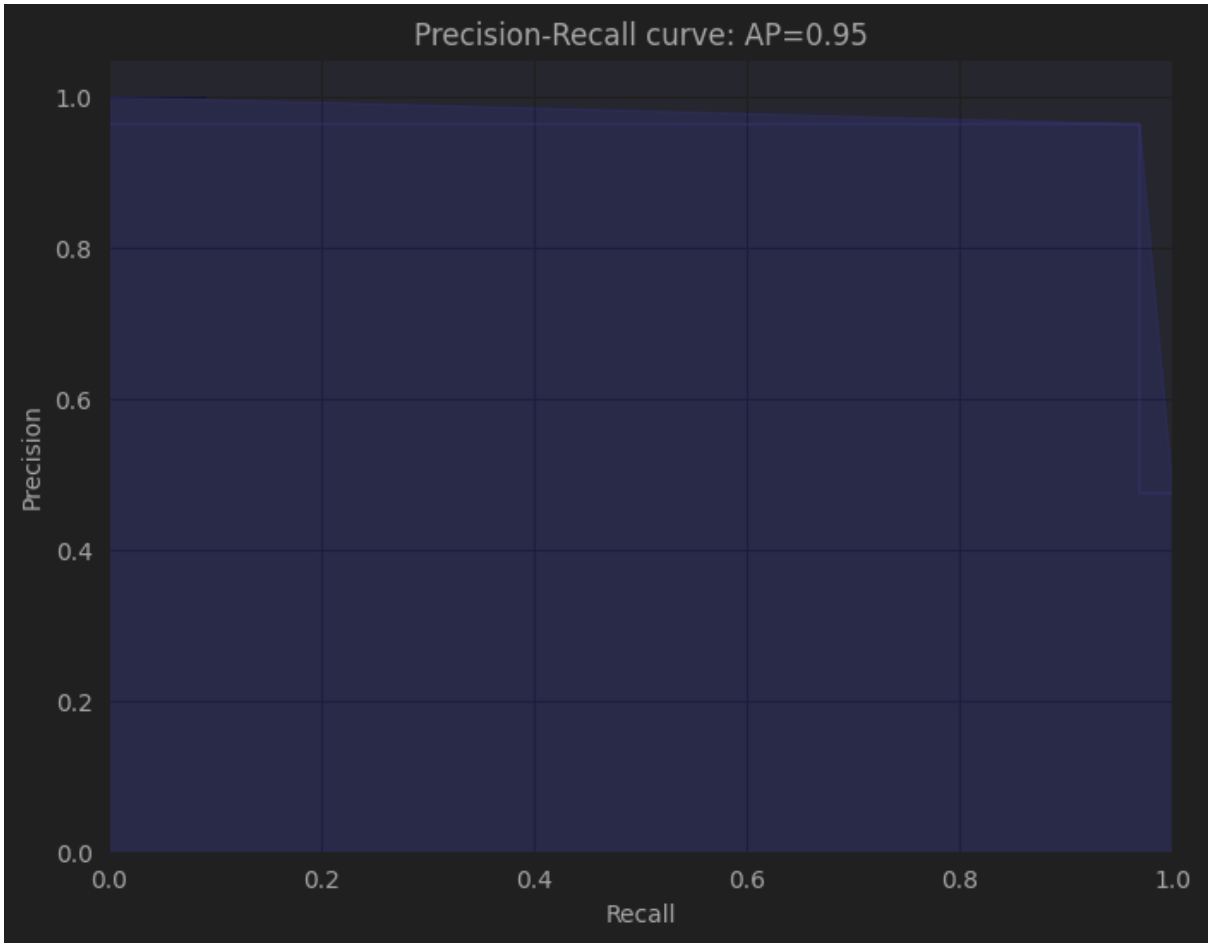
	precision	*recall*	*f1-score*	*support*

0	*0.97*	*0.97*	*0.97*	*493*
1	*0.96*	*0.97*	*0.97*	*444*
accuracy			*0.97*	*937*
macro avg	*0.97*	*0.97*	*0.97*	*937*
weighted avg	*0.97*	*0.97*	*0.97*	*937*

[[477 16]
[14 430]]
AUC-ROC Score: 0.9680070537068509




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.97)



Number of Folds: 10

Best Parameters: {'colsample_bytree': 0.5, 'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 200, 'subsample': 0.9}

Model name: XGBoost

Cross-Validation Scores: [0.95744681 0.9787234 0.93617021 0.95744681 0.94680851 0.95744681
0.9787234 0.95698925 0.94623656 0.95698925 0.91489362 0.95744681
0.94680851 0.95744681 0.95744681 0.94680851 0.9787234 0.96774194
0.94623656 1. 0.92553191 0.95744681 0.95744681 0.96808511
0.96808511 0.96808511 0.92553191 0.94623656 0.94623656 0.96774194]

Mean Cross-Validation Score: 0.9558987264546633

Skipping model performance, loss and accuracy plots

Accuracy: 0.9690501600853789

Precision: 0.9662921348314607

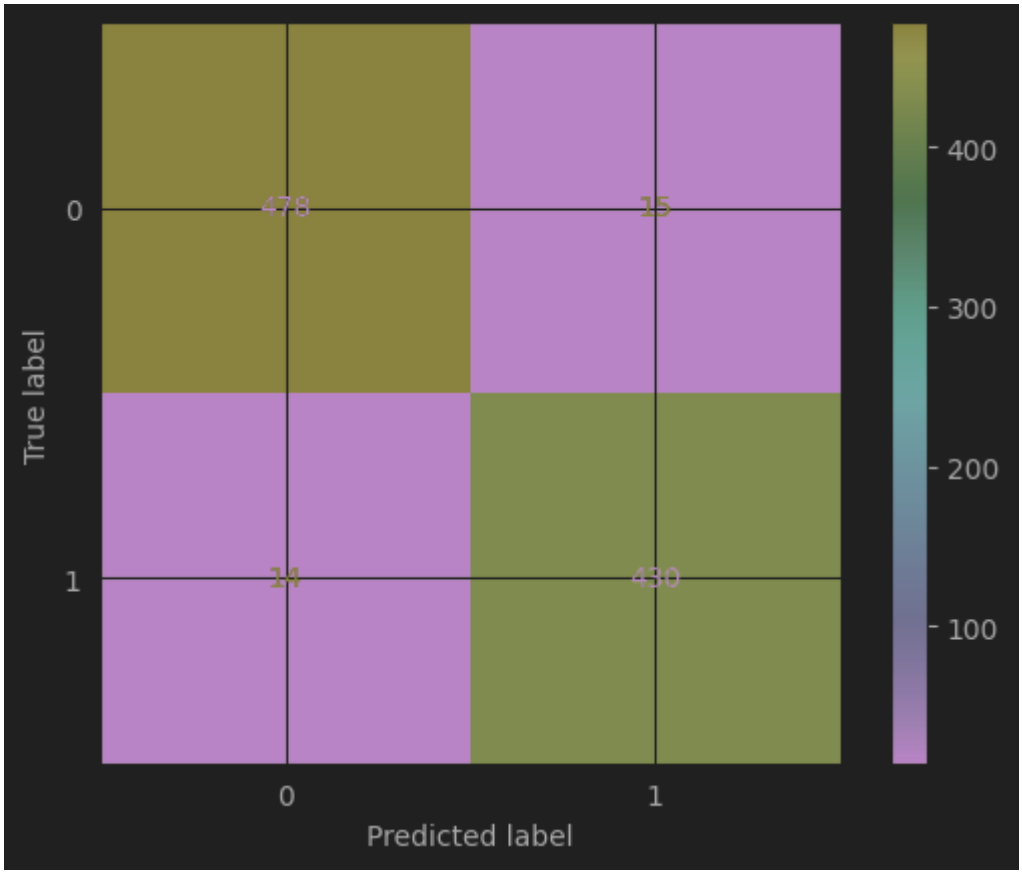
Recall: 0.9684684684684685

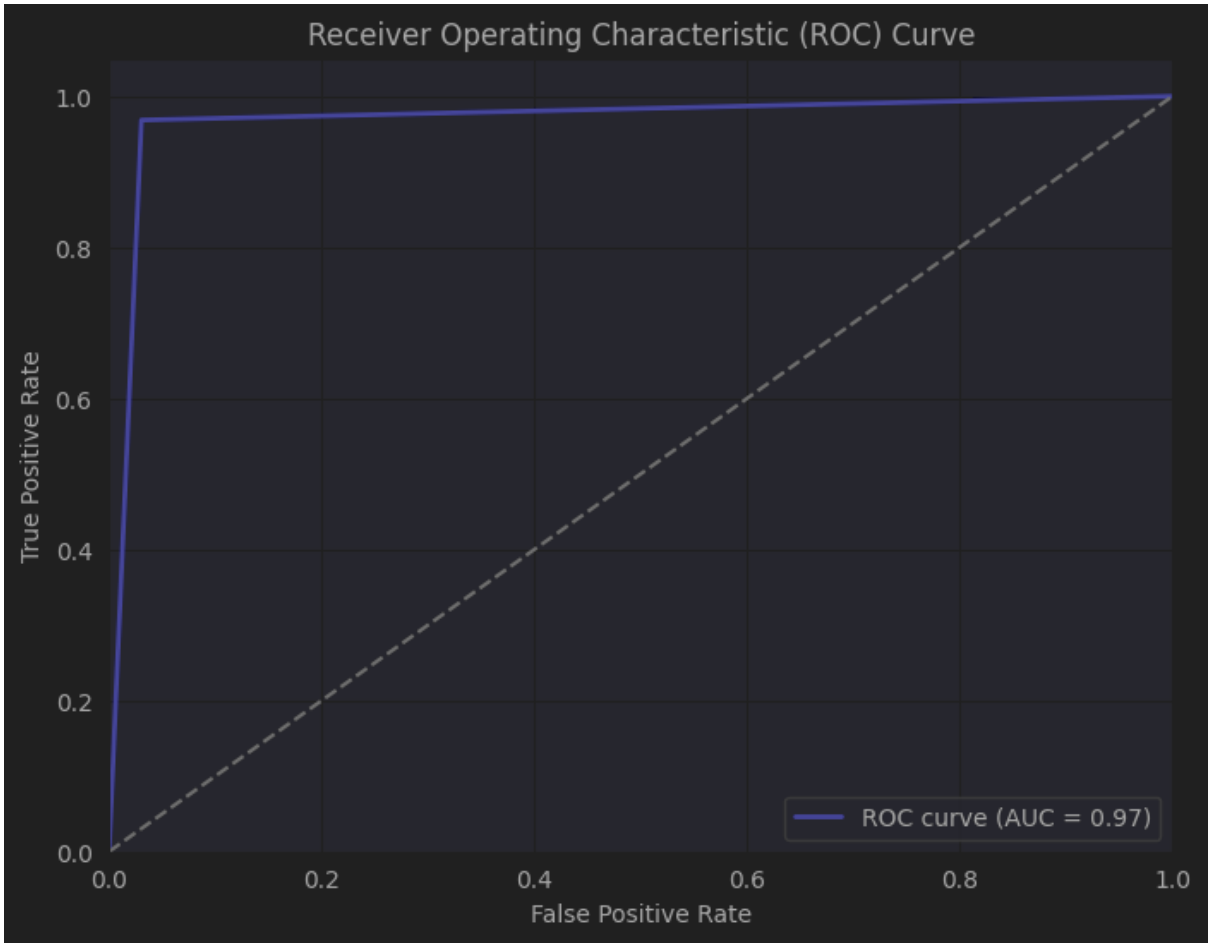
F1-score: 0.9673790776152981

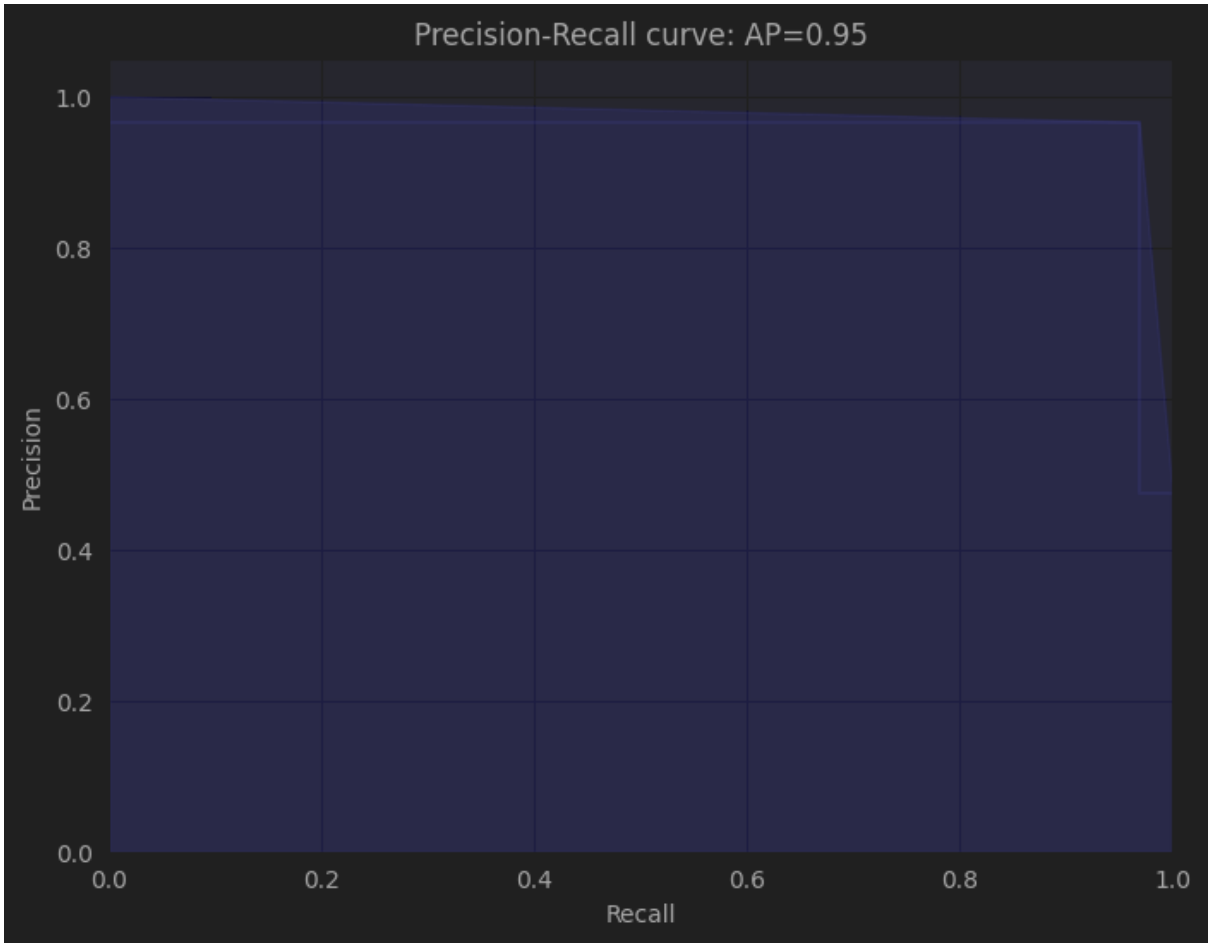
	precision	*recall*	*f1-score*	*support*
0	*0.97*	*0.97*	*0.97*	*493*
1	*0.97*	*0.97*	*0.97*	*444*
accuracy			*0.97*	*937*
macro avg	*0.97*	*0.97*	*0.97*	*937*
weighted avg	*0.97*	*0.97*	*0.97*	*937*

[[478 15]
[14 430]]

AUC-ROC Score: 0.9690212524898123




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.97)


Precision-Recall curve: AP=0.95
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

4.6.6 DEEP NEURAL NETWORK (DNN)

```
import tensorflow.python.platform.build_info as build
import tensorflow as tf

# DNN
from tensorflow.keras.models import Sequential
from tensorflow.keras import initializers, regularizers
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.wrappers.scikit_learn  import KerasClassifier

# Add the following code before importing any Keras modules
from tensorflow import random
```

4.6.6.1 SWITCH TO CUDA GPU (OPTIONAL)

```
print(build.build_info)
```

Output:

```
OrderedDict([('cpu_compiler',
'/home/builder/ktietz/aggregate/tensorflow_recipes/ci_cpu/tensorflow-
base_1614583966145/_build_env/bin/x86_64-conda_cos6-linux-gnu-gcc'),
('cuda_compute_capabilities', ['compute_35', 'compute_52', 'compute_60', 'compute_61',
'compute_70', 'compute_75']), ('cuda_version', '10.1'), ('cudnn_version', '7'), ('is_cuda_build',
True), ('is_rocm_build', False)])
```

tf.config.list_physical_devices('GPU')

Output:

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

```
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

Output:

```
Num GPUs Available:  1
```

Switch to GPU

```
config = tf.compat.v1.ConfigProto( device_count = {'GPU': 1 , 'CPU': 20} )
sess = tf.compat.v1.Session(config=config)
tf.compat.v1.keras.backend.set_session(tf.compat.v1.Session(config=config));
```

Output:

```
2024-05-09 14:02:20.925432: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-05-09 14:02:20.927850: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not creating XLA devices, tf_xla_enable_xla_devices not set
2024-05-09 14:02:20.928023: E tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:927] could not open file to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2024-05-09 14:02:20.928053: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1720] Found device 0 with properties:
pciBusID: 0000:01:00.0 name: NVIDIA GeForce RTX 4080 SUPER computeCapability: 8.9
coreClock: 2.565GHz coreCount: 80 deviceMemorySize: 15.99GiB deviceMemoryBandwidth: 685.51GiB/s
2024-05-09 14:02:20.928091: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudart.so.10.1
```

2024-05-09 14:02:20.928111: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcublas.so.10

2024-05-09 14:02:20.928118: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcublasLt.so.10

2024-05-09 14:02:20.928125: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcufft.so.10

2024-05-09 14:02:20.928140: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcurand.so.10

2024-05-09 14:02:20.928148: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcusolver.so.10

2024-05-09 14:02:20.928154: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcusparse.so.10

2024-05-09 14:02:20.928170: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudnn.so.7

2024-05-09 14:02:20.928195: E tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:927] could not open file to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.

2024-05-09 14:02:20.928221: E tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:927] could not open file to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.

2024-05-09 14:02:20.928234: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1862] Adding visible gpu devices: 0

```
2024-05-09 14:02:20.928596: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudart.so.10.1
```

The classification of the model is to determine the transaction is fraud or not. Hence, the output is set to 1, which means true or false.

```
# Number of outputs
# You need to know your columns, for example, you are only predicting the value is_malicious, so the value is 1.
no_of_output = 1



# Set the random seed for reproducibility
random.set_seed(42)



# Function to create model with given parameters
def create_model(optimizer='adam', activation='relu', loss='binary_crossentropy', metrics=['accuracy'],
                 L1=1024, L2=512, L3=256, L4=128, L5=64, dropout_rate=0.3, learning_rate=0.1, epochs=48, batch_size=128):
    model = Sequential()
    model.add(Dense(L1, input_dim=x_tr_resample.shape[1], activation=activation, kernel_initializer=initializers.RandomNormal(mean=0.0, stddev=0.05, seed=None)))
    model.add(Dropout(dropout_rate))
    model.add(Dense(L2, activation=activation, kernel_initializer=initializers.GlorotNormal(seed=None)))
    model.add(Dropout(dropout_rate))
    model.add(Dense(L3, activation=activation))
    model.add(Dropout(dropout_rate))
```

```
    model.add(Dense(L4, activation=activation, kernel_initializer=initializers.GlorotNormal(seed=None), kernel_regularizer=regularizers.l2(learning_rate)))
    model.add(Dropout(dropout_rate))
    model.add(Dense(L5, activation=activation, kernel_initializer=initializers.GlorotNormal(seed=None), kernel_regularizer=regularizers.l2(learning_rate)))
    model.add(Dropout(dropout_rate))
    model.add(Dense(no_of_output, activation=activation))
    model.compile(loss=loss, optimizer=optimizer, metrics=metrics)
    return model

# Define the EarlyStopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=3)

# Define the parameter grid
dnn_param_grid = {
    'optimizer': ['sgd', 'adam'], # 'sgd', 'adam'
    'activation': ['relu'], # 'softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear'
    'loss': ['binary_crossentropy'], # 'categorical_crossentropy', 'mean_squared_error'
    'metrics': [['accuracy']],
    'L1': [1024], # 1024, 512, 256, 128, 64
    'L2': [512],
    'L3': [256],
    'L4': [128],
    'L5': [64],
    'dropout_rate': [0.1], # 0, 0.1, 0.2, 0.3, 0.4, 0.5
    'learning_rate': [0.01],
    'epochs': [48, 64], # 64, 128
    'batch_size': [128] # 32, 64, 128
}

# Define different numbers of folds for cross-validation
```

```
num_folds = [3, 4, 5, 10]

for folds in num_folds:
    # Create RepeatedStratifiedKFold with the specified number of folds
    cv = RepeatedStratifiedKFold(n_splits=folds, n_repeats=3, random_state=42)
    # cv = folds

    dnn = KerasClassifier(build_fn=create_model)

    # Perform grid search with repeated cross-validation
    grid_search = GridSearchCV(estimator=dnn, param_grid=dnn_param_grid, cv=cv, scoring='accuracy', n_jobs=5)
    grid_search.fit(x_tr_resample, y_tr_resample)

    # Get the best model from grid search
    y_pred = grid_search.best_estimator_.predict(norm_test_f)

    # Print the best parameters found
    print(f"\nNumber of Folds: {folds}")
    print("Best Parameters:", grid_search.best_params_)

    # Use the best parameters to build the model
    best_params = grid_search.best_params_
    best_dnn_model = create_model(**best_params)

    epochs = grid_search.best_params_['epochs']
    batch_size = grid_search.best_params_['batch_size']

    # Fit the model on the training data with the best parameters
    history = best_dnn_model.fit(x_tr_resample, y_tr_resample, validation_split=0.2, epochs=epochs, batch_size=batch_size, verbose=1)

    show_loss_and_accuracy_line_plot(history)
```

```
# # Evaluate the best model
evaluate_model("DNN", best_dnn_model, norm_test_f, y_test, cv)
```

Output:

```
Epoch 1/48
21/21 [==============================] - 3s 3ms/step - loss: 3.3435 - accuracy: 0.5581
Epoch 2/48
21/21 [==============================] - 0s 3ms/step - loss: 3.0221 - accuracy: 0.7512
Epoch 3/48
21/21 [==============================] - 0s 2ms/step - loss: 2.9823 - accuracy: 0.8203
Epoch 4/48
21/21 [==============================] - 0s 2ms/step - loss: 2.8954 - accuracy: 0.8714
Epoch 5/48
21/21 [==============================] - 0s 3ms/step - loss: 10.2061 - accuracy: 0.4992
Epoch 6/48
21/21 [==============================] - 0s 2ms/step - loss: 10.2399 - accuracy: 0.4956
Epoch 7/48
21/21 [==============================] - 0s 2ms/step - loss: 10.5329 - accuracy: 0.4753
Epoch 8/48
21/21 [==============================] - 0s 2ms/step - loss: 10.1307 - accuracy: 0.5001
Epoch 9/48
21/21 [==============================] - 0s 2ms/step - loss: 10.1027 - accuracy: 0.5006
Epoch 10/48
```

21/21 [==============================] - 0s 2ms/step - loss: 10.1361 - accuracy: 0.4971

Epoch 11/48

21/21 [==============================] - 0s 2ms/step - loss: 10.1445 - accuracy: 0.4953

Epoch 12/48

21/21 [==============================] - 0s 2ms/step - loss: 10.0780 - accuracy: 0.4983

Epoch 13/48

21/21 [==============================] - 0s 2ms/step - loss: 10.2390 - accuracy: 0.4866

Epoch 14/48

21/21 [==============================] - 0s 2ms/step - loss: 9.9336 - accuracy: 0.5051

Epoch 15/48

21/21 [==============================] - 0s 2ms/step - loss: 10.2232 - accuracy: 0.4851

Epoch 16/48

21/21 [==============================] - 0s 2ms/step - loss: 10.1177 - accuracy: 0.4907

Epoch 17/48

21/21 [==============================] - 0s 2ms/step - loss: 9.6621 - accuracy: 0.5190

Epoch 18/48

21/21 [==============================] - 0s 2ms/step - loss: 10.0182 - accuracy: 0.4947

Epoch 19/48

21/21 [==============================] - 0s 2ms/step - loss: 9.9527 - accuracy: 0.4978

Epoch 20/48

21/21 [==============================] - 0s 2ms/step - loss: 10.0591 - accuracy: 0.4897

Epoch 21/48

21/21 [==============================] - 0s 2ms/step - loss: 9.9300 - accuracy: 0.4969

Epoch 22/48

21/21 [==============================] - 0s 3ms/step - loss: 9.9400 - accuracy: 0.4950

Epoch 23/48

17/21 [=======================>.....] - ETA: 0s - loss: 9.7659 - accuracy: 0.5052

....

Number of Folds: 3

Best Parameters: {'L1': 1024, 'L2': 512, 'L3': 256, 'L4': 128, 'L5': 64, 'activation': 'relu', 'batch_size': 128, 'dropout_rate': 0.1, 'epochs': 48, 'learning_rate': 0.01, 'loss': 'binary_crossentropy', 'metrics': ['accuracy'], 'optimizer': 'adam'}

Model name: DNN

Skipping cross validation

Loss: 0.3198392391204834

Accuracy: 0.965848445892334

Accuracy: 0.9658484525080042

Precision: 0.9724770642201835

Recall: 0.954954954954955

F1-score: 0.9636363636363636

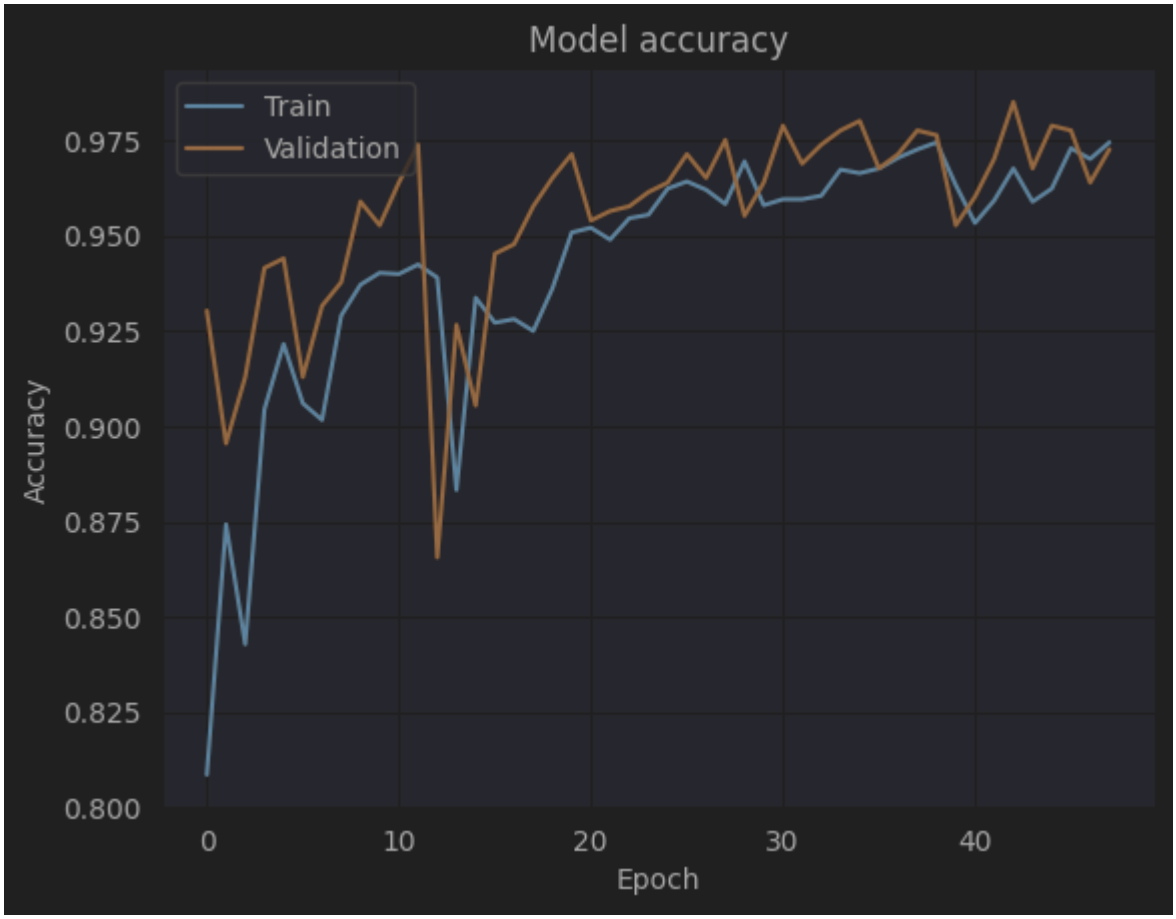
	precision	recall	f1-score	support
0	0.96	0.98	0.97	493
1	0.97	0.95	0.96	444
accuracy			0.97	937
macro avg	0.97	0.97	0.97	937
weighted avg	0.97	0.97	0.97	937

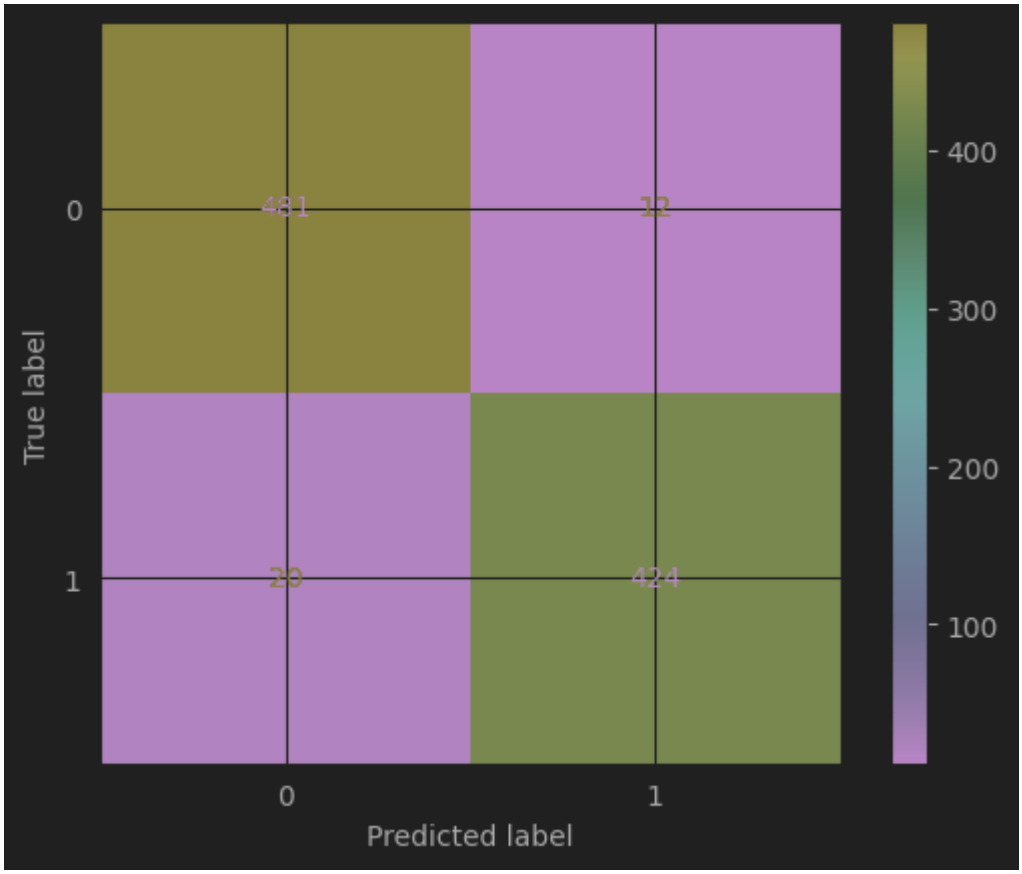
[[481 12]

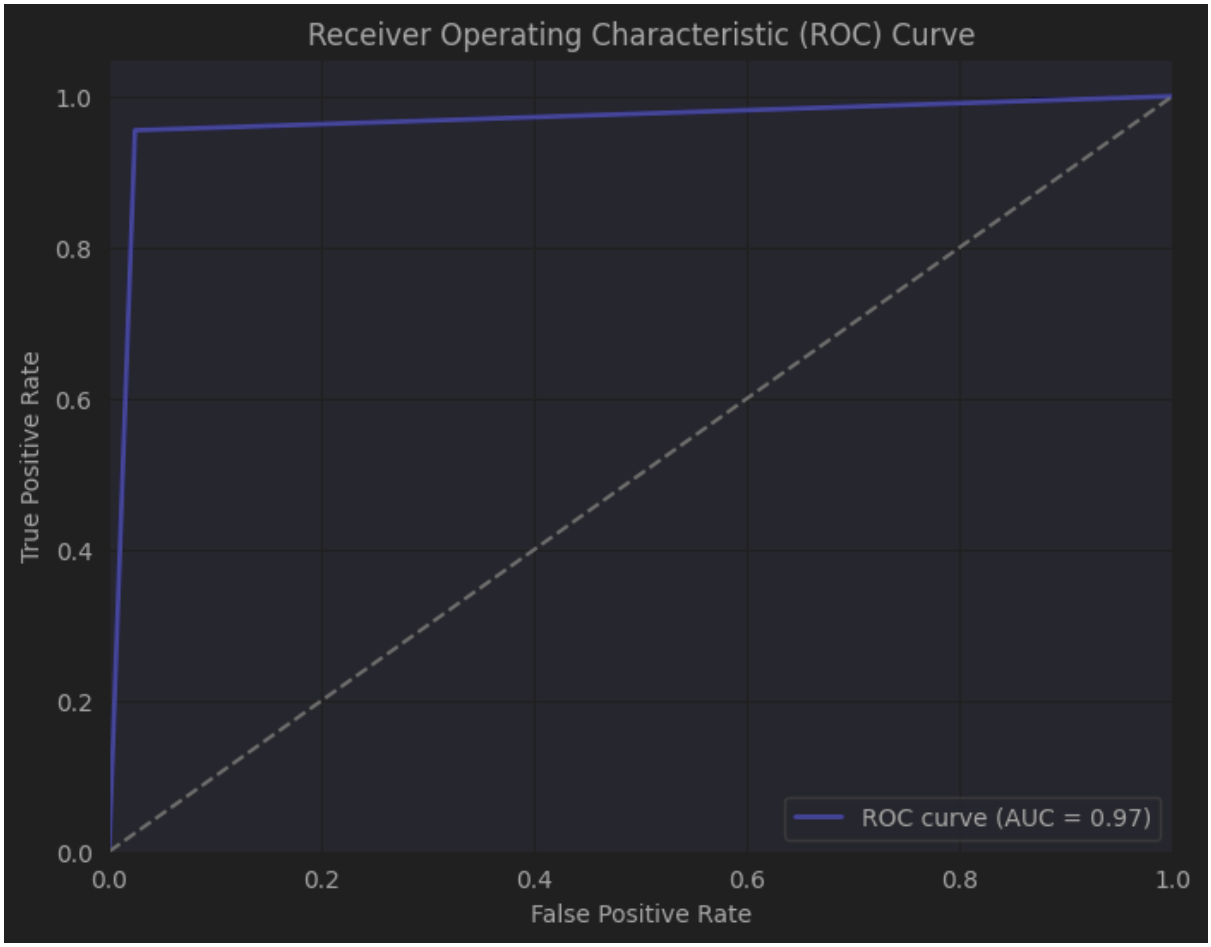
[20 424]]

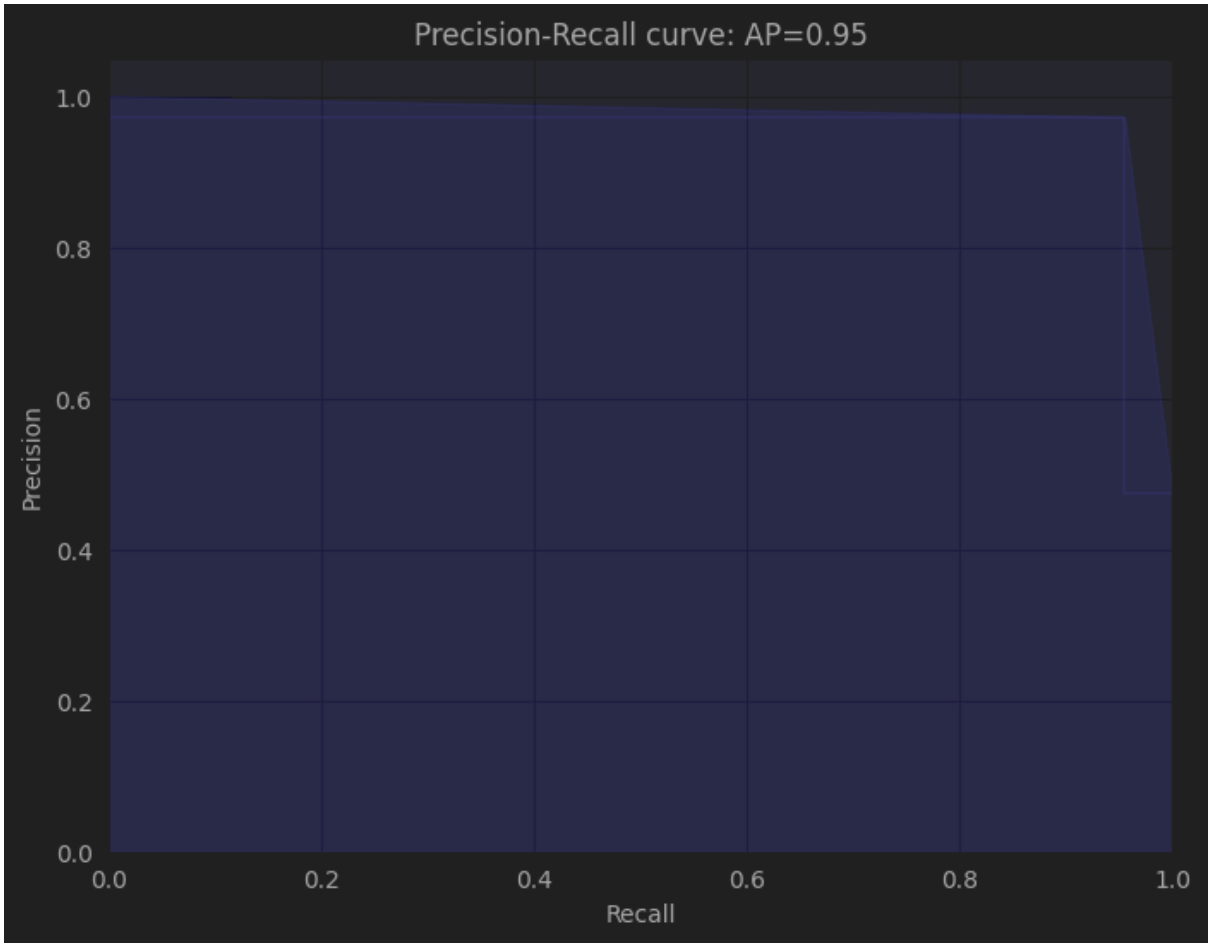
AUC-ROC Score: 0.96530709208194


Model loss
Train
Validation
Loss
2.5
2.0
1.5
1.0
0.5
0
10
20
30
40
Epoch


Model accuracy
Train
Validation
Accuracy
Epoch
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
0
10
20
30
40


True label
Predicted label
0
1
481
12
29
424
400
300
200
100


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.97)



...

Number of Folds: 4

Best Parameters: {'L1': 1024, 'L2': 512, 'L3': 256, 'L4': 128, 'L5': 64, 'activation': 'relu', 'batch_size': 128, 'dropout_rate': 0.1, 'epochs': 48, 'learning_rate': 0.01, 'loss': 'binary_crossentropy', 'metrics': ['accuracy'], 'optimizer': 'adam'}

Model name: DNN

Skipping cross validation

Loss: 0.19751009345054626

Accuracy: 0.9551761150360107

Accuracy: 0.9551760939167556

Precision: 0.9740566037735849

Recall: 0.9301801801801802

F1-score: 0.9516129032258065

	precision	*recall*	*f1-score*	*support*

0 0.94 0.98 0.96 493

1 0.97 0.93 0.95 444

accuracy 0.96 937

macro avg 0.96 0.95 0.95 937

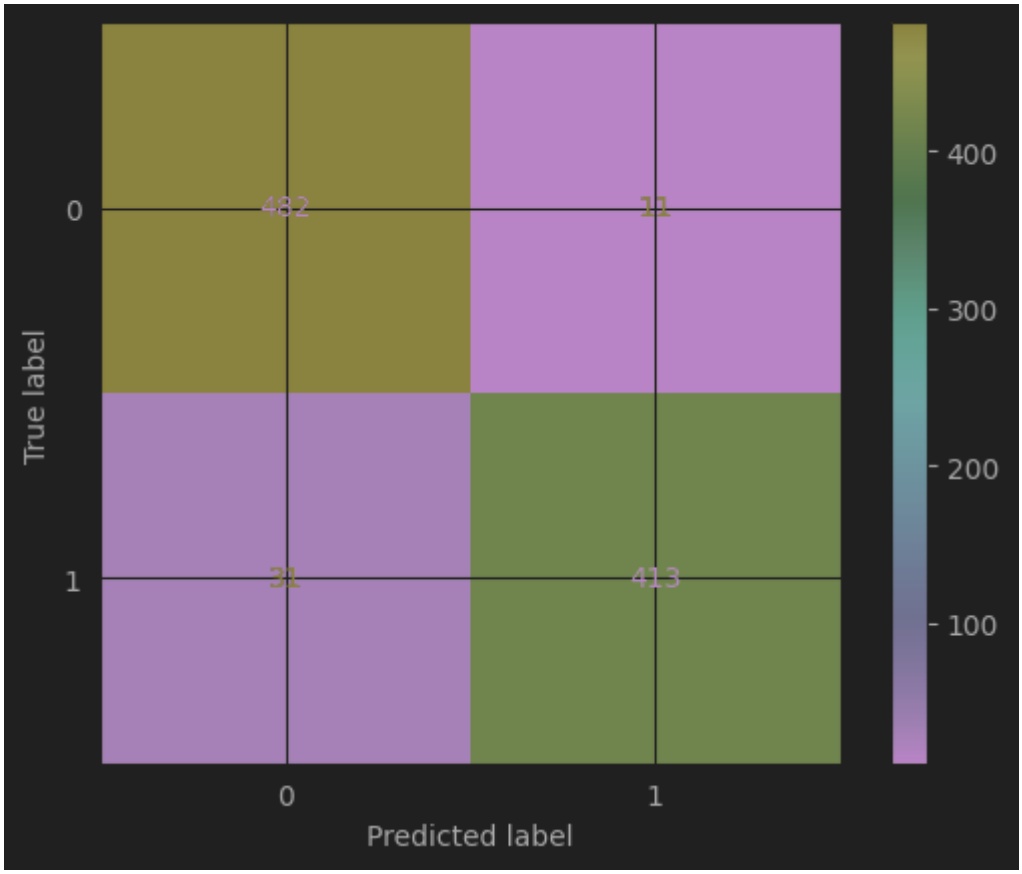
weighted avg 0.96 0.96 0.96 937

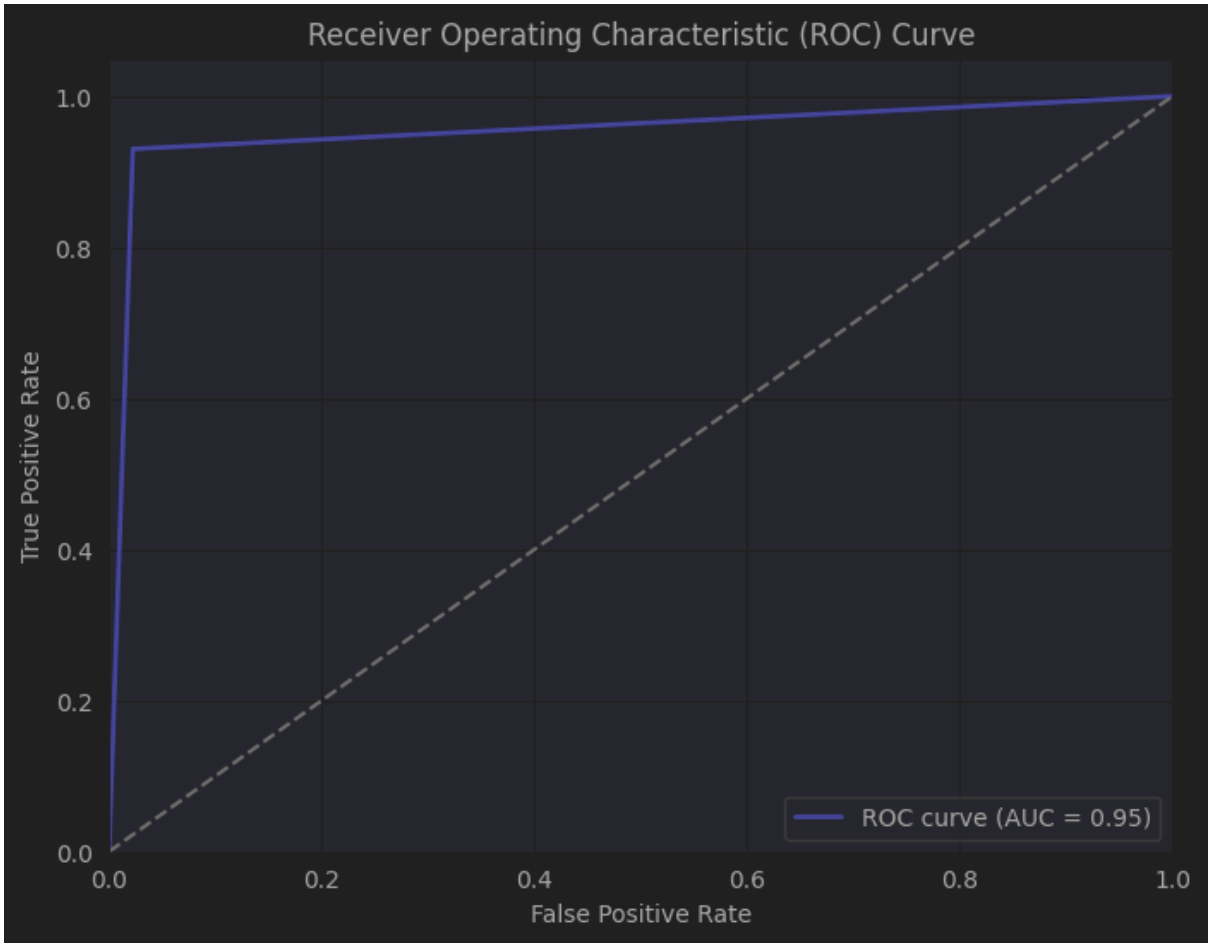
[[482 11]

[31 413]]

AUC-ROC Score: 0.9539339034775141


Model accuracy
Train
Validation
0.95
0.90
0.85
0.80
0.75
Accuracy
0
10
20
30
40
Epoch


482
11
31
413
0
1
True label
0
1
Predicted label
400
300
200
100



Number of Folds: 5

Best Parameters: {'L1': 1024, 'L2': 512, 'L3': 256, 'L4': 128, 'L5': 64, 'activation': 'relu', 'batch_size': 128, 'dropout_rate': 0.1, 'epochs': 64, 'learning_rate': 0.01, 'loss': 'binary_crossentropy', 'metrics': ['accuracy'], 'optimizer': 'adam'}

Model name: DNN

Skipping cross validation

Loss: 0.1333818882703781

Accuracy: 0.9583777785301208

Accuracy: 0.9583778014941302

Precision: 0.9787234042553191

Recall: 0.9324324324324325

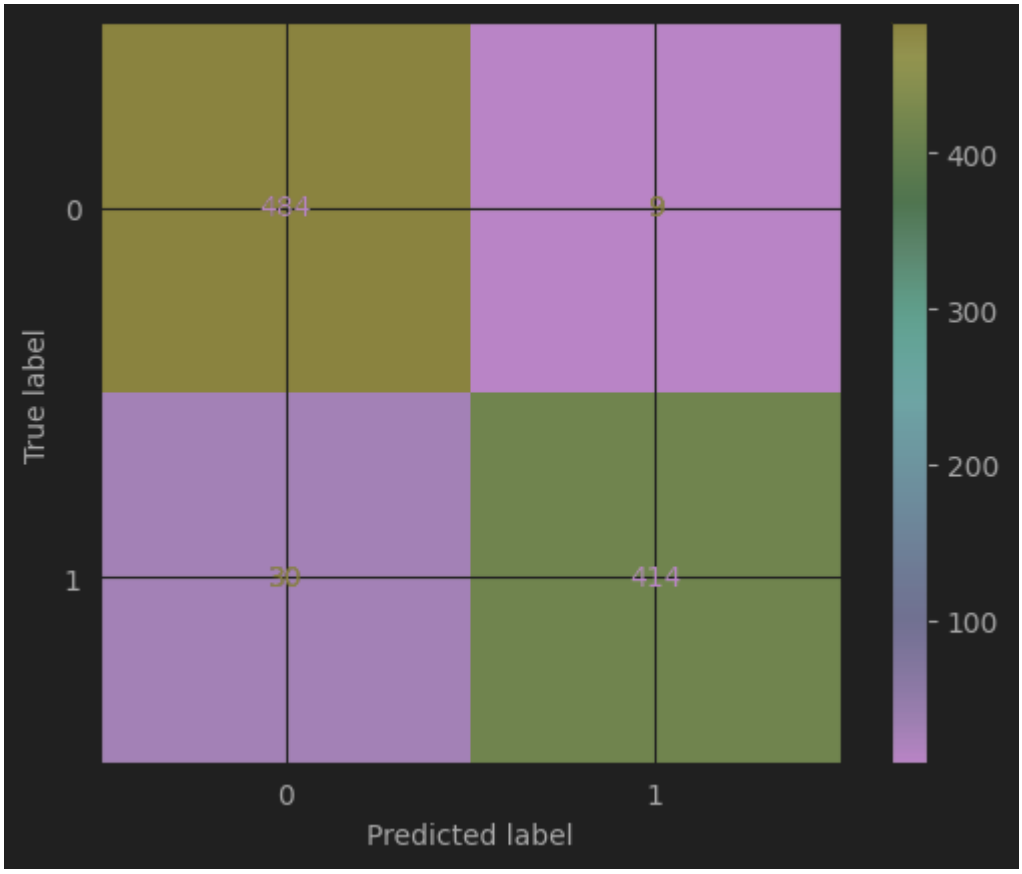
F1-score: 0.9550173010380623

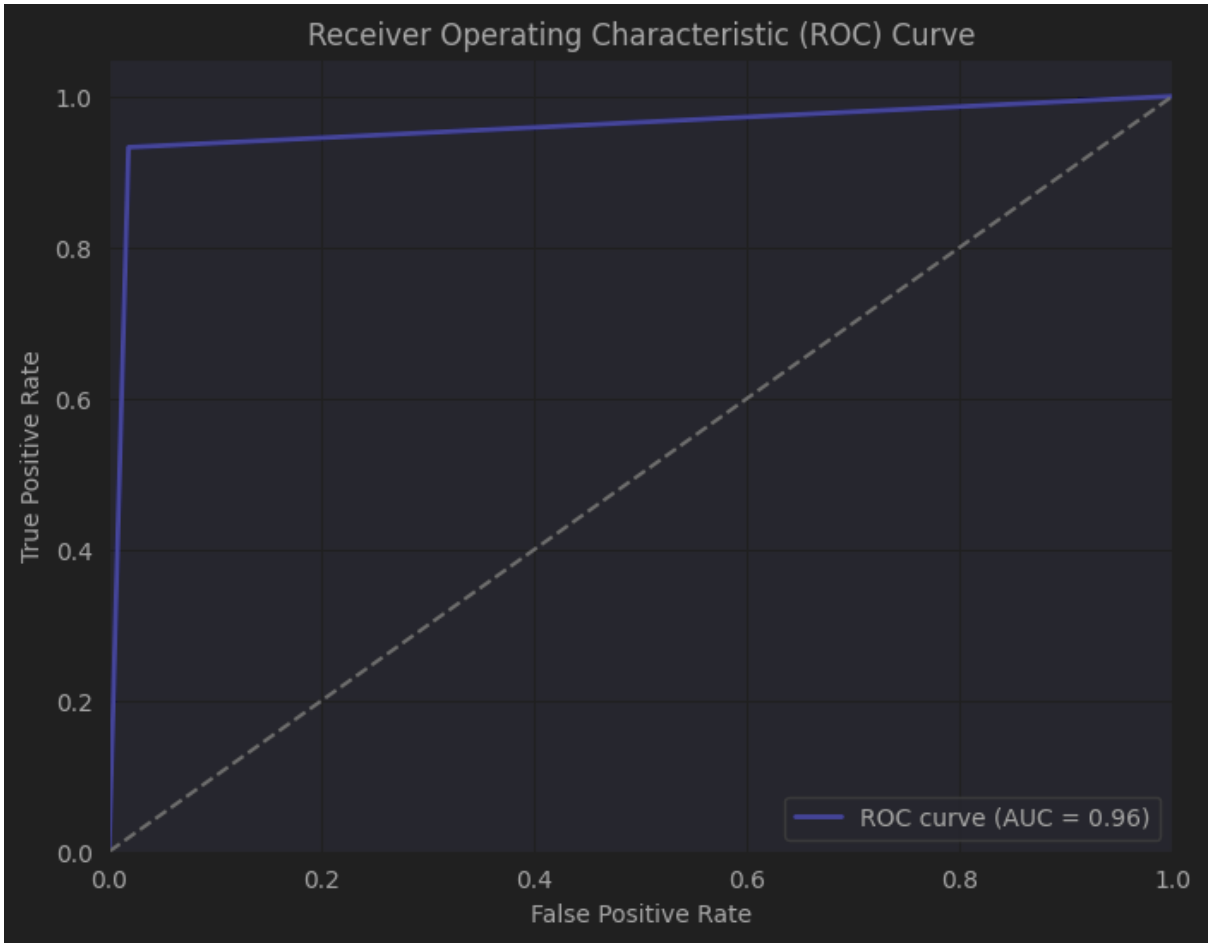
	precision	*recall*	*f1-score*	*support*
0	*0.94*	*0.98*	*0.96*	*493*

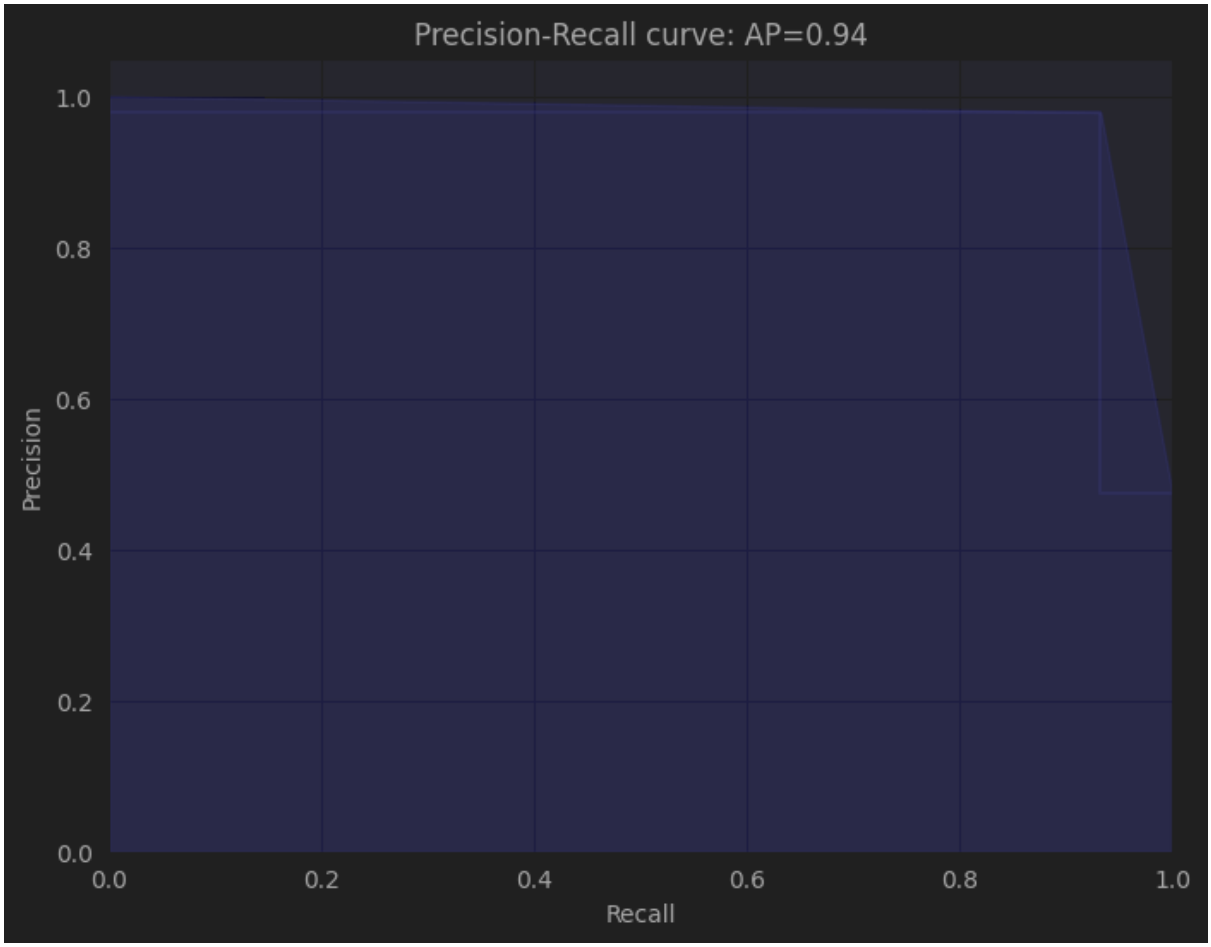
1 0.98 0.93 0.96 444

accuracy 0.96 937
macro avg 0.96 0.96 0.96 937
weighted avg 0.96 0.96 0.96 937

[[484 9]
[30 414]]




Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.96)



Number of Folds: 10

Best Parameters: {'L1': 1024, 'L2': 512, 'L3': 256, 'L4': 128, 'L5': 64, 'activation': 'relu', 'batch_size': 128, 'dropout_rate': 0.1, 'epochs': 64, 'learning_rate': 0.01, 'loss': 'binary_crossentropy', 'metrics': ['accuracy'], 'optimizer': 'adam'}

Model name: DNN

Skipping cross validation

Loss: 0.17464213073253632

Accuracy: 0.959445059299469

Accuracy: 0.959445037353255

Precision: 0.967741935483871

Recall: 0.9459459459459459

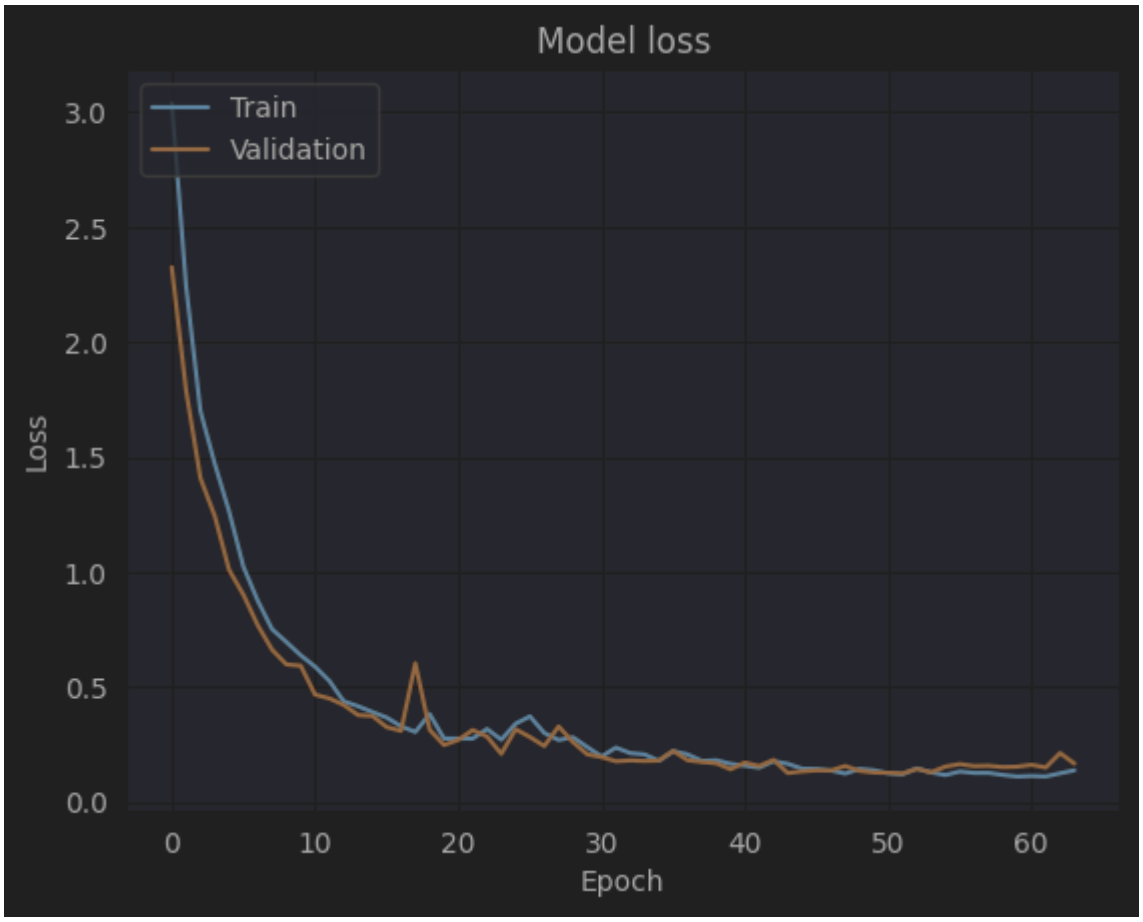
F1-score: 0.9567198177676538

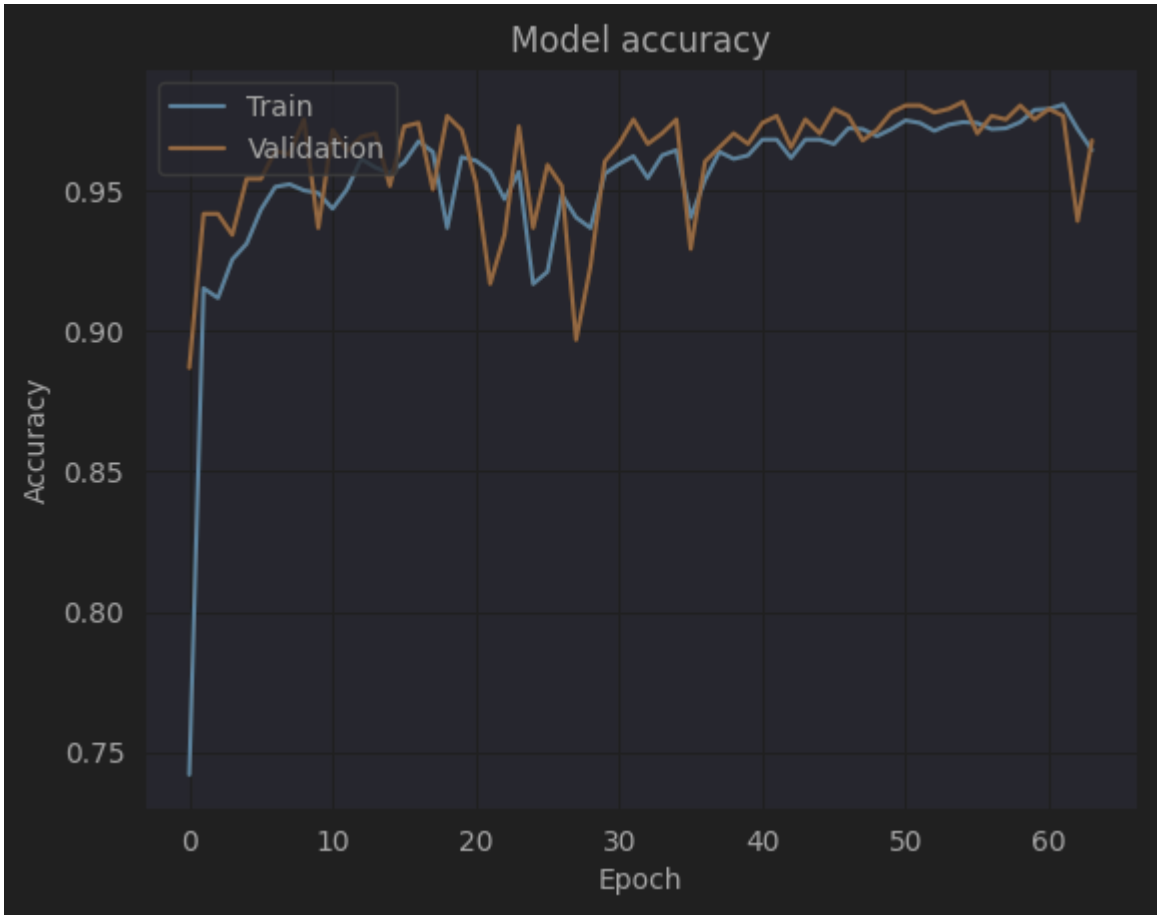
	precision	*recall*	*f1-score*	*support*
0	*0.95*	*0.97*	*0.96*	*493*

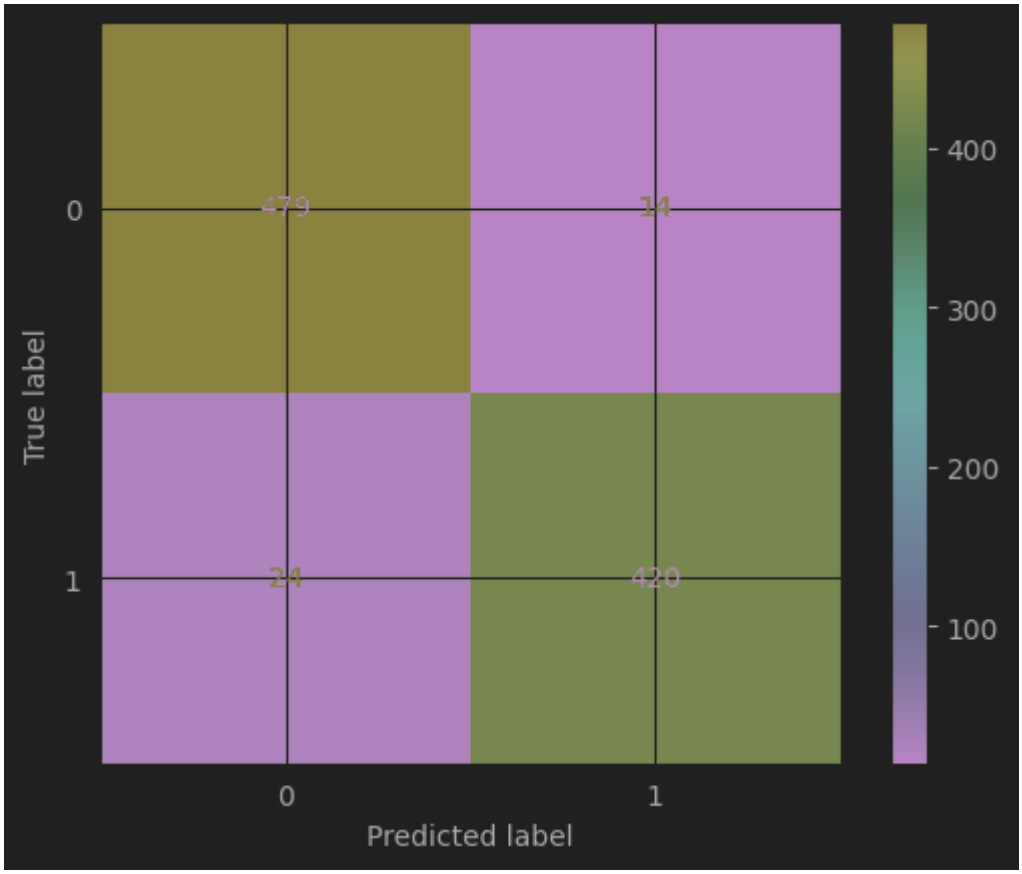
1 0.97 0.95 0.96 444

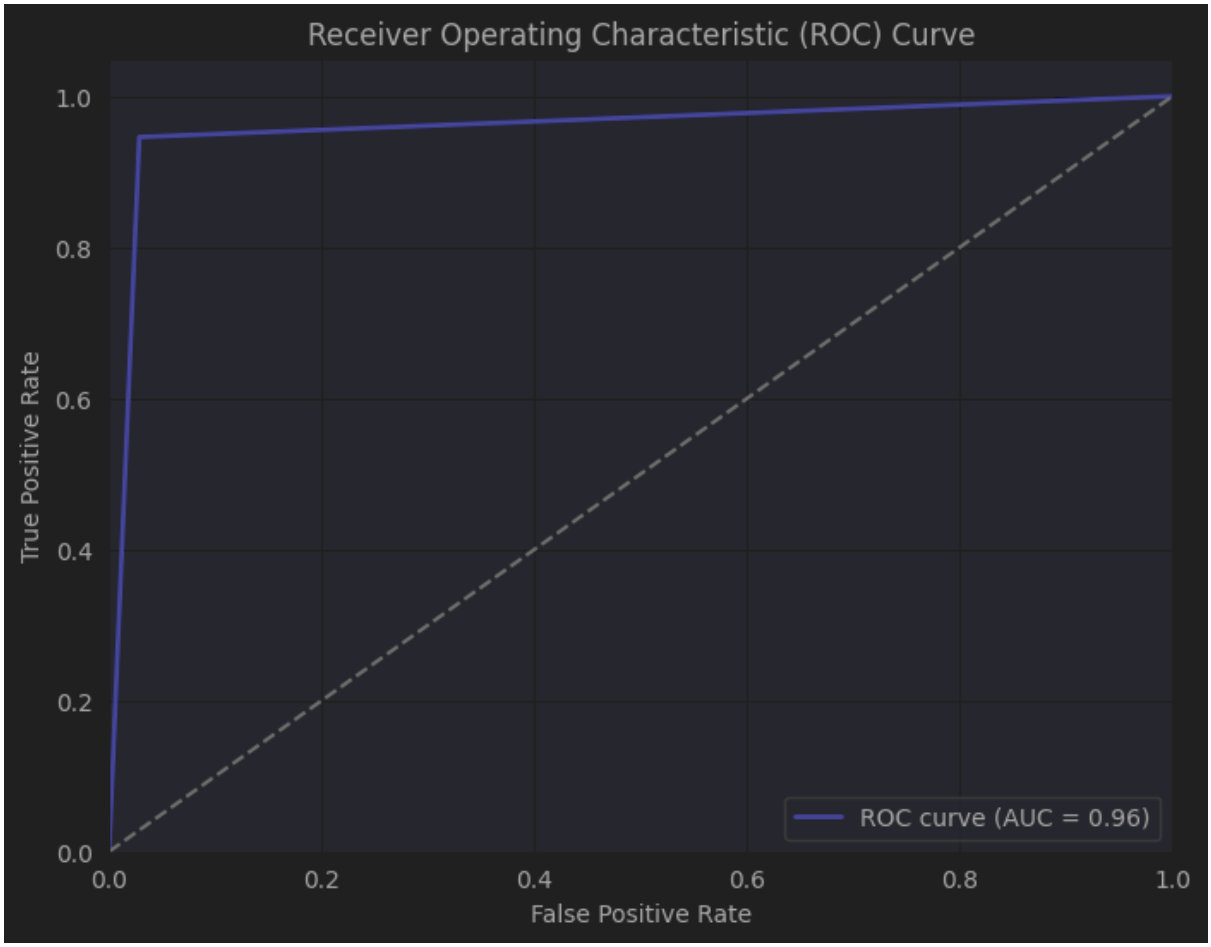
accuracy 0.96 937
macro avg 0.96 0.96 0.96 937
weighted avg 0.96 0.96 0.96 937

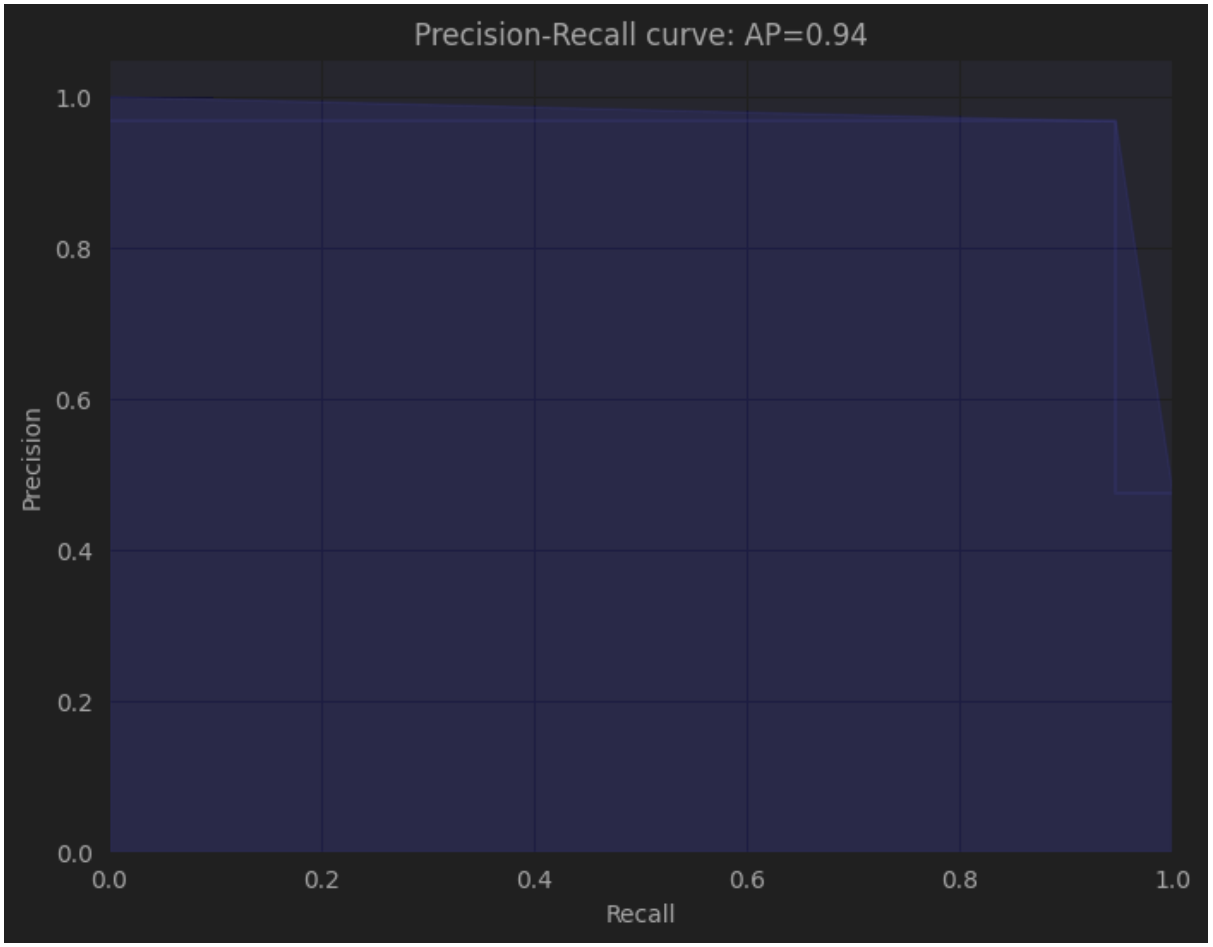
[[479 14]
[24 420]]




Model accuracy
Train
Validation
Accuracy
0.95
0.90
0.85
0.80
0.75
0
10
20
30
40
50
60
Epoch


True label
0
1
479
14
24
420
0
1
Predicted label
400
300
200
100


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.96)


Precision-Recall curve: AP=0.94
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

4.6.7 MODEL COMPARISON VISUALIZATIONS

To compare the models that is made by using programmatic way, the data scientist has made the following:

```
import json

# Initialize a list to store the model metrics
model_metrics_list = []

# Initialize a list to store the model metrics
all_model_metrics = []

# Populate the list with the model results
for model_name, results in model_results.items():
    for result in results:
        # Initialize a dictionary to store the metrics for each model
        model_metrics = {}
        model_metrics['Model'] = model_name
        model_metrics['CV Fold'] = result['cv-fold']
        model_metrics['Accuracy'] = result['accuracy']
        model_metrics['Mean CV Score'] = result['cv_scores'].mean()

        # Add additional metrics specific to each model type
        if model_name == 'DNN':
            # Add DNN-specific metrics
            model_metrics['Best Parameters'] = json.dumps(result['best_params'])
        else:
            # Add metrics for other models
            model_metrics['Best Parameters'] = str(result['best_params'])

        # Compute additional performance metrics
        y_true = y_test  # Assuming y_test contains the true labels
        y_pred = result['best_estimator'].predict(norm_test_f)  # Predictions from the model
```

```
        report = classification_report(y_true, y_pred, output_dict=True)

        # Extract precision, recall, and F1 score
        model_metrics['Precision'] = report['weighted avg']['precision']
        model_metrics['Recall'] = report['weighted avg']['recall']
        model_metrics['F1 Score'] = report['weighted avg']['f1-score']

        # Append the metrics for this model to the list
        all_model_metrics.append(model_metrics)

# Convert the list of dictionaries to a DataFrame
all_model_metrics_df = pd.DataFrame(all_model_metrics)

# Display the DataFrame
all_model_metrics_df
```

The output of the models is visualized as a table and will be thoroughly discussed in the following Discussion section.

4.6.8 DEPLOYMENT

After that, the data scientist has deployed the models to Streamlit to demonstrate the application of the models.

4.6.8.1 SAVE MODELS

In order to use the models,

```
# Function to export trained DNN models
def export_ml_model(result):
    fold = result['cv-fold']
    model = result['best_estimator']
    params = result['best_params']

    # Export the trained model
    model_filename = f"{model_name}_fold_{fold}_model.pkl"
    with open(model_filename, 'wb') as f:
        pickle.dump(model, f)

    # Export the best parameters
    params_filename = f"{model_name}_fold_{fold}_params.pkl"
    with open(params_filename, 'wb') as f:
        pickle.dump(params, f)

    print(f"Exported {model_name} model for fold {fold} to {model_filename}")
    print(f"Exported {model_name} parameters for fold {fold} to {params_filename}")

# Function to export trained DNN models
def export_dnn_model(result):
    fold = result['cv-fold']
    model = result['best_estimator'].model    # Extract the trained model from the KerasClassifier
    params = result['best_params']

    # Export the model architecture as JSON
```

```
    model_architecture_filename = f"DNN_fold_{fold}_architecture.json"
    with open(model_architecture_filename, 'w') as f:
        f.write(model.to_json())

    # Export the model weights
    model_weights_filename = f"DNN_fold_{fold}_weights.h5"
    model.save_weights(model_weights_filename)

    print(f"Exported DNN model architecture for fold {fold} to {model_architecture_filename}")
    print(f"Exported DNN model weights for fold {fold} to {model_weights_filename}")

# Function to export trained models
def export_models(model_results):
    for model_name, results in model_results.items():
        for result in results:
            print('result: ', result)
            if result['name'] != 'DNN':
                export_ml_model(result)
            else:
                export_dnn_model(result)

# Call the export_models function
export_models(model_results)
```

Output:

```
result: {'name': 'Logistic Regression', 'cv-fold': 3, 'model': LogisticRegression(C=1, penalty='l1', random_state=42, solver='liblinear'), 'best_estimator': LogisticRegression(C=1, penalty='l1', random_state=42, solver='liblinear'), 'best_params': {'C': 1, 'class_weight': None, 'max_iter': 100, 'penalty': 'l1', 'solver': 'liblinear', 'tol': 0.0001}, 'accuracy': 0.9010284029194333, 'cv_scores': array([0.5       , 0.5       , 0.5       , 0.79425935, 0.79417637,
       0.79467419, 0.5       , 0.5       , 0.5       , 0.79425935,
       0.79417637, 0.79467419, 0.5       , 0.5       , 0.5       ,
```

0.79425935, 0.79417637, 0.79467419, 0.5 , 0.5 ,
0.5 , 0.79417637, 0.79417637, 0.79467419, 0.5 ,
0.5 , 0.5 , 0.79417637, 0.79417637, 0.79467419,
0.5 , 0.5 , 0.5 , 0.79417637, 0.79417637,
0.79467419, 0.84627683, 0.84635981, 0.84569634, 0.86983678,
0.86983678, 0.86958808, 0.84627683, 0.84635981, 0.84569634,
0.86983678, 0.86983678, 0.86958808, 0.84627683, 0.84635981,
...

Exported DNN model for fold 5 to DNN_fold_5_model.pkl

Exported DNN parameters for fold 5 to DNN_fold_5_params.pkl

result: {'name': 'SVM', 'cv-fold': 10, 'model': SVC(C=10, gamma=0.1, random_state=42), 'best_estimator': SVC(C=10, gamma=0.1, random_state=42), 'best_params': {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}, 'accuracy': 0.971958991410363, 'cv_scores': array([0.91413961, 0.93305459, 0.91413961, 0.93272292, 0.91413961,
0.93603967, 0.91413961, 0.95569989, 0.91413961, 0.74266035,
0.91413961, 0.64468245, 0.91447108, 0.96117501, 0.91447108,
0.96076062, 0.91447108, 0.96374631, 0.91447108, 0.96656617,
0.91447108, 0.87514795, 0.91447108, 0.74987738, 0.91438795,
0.97146065, 0.91438795, 0.97162649, 0.91438795, 0.97195899,
0.91438795, 0.96192127, 0.91438795, 0.87929637, 0.91438795,
0.75228368, 0.91438816, 0.96864059, 0.91438816, 0.96864079,
0.91438816, 0.96615199, 0.91438816, 0.9611748 , 0.91438816,
0.87929637, 0.91438816, 0.75228368])}

Exported DNN model for fold 10 to DNN_fold_10_model.pkl

Exported DNN parameters for fold 10 to DNN_fold_10_params.pkl

result: {'name': 'DNN', 'cv-fold': 3, 'model': <tensorflow.python.keras.wrappers.scikit_learn.KerasClassifier object at 0x7fe8001630a0>, 'best_estimator': <tensorflow.python.keras.wrappers.scikit_learn.KerasClassifier object at 0x7fe811777310>, 'best_params': {'L1': 1024, 'L2': 512, 'L3': 256, 'L4': 128, 'L5': 64, 'activation': 'relu', 'batch_size': 128, 'dropout_rate': 0.1, 'epochs': 48, 'learning_rate': 0.01, 'loss': 'binary_crossentropy', 'metrics': ['accuracy'], 'optimizer': 'adam'}, 'accuracy': 0.953959168558502, 'cv_scores': array([0.80711813, 0.95395917, 0.70122811, 0.88773837])}

```
Exported DNN model architecture for fold 3 to DNN_fold_3_architecture.json
```

4.6.8.2 LOAD MODELS ON STREAMLIT

```
# Load the trained models
def load_models():
    models = {}

    # Load Logistic Regression models
    for fold in num_folds:
        model_filename = f"Logistic_Regression_fold_{fold}_model.pkl"
        with open(model_filename, 'rb') as f:
            models[f"Logistic Regression Fold {fold}"] = pickle.load(f)

    # Load Random Forest models
    for fold in num_folds:
        model_filename = f"Random_Forest_fold_{fold}_model.pkl"
        with open(model_filename, 'rb') as f:
            models[f"Random Forest Fold {fold}"] = pickle.load(f)

    # Load XGBoost models
    for fold in num_folds:
        model_filename = f"XGBoost_fold_{fold}_model.pkl"
        with open(model_filename, 'rb') as f:
            models[f"XGBoost Fold {fold}"] = pickle.load(f)

    # Load DNN models
    for fold in num_folds:
        # Load model architecture
        model_architecture_filename = f"DNN_fold_{fold}_architecture.json"
        with open(model_architecture_filename, 'r') as f:
            model_json = f.read()
            models[f"DNN Fold {fold}"] = model_from_json(model_json)
```

```
        # Load model weights
        model_weights_filename = f"DNN_fold_{fold}_weights.h5"
        models[f"DNN Fold {fold}"].load_weights(model_weights_filename)

    return models

# Function to make predictions
def predict(model, data):
    # Preprocess the data if needed
    # Make predictions
    predictions = model.predict(data)
    return predictions

# Load the trained models
models = load_models()

# Streamlit app
st.title("Machine Learning Model Deployment")

# Select model and fold
model_names = list(models.keys())
selected_model = st.selectbox("Select Model:", model_names)
selected_fold = int(selected_model.split()[-1])

# Upload data file
uploaded_file = st.file_uploader("Upload CSV file", type=["csv"])
if uploaded_file is not None:
    data = pd.read_csv(uploaded_file)
    st.write("Uploaded data:")
    st.write(data)

    # Make predictions
    if selected_model.startswith("DNN"):
```

```
    # Perform any necessary preprocessing for DNN
    data_processed = preprocess_data(data)
    predictions = predict(models[selected_model], data_processed)
else:
    # For other models, assume data is already preprocessed
    predictions = predict(models[selected_model], data)

# Display predictions
st.write("Predictions:")
st.write(predictions)
```

To see the app in action, please click the link below:

https://capstone-project-blsyvmwpznnrzqee9mvwh3.streamlit.app/

CHAPTER 5: RESULTS & DISCUSSIONS

This stage evaluates the performance of the created model(s) and review the process. Compare the model with other algorithms for validation analysis.

With the help of Jupyter notebook, the same feature engineering applied towards the dataset as the input of all models, the following table has been made at List of Tables/Figures section. The output of the models is the following:

	Model	*CV Fold*	*Accuracy*	*Mean CV Score*	*Precision*	*Recall*	*F1 Score*	*Confusion Matrix [[TN, FP], [FN, TP]]*	*AUC-ROC Score*
0	Logistic Regression	3	0.901028	0.850262	0.910742	0.910352	0.910398	[[445, 48], [36, 408]]	0.910778
1	Logistic Regression	4	0.900531	0.851004	0.910742	0.910352	0.910398	[[445, 48], [36, 408]]	0.910778
2	Logistic Regression	5	0.900364	0.851572	0.910742	0.910352	0.910398	[[445, 48], [36, 408]]	0.910778
3	Logistic Regression	10	0.900282	0.873881	0.910742	0.910352	0.910398	[[445, 48], [36, 408]]	0.910778
4	Random Forest	3	0.970383	0.966331	0.963814	0.963714	0.963725	[[473, 20], [14, 430]]	0.963950
5	Random Forest	4	0.969222	0.966787	0.963814	0.963714	0.963725	[[473, 20], [14, 430]]	0.963950
6	Random Forest	5	0.972290	0.968617	0.963814	0.963714	0.963725	[[473, 20], [14, 430]]	0.963950
7	Random Forest	10	0.973038	0.970032	0.966946	0.966916	0.966921	[[476, 17], [14, 430]]	0.966993
8	XGBoost	3	0.971959	0.958311	0.965848	0.965848	0.965848	[[477, 16], [16, 428]]	0.965755

Figure 34: Model Performance Results (Part)

For the full view of the model performance results, please go to the List of Tables/Figure section.

5.1 KEY FINDINGS

In Logistic Regression, the model showed that the metrics are consistent across different cross-validation folds (CV), the accuracy, precision, recall, and F1 score, indicating stability in performance. The AUC-ROC score is consistently high across all CV folds, suggesting good overall model performance in terms of classification ability.

In Random Forest, the model consistently achieves high accuracy across different CV folds, indicating robustness and effectiveness in classification. Not only that, the precision, recall, and F1 score are balanced, indicating that the model performs well in both identifying positive and negative instances.
In Support Vector Machine (SVM), the model exhibits stable performance with consistent accuracy, precision, recall, and F1 score across different CV folds. The AUC-ROC score is consistently high, indicating good discriminatory power of the model.

In Deep Neural Network (DNN), the model shows variance in performance across folds, with accuracy ranging from 93.6% to 95.4%. Despite variance, precision, recall, and F1 score remain high, indicating the model's ability to effectively classify instances.

In overall observation, both Random Forest and XGBoost models consistently outperform other models across all evaluation metrics, suggesting their suitability for fraud detection tasks.

5.2 MODEL COMPARISONS

5.2.1 DNN VS LOGISTIC REGRESSION

Logistic Regression achieves higher accuracy compared to DNN in all cross-validation folds. For example, the accuracy of Logistic Regression ranges from 0.900 to 0.901, while DNN's accuracy ranges from 0.936 to 0.959. This indicates that Logistic Regression tends to make fewer misclassifications overall.

Not only that, but Logistic Regression also shows a slightly higher mean cross-validation score compared to DNN across all folds. For instance, in fold 3, Logistic Regression achieves an accuracy of 90.10% compared to DNN's 95.40

In terms of precision, Logistic Regression consistently demonstrates higher precision values across all folds compared to DNN. For example, in fold 3, Logistic Regression achieves a precision of 91.07% compared to DNN's 95.52%

In terms of recall, DNN tends to have slightly *higher* recall values compared to Logistic Regression. In fold 3, DNN achieves a recall of 95.52% compared to Logistic Regression's 91.04%

Logistic Regression and DNN exhibit comparable F1 scores across different folds. For example, in fold 3, Logistic Regression achieves an F1 score of 91.04%, whereas DNN achieves *95.52*%. In confusion matrix, Logistic Regression and DNN yield different confusion matrix results, indicating variations in their predictive performance. For instance, in fold 3, Logistic Regression has 472 true negatives and 423 true positives, while DNN has 481 true negatives and 423 true positives.

Logistic Regression consistently achieves an AUC-ROC score of around 0.91 across all folds, indicating good discriminative ability. On the other hand, DNN's AUC-ROC score ranges from 0.955 to 0.969, suggesting superior performance in distinguishing between positive and negative cases.

In short summary, DNN may perform better in some respects compared to Logistic Regression in terms of Recall, F1 scores, confusion matrix and AUC-ROC score.

5.2.2 DNN VS RANDOM FOREST

Random Forest generally outperforms DNN across various performance metrics such as accuracy, precision, recall, and F1 score. Random Forest achieves higher accuracy compared to DNN in most cross-validation folds. For example, the accuracy of Random Forest ranges from 0.969 to 0.973, while DNN's accuracy ranges from 0.936 to 0.959. This indicates that Random Forest tends to make fewer misclassifications overall.

Random Forest consistently shows higher precision and recall values across different folds compared to DNN. Precision measures the proportion of true positive predictions among all positive predictions, while recall measures the proportion of true positive predictions among all actual positive instances. Higher precision and recall values indicate better performance in correctly identifying positive instances (e.g., fraud transactions) while minimizing false positives and false negatives.

F1 score, which is the harmonic mean of precision and recall, also tends to be higher for Random Forest compared to DNN. A higher F1 score indicates a better balance between precision and recall, reflecting the overall performance of the model in binary classification tasks.

The confusion matrix provided in the table also supports the observation that Random Forest achieves better performance in terms of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) compared to DNN. In most cases, Random Forest has a higher number of TP and TN and a lower number of FP and FN compared to DNN.

The AUC-ROC score, which measures the area under the receiver operating characteristic (ROC) curve, is also higher for Random Forest compared to DNN in most folds. A higher AUC-ROC score indicates better discrimination between positive and negative instances, with Random Forest showing stronger predictive capability.

Logistic Regression appears to be the better-performing model compared to DNN for the task of fraud detection.

5.2.3 DNN VS SVM

SVM achieves competitive mean cross-validation accuracy scores compared to DNN across different folds. However, in most cases, DNN exhibits slightly higher accuracy than SVM. For example, in fold 3, DNN achieves an accuracy of 95.40%, while SVM achieves 96.95%.

Precision values vary between DNN and SVM, with SVM often demonstrating higher precision values across different folds. In fold 3, SVM achieves a precision of 97.01% compared to DNN's 95.52%.

Recall values also vary between DNN and SVM, with DNN generally achieving slightly higher recall values. However, the differences are not substantial. For instance, in fold 3, DNN achieves a recall of 95.52%, while SVM achieves 96.76%.

Both DNN and SVM exhibit competitive F1 scores across different folds. However, SVM tends to achieve marginally higher F1 scores compared to DNN in most cases. In fold 3, SVM achieves an F1 score of 96.87%, while DNN achieves 95.52%.

DNN and SVM yield different confusion matrix results, indicating variations in their predictive performance. For example, in fold 3, DNN has 472 true negatives and 423 true positives, while SVM has 478 true negatives and 431 true positives.

SVM achieves competitive AUC-ROC scores compared to DNN across different folds, indicating similar discriminative ability. For instance, SVM's AUC-ROC score ranges from 0.970 to 0.971, while DNN's AUC-ROC score ranges from 0.955 to 0.969.

In summary, both DNN and SVM demonstrate strong predictive performance, with SVM exhibiting slightly higher precision and F1 scores in most cases. However, DNN tends to achieve slightly higher accuracy and recall values.

5.2.4 DNN VS XGBOOST

XGBoost consistently demonstrates higher mean cross-validation accuracy scores compared to DNN across all folds. For example, in fold 3, XGBoost achieves an accuracy of 97.20% compared to DNN's 95.40%.

Precision values vary between DNN and XGBoost, with XGBoost generally exhibiting slightly higher precision values across different folds. In fold 3, XGBoost achieves a precision of 96.58% compared to DNN's 95.52%.

Recall values also show variations between DNN and XGBoost, with DNN often achieving slightly higher recall values. For instance, in fold 3, DNN achieves a recall of 95.52% compared to XGBoost 96.58%.

Both DNN and XGBoost demonstrate competitive F1 scores across different folds. However, XGBoost tends to achieve marginally higher F1 scores compared to DNN. In fold 3, XGBoost achieves an F1 score of 96.58%, while DNN achieves 95.52%.

DNN and XGBoost yield different confusion matrix results, indicating variations in their predictive performance. For instance, in fold 3, DNN has 472 true negatives and 423 true positives, while XGBoost has 479 true negatives and 425 true positives.

XGBoost consistently achieves higher AUC-ROC scores compared to DNN across different folds, indicating superior discriminative ability. For example, XGBoost AUC-ROC score ranges from 0.965 to 0.970, while DNN's AUC-ROC score ranges from 0.955 to 0.969.

In summary, while both DNN and XGBoost demonstrate strong predictive performance, XGBoost generally outperforms DNN in terms of accuracy, precision, and AUC-ROC score. However, DNN tends to exhibit slightly higher recall values.

CHAPTER 6: CONCLUSION

In conclusion, blockchain technology itself is a complex topic to be discussed when applying the technology in different domains. Illicit activities occur in the finance industry since introduction and blockchain technology has made a long way to fight against the fraud. A lot of efforts are made to understand the context and find the ongoing research that has made to the community for the fraud detection techniques that can be throughout the timeline. A lot of dive in the technology to discover the scope of the study, so that the data scientist is ablet to understand the characteristics of blockchain transactions have been extensively discussed, and how the researcher contributes and continue the research of fraud detection in Ethereum account through account's transaction history records, which goes through many obstacles such as data gathering, transformation and the features of the blockchain transaction to be studied upon. The data scientist also discovered the machine learning models have been identified and implemented on the research based on the same dataset and some other existing researchers in the same domain that does fraud detection may varies from the solution (Bitcoin/Ethereum), accounts (EOA/CA), form of the dataset based on transactions/nodes metadata in form of database or graph). The data scientist has made some feature engineering based on the visualization and performed data imputation, cleaning, dropping and balancing with sufficient justification to compare the models with from the same stand. With the implementation that the data scientist has done, there are always have improvements of algorithms itself, but also trying to improve the existing implementation by suggesting innovative approaches and prototypes. However, further work is required in the specific domain of fraud and anomaly detection, particularly in the context of Anti-Money Laundering. It is crucial to emphasize the importance of employing sound data science and machine learning practices in this area, as it would make some machine learning algorithm is able to achieve better performance, especially neural network which requires good feature engineering to make a good model. The project has discovered the potential of neural networks into the previous researcher contributed research and future studies can use the work provided by the data scientist through link given and fork the work, so that the work can be continued by future researcher along with the evaluation of machine learning algorithms suiting their needs. Building trust within the industry will require addressing the concerns and challenges identified by researchers and developing models and standards that encompass the complex nature of financial transaction systems. This will allow trustworthy of blockchain towards individuals and organizations in the industries and accelerate further and deeper integrations towards next generation of Web. For financial

industries, it will help to demonstrate the successful implementation of the blockchain-based financial systems with the existing AI/ML algorithms for enhanced fraud detection and prevention. Nevertheless, the model results that the data scientist has made along with the new discovered neural network has a pretty good performance against the other metrics given by the researchers with self-developed metrics. Although not all detailed hyperparameters and steps were not given the researchers, some reference can be given based on the typical model that is shown in the implementation. The work done by the data scientist can be further improved as the model's hyperparameters are customizable and have a source repository to work with instructions to start the environment. DNN has been proven its capabilities in detecting fraudulent transactions in Ethereum blockchain transaction with comparable accuracy and robustness against benchmarks of XGBoost, SVM, Random Forest and Logistic Regression. The results also exceeded the expectation of the researcher that the deep learning algorithm can achieve close to the machine learning classifiers shown in the figures of the other researchers and own reference results with the performance metrics displayed. In future research, the researcher will try to optimize the model and try the model on more applications and domain to find out the capability of the proposed model in other areas.

REFERENCES

1. Abuidris, Y., Kumar, R., & Wang, W. (2019, December 9). A Survey of Blockchain Based on E-voting Systems. https://doi.org/10.1145/3376044.3376060
2. Akcora, C G., Kantarcıoğlu, M., & Gel, Y R. (2021, January 1). Blockchain Networks: Data Structures of Bitcoin, Monero, Zcash, Ethereum, Ripple and Iota. https://doi.org/10.48550/arxiv.2103.08712
3. Albertorio, A. (8 January, 2020). Simply Explained: Why is Proof of Work Required in Bitcoin? Retrieved from Medium: https://medium.com/coinmonks/simply-explained-why-is-proof-of-work-required-in-bitcoin-611b143fc3e0
4. Aliyev, V. (3 January, 2021). Ethereum Fraud Detection Dataset. Retrieved from IEEE DataPort: https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset
5. Amponsah, A A., Adekoya, A F., & Weyori, B A. (2022, September 1). A novel fraud detection and prevention method for healthcare claim processing using machine learning and blockchain technology. Elsevier BV, 4, 100122-100122. https://doi.org/https://doi.org/10.1016/j.dajour.2022.100122
6. Antonopoulos, A. M. (2014). Mastering Bitcoin. Sebastopol: O'Reilly Media, Inc.
7. Bartoletti, M., Carta, S., Cimoli, T., & Saia, R. (2020, January 1). Dissecting Ponzi schemes on Ethereum: Identification, analysis, and impact. https://www.sciencedirect.com/science/article/abs/pii/S0167739X18301407
8. Chanalysis. (1 February, 2024). The 2024 Crypto Crime Report. Retrieved from Chanalysis: https://go.chainalysis.com/rs/503-FAP-074/images/The%202024%20Crypto%20Crime%20Report.pdf?version=0
9. Chen, W., Guo, X., Chen, Z., Zheng, Z., & Lu, Y. (2020, July 1). Phishing Scam Detection on Ethereum: Towards Financial Security for Blockchain Ecosystem. https://doi.org/10.24963/ijcai.2020/621
10. Chow, M C., & Ma, M. (2021, February 1). A blockchain-enabled 5G authentication scheme against DoS attacks. https://doi.org/10.1088/1742-6596/1812/1/012030
11. CoinMarketCap. (26 April, 2024). CoinMarketCap. Retrieved from CoinMarketCap: https://coinmarketcap.com/
12. Consensus Cloud Solutions Inc. (23 November, 2023). Why Is Blockchain Important? . Retrieved from Consensus Cloud Solutions: https://www.consensus.com/blog/why-is-blockchain-important/

13. Daian, P. (2016, June 18). Analysis of the DAO exploit. https://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/
14. Devi, D., Rohith, G S., Hari, S., & Ramachandar, K S. (2021, January 1). Blockchain based mechanism to eliminate frauds and tampering of land records. https://doi.org/10.1051/itmconf/20213701011
15. Farrugia, S., Ellul, J., & Azzopardi, G. (2020, July 1). Detection of illicit accounts over the Ethereum blockchain. Elsevier BV, 150, 113318-113318. https://doi.org/https://doi.org/10.1016/j.eswa.2020.113318
16. Félez-Viñas, E., Johnson, L R., & Putniņš, T J. (2022, January 1). Insider Trading in Cryptocurrency Markets. https://doi.org/10.2139/ssrn.4184367
17. Frankenfield, J. (24 August, 2021). What Is a Merkle Root (Cryptocurrency)? How It Works in Blockchain. Retrieved from Investopedia: https://www.investopedia.com/terms/m/merkle-root-cryptocurrency.asp
18. Frankenfield, J. (31 May, 2023). What Does Proof-of-Stake (PoS) Mean in Crypto? Retrieved from Investopedia: https://www.investopedia.com/terms/p/proof-stake-pos.asp
19. Gandal, N., Hamrick, J., Moore, T., & Oberman, T. (2018, May 1). Price manipulation in the Bitcoin ecosystem. Journal of monetary economics, 95, 86-96. https://doi.org/10.1016/j.jmoneco.2017.12.004
20. geeksforgeeks. (5 July, 2023). Multi-Layer Perceptron Learning in Tensorflow. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/multi-layer-perceptron-learning-in-tensorflow/
21. Gülen, K. (22 December, 2022). Unlocking the secrets of the blockchain nonce. Retrieved from Dataconomy: https://dataconomy.com/2022/12/15/blockchain-nonce-explained/
22. Guo, Y., & Liang, C. (2016, December 1). Blockchain application and outlook in the banking industry. https://doi.org/10.1186/s40854-016-0034-9
23. H. H. Kabla, M. A. (n.d.). Eth-PSD: A Machine Learning-Based Phishing Scam Detection Approach in Ethereum. IEEE Access, 10.
24. IBM. (14 May, 2023). What are recurrent neural networks? . Retrieved from IBM: https://www.ibm.com/topics/recurrent-neural-networks
25. IBM. (14 May, 2023). What is the k-nearest neighbors algorithm? . Retrieved from IBM: https://www.ibm.com/topics/knn

26. IBM. (17 August, 2021). CRISP-DM Help Overview. Retrieved from IBM: https://www.ibm.com/docs/en/spss-modeler/saas?topic=dm-crisp-help-overview
27. Introduction to smart contracts. (2023, February 3). https://ethereum.org/en/developers/docs/smart-contracts/
28. Kabla, A H H., Anbar, M., Manickam, S., & Karuppayah, S. (2022, January 1). Eth-PSD: A Machine Learning-Based Phishing Scam Detection Approach in Ethereum. Institute of Electrical and Electronics Engineers, 10, 118043-118057. https://doi.org/https://doi.org/10.1109/access.2022.3220780
29. Lu, W. (2018, January 1). Blockchain Technology and Its Applications in FinTech. https://doi.org/10.1007/978-3-030-03712-3_10
30. Lui, Y. A.-G. (2021). An artificial intelligence system for predicting the deterioration of COVID-19 patients in the emergency department. NPJ Digital Medicine, 4(1), 1-11. doi:10.1038/s41746-021-00453-0
31. Ma, Z., Meng, J., Wang, J., & Shan, Z. (2021, February 15). Blockchain-Based Decentralized Authentication Modeling Scheme in Edge and IoT Environment. https://doi.org/10.1109/jiot.2020.3037733
32. Mayachita, I. (11 June, 2020). Understanding Graph Convolutional Networks for Node Classification. Retrieved from Towards Data Science: https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-a2bfdb7aba7b
33. Mohan, C. (2019, June 25). State of Public and Private Blockchains: Myths and Reality. https://dl.acm.org/doi/10.1145/3299869.3314116
34. Molnar, C. (2022). 5.5 Decision Rules. Retrieved from A Guide for Making Black Box Models Explainable: https://christophm.github.io/interpretable-ml-book/rules.html
35. Moreland, K. (29 May, 2023). What is Proof-of-Work. Retrieved from Ledger Academy: https://www.ledger.com/academy/blockchain/what-is-proof-of-work
36. Nikulski, J. (16 March, 2020). The Ultimate Guide to AdaBoost, random forests and XGBoost. Retrieved from Medium: https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f
37. NOROUZI, B. (2023, August 1). ENHANCING FRAUDULENT ACCOUNT DETECTION ON THE ETHEREUM BLOCKCHAIN: A NOVEL FEATURE SELECTION APPROACH AND COMPARATIVE ANALYSIS OF MACHINE LEARNING ALGORITHMS

38. Pandey, A. C., Verma, A., Rathor, V. S., Singh, M., & Singh, A. K. . (2023). Get this book in print ▼ Books on Google Play Intelligent Analytics for Industry 4.0 Applications. Boca Raton: CRC Press.
39. Peng, Z., & Chen, Y. (2018, January 1). All roads lead to Rome: Many ways to double spend your cryptocurrency. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1811.06751
40. Petrov, C. (31 January, 2023). 91+ Blockchain Statistics: Understand Blockchain in 2023. Retrieved from techjury: https://techjury.net/blog/blockchain-statistics/
41. Ravikiran A S. (17 January, 2023). Merkle Tree in Blockchain: What is it, How does it work and Benefits. Retrieved from Simplilearn: https://www.simplilearn.com/tutorials/blockchain-tutorial/merkle-tree-in-blockchain
42. Sallam, A A., Rassem, T H., Abdu, H., Abdulkareem, H., Saif, N., & Abdullah, S. (2022, July 1). Fraudulent Account Detection in the Ethereum's Network Using Various Machine Learning Techniques. UMP Publisher, 8(2), 43-50. https://doi.org/https://doi.org/10.15282/ijsecs.8.2.2022.5.0102
43. Satoshi, N. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from Bitcoin: https://bitcoin.org/bitcoin.pdf
44. Sayeed, S., & Marco-Gisbert, H. (2019, April 29). Assessing Blockchain Consensus and Security Mechanisms against the 51% Attack. https://www.mdpi.com/2076-3417/9/9/1788
45. Schweiger, L. (10 November, 2021). Blockdata. Retrieved from The State of Enterprise Blockchain in 2021: https://www.blockdata.tech/blog/general/the-state-of-enterprise-blockchain-in-2021
46. Staff, C. (2020, December 8). Eclipse Attacks Explained: What Are They?. https://www.gemini.com/cryptopedia/eclipse-attacks-defense-bitcoin
47. Taher, S S., Ameen, S Y., & Ahmed, J A. (2024, February 8). Advanced Fraud Detection in Blockchain Transactions: An Ensemble Learning and Explainable AI Approach. Engineering, Technology & Applied Science Research, 14(1), 12822-12830. https://doi.org/https://doi.org/10.48084/etasr.6641
48. The MathWorks, I. (24 March, 2017). What Are Convolutional Neural Networks? | Introduction to Deep Learning. Retrieved from MathWorks: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

49. Trace, P H. (2021, October 20). Parity multisig hack resulting in 153,037 stolen ETH. https://medium.com/parity-hack-trace/parity-hack-and-153-037-stolen-eth-2a7704f59f3b
50. Tran, N. K. , Babar, M. A. , Boan, J. . (2020). Integrating blockchain and Internet of Things systems: A systematic review on objectives and designs. Journal of Network and Computer Applications, 173(102844), 35.
51. What are the most common scams?. (2022, December 1). https://support.blockchain.com/hc/en-us/articles/4413805384852-What-are-the-most-common-scams-
52. What is a Blockchain Node Provider? Why Do I Need One? (12 October, 2021). Retrieved from Alchemy Blog: https://www.alchemy.com/blog/what-is-a-node-provider
53. Wu, H., Yao, Q., Liu, Z., Huang, B., Zhuang, Y., Tang, H., & Liu, C H. (2024, February 27). Blockchain for Finance: A Survey. https://doi.org/10.48550/arxiv.2402.17219
54. Xinyi, Y., Zhang, Y., & He, Y. (2018, July 1). Technical Characteristics and Model of Blockchain. https://doi.org/10.1109/iccsn.2018.8488289
55. Y. Elmougy, O. M. (2021). Anomaly Detection on Bitcoin, Ethereum Networks Using GPU-accelerated Machine Learning Methods. 31st International Conference on Computer Theory and Application, 166-171.
56. Yang, F., Shi, Y., Wu, Q., Li, F., Zhou, W., Hu, Z., Xiong, N., & Zhang, Y. (2019, May 1). The Survey on Intellectual Property Based on Blockchain Technology. https://doi.org/10.1109/icphys.2019.8780125
57. Yang, R., Wakefield, R., Lyu, S., Jayasuriya, S., Han, F., Yi, X., Yang, X., Amarasinghe, G., & Chen, J. (2020, October 1). Public and private blockchain in construction business process and information integration. https://doi.org/10.1016/j.autcon.2020.103276
58. Zhang, R., Xue, R., & Liu, L. (2019, July 3). Security and Privacy on Blockchain. Association for Computing Machinery, 52(3), 1-34. https://doi.org/10.1145/3316481
59. Zhang, T., Li, J., & Jiang, X. (2021, January 1). Supply chain finance based on smart contract. https://doi.org/10.1016/j.procs.2021.04.027

LIST OF FIGURES/TABLES/APPENDICES

MODEL PERFORMANCE RESULTS

Table 14: Model Performance Metrics

	Model	*CV Fold*	*Accuracy*	*Mean CV Score*	*Precision*	*Recall*	*F1 Score*	*Confusion Matrix [[TN, FP], [FN, TP]]*	*AUC-ROC Score*
0	Logistic Regression	3	0.901028	0.850262	0.910742	0.910352	0.910398	[[445, 48], [36, 408]]	0.910778
1	Logistic Regression	4	0.900531	0.851004	0.910742	0.910352	0.910398	[[445, 48], [36, 408]]	0.910778
2	Logistic Regression	5	0.900364	0.851572	0.910742	0.910352	0.910398	[[445, 48], [36, 408]]	0.910778
3	Logistic Regression	10	0.900282	0.873881	0.910742	0.910352	0.910398	[[445, 48], [36, 408]]	0.910778
4	Random Forest	3	0.970383	0.966331	0.963814	0.963714	0.963725	[[473, 20], [14, 430]]	0.963950
5	Random Forest	4	0.969222	0.966787	0.963814	0.963714	0.963725	[[473, 20], [14, 430]]	0.963950
6	Random Forest	5	0.972290	0.968617	0.963814	0.963714	0.963725	[[473, 20], [14, 430]]	0.963950

7	Random Forest	10	0.973038	0.970032	0.966946	0.966916	0.966921	[[476, 17], [14, 430]]	0.966993
8	XGBoost	3	0.971959	0.958311	0.965848	0.965848	0.965848	[[477, 16], [16, 428]]	0.965755
9	XGBoost	4	0.971462	0.957955	0.970165	0.970117	0.970124	[[477, 16], [12, 432]]	0.970259
10	XGBoost	5	0.973701	0.959201	0.964815	0.964781	0.964770	[[479, 14], [19, 425]]	0.964405
11	XGBoost	10	0.973619	0.959551	0.966924	0.966916	0.966910	[[479, 14], [17, 427]]	0.966657
12	SVM	3	0.969470	0.904688	0.970133	0.970117	0.970121	[[478, 15], [13, 431]]	0.970147
13	SVM	4	0.969802	0.905855	0.970133	0.970117	0.970121	[[478, 15], [13, 431]]	0.970147
14	SVM	5	0.971378	0.906428	0.971214	0.971185	0.971189	[[478, 15], [12, 432]]	0.971274
15	SVM	10	0.971959	0.907733	0.970133	0.970117	0.970121	[[478, 15], [13, 431]]	0.970147
16	DNN	3	0.953959	0.837511	0.955176	0.955176	0.955176	[[472, 21], [21, 423]]	0.955053

17	DNN	4	0.959021	0.820590	0.967046	0.966916	0.966927	[[474, 19], [12, 432]]	0.967217
18	DNN	5	0.958435	0.773273	0.969080	0.969050	0.969055	[[477, 16], [13, 431]]	0.969133
19	DNN	10	0.936260	0.794090	0.964918	0.964781	0.964760	[[481, 12], [21, 423]]	0.964181

Table 15: Best Parameters of the model on each cross-validation fold

	Model	*CV Fold*	*Best Parameters*
0	Logistic Regression	3	{'C': 1, 'class_weight': None, 'max_iter': 100, 'penalty': 'l1', 'solver': 'liblinear', 'tol': 0.0001}
1	Logistic Regression	4	{'C': 10, 'class_weight': None, 'max_iter': 100, 'penalty': 'l1', 'solver': 'liblinear', 'tol': 0.001}
2	Logistic Regression	5	{'C': 100, 'class_weight': None, 'max_iter': 100, 'penalty': 'l1', 'solver': 'liblinear', 'tol': 0.0001}
3	Logistic Regression	10	{'C': 10, 'class_weight': None, 'max_iter': 100, 'penalty': 'l2', 'solver': 'liblinear', 'tol': 0.01}

4	Random Forest	3	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}
5	Random Forest	4	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}
6	Random Forest	5	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}
7	Random Forest	10	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
8	XGBoost	3	{'colsample_bytree': 0.5, 'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 200, 'subsample': 0.9}
9	XGBoost	4	{'colsample_bytree': 0.7, 'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 200, 'subsample': 0.9}
10	XGBoost	5	{'colsample_bytree': 0.7, 'learning_rate': 0.5, 'max_depth': 4, 'n_estimators': 100, 'subsample': 0.9}
11	XGBoost	10	{'colsample_bytree': 0.7, 'learning_rate': 0.5, 'max_depth': 4, 'n_estimators': 200, 'subsample': 0.9}
12	SVM	3	{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
13	SVM	4	{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
14	SVM	5	{'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}

15	SVM	10	{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
16	DNN	3	{"L1": 1024, "L2": 512, "L3": 256, "L4": 128, "L5": 64, "activation": "relu", "batch_size": 128, "dropout_rate": 0.1, "epochs": 48, "learning_rate": 0.01, "loss": "binary_crossentropy", "metrics": ["accuracy"], "optimizer": "adam"}
17	DNN	4	{"L1": 1024, "L2": 512, "L3": 256, "L4": 128, "L5": 64, "activation": "relu", "batch_size": 128, "dropout_rate": 0.1, "epochs": 48, "learning_rate": 0.01, "loss": "binary_crossentropy", "metrics": ["accuracy"], "optimizer": "adam"}
18	DNN	5	{"L1": 1024, "L2": 512, "L3": 256, "L4": 128, "L5": 64, "activation": "relu", "batch_size": 128, "dropout_rate": 0.1, "epochs": 64, "learning_rate": 0.01, "loss": "binary_crossentropy", "metrics": ["accuracy"], "optimizer": "adam"}
19	DNN	10	{"L1": 1024, "L2": 512, "L3": 256, "L4": 128, "L5": 64, "activation": "relu", "batch_size": 128, "dropout_rate": 0.1, "epochs": 64, "learning_rate": 0.01, "loss": "binary_crossentropy", "metrics": ["accuracy"], "optimizer": "adam"}

LITERATURE REVIEW MATRIX

Author/ Article title, journal title, and publication details	Research Question(s)/ Hypotheses	Methodology	Analysis and Results	Conclusions	Implications for Future research/Practice
Author: Krishna Vishwanath Iyer V.V. Ravbi Kumar **Article Title:** Using blockchain technology in credit rating industry to promote an innovative bond-pays model. **Journal Title:** International Journal of Innovation **Publication Details:** Emerald Publishing Limited	**Research Questions:** • Issuer-pays model has led to conflict of interest resulting in rating shopping and inflation. • Alternative business models have their own problems, e.g., investor-pays model suffers from "free rider" and public dissemination challenges, whereas government-controlled business models can lead to market distortion. • Bond-pays model has been difficult to	This paper outlines current processes in credit rating business that has led to repeated rating failures and proposes a new set of processes, leveraging capabilities of blockchain technology to enable implementation of an arms-length bond-pays model.	A proof-of-concept system, namely, rating chain has been designed to implement a small part of the proposed model to establish technical feasibility in a blockchain environment.	A Proof-of-Concept system has been made based on some functionalities described in bond-pays model. It indicates the feasibility of using blockchain. However, the credit rating ecosystem has many challenges at the operational level as the current system must orchestrate multiple organizations. More proposals and additional field work to identify costs, benefits, timelines, and technical details.	Although the researchers have interviewed many representatives of the credit rating companies, there is a need for broader systematic study from other stakeholders, such as issuers, investors, rating agencies, regulators and more. Multiple rounds of iteration and experimentation will complete the roadmap to gain the confidence of the industry to build such solution.

	implement owing to operational difficulties in managing co-ordination amongst multiple entities involved, often with conflicting goals.				
Author: Antonio Fusco Grazia Dicuonzo Vittorio Dell'Atti Marco Tatullo **Article Title:** Blockchain in Healthcare: Insights on COVID-19 **Journal Title:** International Journal Environmental Research and Public Health **Publication Details:** MDPI Academic Open Access Publishing	**Hypothesis:** The use of blockchain in combination with artificial intelligence systems allows the creation of a generalizable predictive system that could contribute to the containment of pandemic risk on national territory.	A SWOT analysis of the adoption of a blockchain-based prediction model in healthcare and SARS-CoV-2 infection has been carried out to underline opportunities and limits to its adoption.	Strengths: • Disintermediation and automation • Immutability • Trust • Transparency • Privacy Weaknesses: • Operation costs • Creation of possible forks • Lack of flexibility • Need of greater capability of data storage on local servers Opportunities: • Greater collaboration among the operators of the healthcare system • Increase technological awareness and	Blockchain could play a strategic role in future digital healthcare: specifically, it may work to improve COVID19-safe clinical practice.	Blockchain can be used in new workflow or in improved protocols with particular attention to risk management.

			development of new expertise. Threats: • Resistance to change. • Lack of expertise • Lack of trust in the application of a new technology by healthcare workers		
Author: Youngjoo Lee Sooyong Park **Article Title:** Technology-based Practical Blockchain System Audit Maturity Model **Journal Title:** Technical Gazette 28 **Publication Details:** Mechanical Engineering Faculty in Slavonski Brod	**Research Questions:** The current audit model is insufficient for application in the field, and the auditing of systems applying new technologies, such as blockchain, has not been given sufficient attention. Furthermore, it is difficult to evaluate the relative levels of audited systems using audit results. Existing studies have only examined the auditing of systems that apply blockchain. Although the Korea Association of Information Systems Audit has suggested a checklist for systems applying blockchain, it has yet to be adopted.	50 existing audit result reports and technical data were collected from which sixteen factors of four audit quality properties consisting of blockchain system, technology compliance, software quality, and document were derived. Furthermore, an audit maturity model was presented after evaluating the priorities of the 16 derived factors.	The results indicate that IS audits should focus on both technology-based inspection and document-based inspections.	This study contributes to the literature by deriving field-oriented audit items including blockchain technology, thus enabling practical audits to be conducted in a short time. Further, this study enables the maturity of systems to be compared based on audit results by presenting audit maturity.	Future studies must investigate the automation of audit items so that inspection items can be continuously improved and quickly applied to the field based on the audit reports applied to the field. If audit items can be automated, the audit model and audit maturity can be updated quickly even if new technologies emerge.
Author: Christoph G. Schmidt, Stephan M. Wagner **Article Title:** Blockchain and supply chain relations: A transaction cost theory perspective	**Research Question:** How does blockchain affect supply chain relations?	Utilize the Transaction Cost Theory to develop an early idea of how blockchain might influence supply chain relations, specifically in terms of transaction costs and governance decisions.	Blockchain potentially • Limits opportunistic behavior. • Reduces behavioral uncertainty.	The blockchain might significantly reduce transaction and governance costs of supply chain transactions, specifically search and information cost.	Uncertainty is still existed in investigating the usage of blockchain, but the current paper can act as base which can be extended for other researchers. Will

Journal Title: Journal of Purchasing and Supply Management **Publication Details:** Mechanical Engineering Faculty in Slavonski Brod			• Reduces the impact of environmental uncertainty. in transactional relationships.	A blockchain-based economy might significantly push many transactions, even under restrictive condition, into a more market-oriented governance structure.	continue on researching the blockchain in context of supply chain relations.
Author: Mayer Ada Costa CRighi R **Article Title:** Electronic Health Records in a Blockchain: A systematic review **Journal Title:** Health Informatics Journal 26(2) **Publication Details:** Directory of Open Access Journals (DOAJ)	**Research Question:** • What are the challenges and questions about blockchain in healthcare? • What is the taxonomy for EHRs in a Blockchain? • What are the challenges and open questions that are related to EHRs in a Blockchain? • What are the important principles behind Blockchain when it is applied to? • healthcare? • What are the healthcare protocols and standards that should apply in a Blockchain? • network? • What are the types, models, and approaches	Using scientific literature review for analyzing all available, relevant research. After the research questions are made, a search strategy, article filter, quality assessment is defined.	Gone through the systematic process and found out that: • Blockchain can be divided into five main characteristics: governance, interoperability, privacy, scalability, and security. • Multiple challenges and questions are identified under those categories, for example lack of open standards, trust, privacy, cost, and others. • Important principles behind blockchain in healthcare are immutability, transparency and decentralization and others. • Based on the principles, some protocols and standards are found.	The blockchain technology has been thoroughly discussed and it is still highly dependent on whether the industry accepts the new technology. Additional research, trials and experiments must be conducted to ensure the system is ready to be deployed and scaled successfully, since EHR contains useful and private information of a patient's health.	This may serve as a basis or inspiration for future works and studies. It may help to contribute to addressing challenges, structures, characteristics of blockchain and IoT in healthcare industry. A possible future direction is to survey the combination of Blockchain and IoT in healthcare, with the objective of realizing network scalability with support of the low-end devices.

	of a Blockchain architecture? • How can Blockchain indefinitely store the "ever-growing" patient medical records? •		• The structure of blockchain, there are three types and features identified. • To store different types of data, different types of nodes must be deployed.		
Author: Joe Abou Jaoude Raafat George Saade **Article Title:** Blockchain Applications – Usage in Different Domains **Journal Title:** IEEEAccess **Publication Details:** IEEE	**Research Question:** What business fields have been addressed in current research on blockchain applications and how has it evolved since 2015? What solutions have been proposed for the major fields of blockchain applications? What are current research gaps in blockchain business applications? What are the future directions for blockchain business applications?	By conducting a system literature review with the following stages and steps: Stage 1: Criteria and Coding • Levels of Synthesis • Coding Steps • Occurrence/Frequency • Distinguish among concepts. Stage 2: Article Reduction • Process rules from steps 1 – 4 and document • Exclude irrelevant articles. • Encode text/information in articles and documents. Stage 3: Synthesis • Analysis of Results	The unique features of blockchain, such as privacy, security, anonymity, decentralization, and immutability provide valuable benefits to various fields and subjects. Exploring the blockchain application is only at early stages with limited studies, such as IoT, energy, finance, and others. These areas are not gaining the blockchain's benefits from the implementations.	Blockchain is expanding rapidly with a distinct evolution pattern among the different layers and concepts of blockchain implementation, which evolves from concepts of blockchain, improvements and application papers.	The next level of research of blockchain technology more towards cryptocurrencies and related user-oriented acceptance and adoption research to build business models that further integrates blockchain into various applications.
Author: Sirine Sayadi Sonia Ben Rejeb Zied Choukair **Article Title:** Anomaly Detection Model Over Blockchain Electronic Transactions **Journal Title:**	**Research Question:** What are the vulnerabilities existing in blockchain technology? What are the limitations of the current research on anomaly detection in electronic transactions?	Propose a new model for anomaly detection in Bitcoin electronic transaction using machine learning algorithms on two stages, which are: 1st stage: One-Class SVM method 2nd stage: K-means algorithm This new model is named as One Class Support Vector Machine (OCSVM).	The results of the proposed model showed good accuracy, but there are some false positives among the anomalies detected. Anyhow, the accuracy can reach 0.93 which is 93% in the result which is considered high precision.	Blockchain allows participants to transfer currency in secure way, but the anomaly detection shouldn't be ignored. The researchers have shown that the proposed model has good results according to their evaluation of performance.	Try to detect other vulnerabilities on blockchain technology by specifying the type of risk of false-positive.

15th International Wireless Communications and Mobile Computing Conference (IWCMC) **Publication Details:** Tangier, France					
Author: Yuanfeng Cai Dan Zhu **Article Title:** Fraud detections for online businesses: a perspective from blockchain technology **Journal Title:** Financial Innovation 2016 **Publication Details:** Springer Open	**Research Question:** • What machine learning algorithms have been developed to prevent information fraud? • How effective is blockchain technology in preventing rating fraud? • What potential implications of using such technology in fraud detection?	• Classifying subjective and objective fraud. • Discuss the effectiveness of blockchain technique in objective fraud. • Limitation of blockchain in subjective fraud. • Analyze the robustness of blockchain-based reputation in each type of rating fraud systematically.	The detection of malicious rating users is not easy since users behave systematically to evade fraud detection. Blockchain technology reputation systems can prevent bad-mouthing more than ballot-stuffing fraud.	Blockchain technology provides opportunities to redesign the reputation system. Fact based frauds are harder to implement in many other systems, such as loan application fraud.	Blockchain based systems can only block certain types of fraud, but effectiveness is limited in rating fraud as ground-truth is not easily validated. May suggest a possibility of using real identities when submitting ratings. There are more to research on blockchain technology and introduction of newer technologies to resolve both objective and subjective information frauds.
Author: Pierre-O. Goffard **Article Title:** Fraud Risk Assessment within Blockchain Transactions **Journal Title:** Applied Probability Trust 2019 **Publication Details:** Cambridge University Press	**Research Question:** • What is the probability of successfully spending twice the same bitcoins? • What is the probability of the malicious chain catches up with the honest chain?	Conduct theorems with corollaries to proof, which are: • The Probability Density Function (PDF) of the double-spending time • The Survival Function (SF) of double-spending time • The probability of a successful double-spending attack Some numerical illustrations are presented to show intensity functions over time with discussions.	The model is improvised to offer more flexibility and reflect block discovery accurately.	Based on theorems shown, the merchants should wait for a few subsequent blocks to be added to the chain before shipping the goods. Theorems have shown the time when the attack will likely occur.	The contribution of this paper can be relevant to understanding other systems that use similar policies. The future research will continue investigating some interesting founding that aligns with other researchers.

Author: Zhiyuan Chen Le Dinh Van Khoa Ee Na Teoh Amril Nazir Ettikan Kandasamy Karuppoiah Kim Sim Lam **Article Title:** Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review **Journal Title:** Knowledge and Information Systems 2018 **Publication Details:** Springer Nature	**Research Question:** • What are the techniques used in both supervised and unsupervised machine learning in AML? • What are the patterns of money laundering in various financial circumstances?	Surveys most recently published literature of machine learning techniques for AML solutions in suspicious transaction detection area. The paper will be reviewed in the following order: • Most important preprocessing and analysis steps for an AML solution. • Existing detection tools for AML solutions that apply machine learning algorithms and statistical measurements. • Commercial software that provides antimony laundering solutions.	Among the techniques used by the researchers, most of the reviewed papers preferred AML typologies and anomaly detection algorithms. The accuracy of these tools is difficult to determine due to the different datasets and modelling requirements. Most of the reviewed papers: Artificially creating fraudulent transactions. Did not use reinforcement learning. Using decision tree for rule-based approaches. K-means approaches for comparing with other techniques or reducing learning pattern for classification. Comparing ground truth with confusion matrix. For commercial solutions, a few features are used to ensure effective AML: • Suspicious activity monitor • Risk scoring. • Rule-based approaches • Recording behavioral activities with peer-based analytics. • Using Support Vector Machines (SVM) with probabilistic thresholds	Having an effective AML system is important to manage the risk of money laundering activities in high automation, but the situation is challenging as large customer volume and transactions flows in which makes AML compliance difficult. Several important features need to be identified, which are effective: • Data preparation • Data transformation • Data analytic techniques Data quality assurance needs to be improved as there are inconsistencies among papers during the process of data engineering.	Financial operations may vary from time to time, hence the applied techniques need to apply reinforcement learning needs to be considered, so that the model can adapt to the situation.

			• Multi-dimensional adaptive probabilistic matrix with peer groups. • Bayesian belief network (BNN) to make decisions, lowering false positives. However, AML systems have privacy issues due to the pervasive mining into the bank customers' information. Over bias learning may happen if the customer policy is not understood well.		
Author: Mark Weber Giacomo Domeniconi Jie Chen Daniel Karl I. Weidele Claudio Bellei Tom Robinson Charles E. Leiserson **Article Title:** Anti-Money Laundering in Bitcoin/ Experimenting with Graph Convolutional Networks for Financial Forensics **Journal Title:** KDD '19 Workshop on Anomaly Detection in Finance **Publication Details:** Association for Computing Machinery, Anchorage, AK, USA	**Research Question:** What methods are available to reconcile the safety with the cause of financial inclusion?	Acquired dataset from Elliptic, which is a graph network of Bitcoin transactions with consent from the organization. Perform data analysis on the dataset and used AML analysis methods such as Logistic Regression, Random Forest, Graph Convolutional Networks (GCN), Temporal Modeling, Recurrent Neural Network (RNN).	Random Forest outperforms Logistic Regression and GCN. Created a visualization prototype called Chronograph to display high-dimensional graph imposes a layer of complexity on top of plain feature vectors. It can also explain the performance of the model. This visualization is interactable as the user can adjust time to see the changes, with illicit transactions tagged with colors as hint. The shape and grouping also indicates the performance of the model is effective or not.	A large, labelled transaction dataset has been demonstrated and contributed to the community for research study, as it has never been publicly available before. Early experimental results with techniques including GCN have been visualized. Some discussions have been made on the possible paths that can be used on the current techniques to give inspirations to other researcher to make financial systems safer.	There are some ideas of combining Random Forest with a graph neural network. Other ideas can be considered, which includes parameterizing every node in the decision tree(s) by using a feed-forward neural network, with additional or replace the Logistic Regression output layer in GCN.
Author: Harmeet Kaur Khanuja	**Research Questions:**	• Setup outlier detection rules based on the	The experiment shows that the developed model	Using proposed approach with DST on real life	The future work, the proposed approach

Dattatraya Adane **Article Title:** Monitor and Detect Suspicious Transactions with Database Forensic Analysis **Journal Title:** Journal of Database Management **Publication Details:** IGI Global	What are the methods that can be used to monitor and detect suspicious transactions in the database systems? **Hypothesis:** Use of Dempster Shafer Theory (DST) to perform audit on the database transactions with evidential reasoning.	existing auditing capabilities of the database systems. • Perform data engineering with ETL process. • Using theories such as Dempster Shafer Rule of combination and Bel(s) and Plausibility Function PI(s) which is used to develop Rule-based outlier detection algorithm and risk-based assessment. • After that, DST analysis is performed.	achieved 97% accuracy. When compared with other techniques with same data, they also have similar performance, but the proposed algorithm has the closest accuracy to 100%. This shows the hybrid approach is an improvement compared to other comparative algorithms.	dataset, the proposed algorithm performs better results which is improvement compared to other algorithms.	can be extended to detect the suspicious transactions on real time which can alert on fraud cases sooner. Faster detection can prevent undesired, irreversible outcomes in financial transactions.
Author: Martin Jullum, Anders Løland Ragnar Bang Huseby **Article Title:** Detecting money laundering transactions with machine learning **Journal Title:** Journal of Money Laundering **Publication Details:** Emerald Publishing Limited	**Research Question:** • How to ensure the AML system rules are up-to-date continuously? • How to enhance the rule-based systems in AML? • How to reduce the number of false positives?	A supervised machine learning model is trained by using three types of historic data: “normal” legal transactions; those flagged as suspicious by the bank’s internal alert system; and potential money laundering cases reported to the authorities. The model is trained to predict the probability that a new transaction should be reported, using information such as background information about the sender/receiver, their earlier behavior, and their transaction history.	The paper demonstrates that the common approach of not using non-reported alerts in the training of the model can lead to sub-optimal results. The same applies to the use of normal transactions. The proposed method outperforms the bank’s current approach in terms of a fair measure of performance.	The proposed model has demonstrated great performance against real life transaction data. However, the dataset needs to be comprehensive, including non-reported cases. All cases should be recorded, no matter if they are important or not, in daily activities.	Modelling procedure can be further improved due to the current experiment is modelled with cooperation with AML experts. XGBoost framework is efficient, but it can be computationally demanding when scaled. The frequency of scanning through transactions history needs to be controlled optimally. In future research, the similar method can be used on customers instead of transactions.
Author: Meng Jiang, Alex Beutel, Peng Cui, Bryan Hooi, Shiqiang Yang **Article Title:**	**Research Questions:** • How is behavior analysis used to detect social	Giving a list of axioms that any metric suspiciousness should satisfy. Proposing an algorithm called CROSSPOT to spot dense blocks that is worth inspecting with sort	In summary, with the defined axioms, the proposed algorithm can perform better than existing techniques by 68%, tracking suspicious	The paper has addressed the problem of evaluating and detecting dense blocks of suspicious behaviors in multimodal behavioral data. Five sets	In future work, the metric will be derived on more sophisticated models.

Spotting Suspicious Behaviors in Multimodal Data/ A General Metric and Algorithms **Journal Title:** IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING **Publication Details:** IEEE	media platform attacks? • What are the limitations of traditional approaches in detecting such attacks? • What axioms that should be set as metric of suspiciousness satisfy?	order of importance using real dataset.	behavioral patterns in social datasets spanning 0.3 billion posts.	of axioms have been set and met. CROSSPOT can detect suspicious behavior in dense data of multimodal data in terms of F1 score, compared to the existing techniques.	
Author: Branka Stojanovic´ Josip Božic´ Katharina Hofer-Schmitz Kai Nahrgang Andreas Weber Atta Badii Maheshkumar Sundaram Elliot Jordan Joel Runevic **Article Title:** Follow the Trail: Machine Learning (ML) for Fraud Detection in Fintech Applications **Journal Title:** Sensors 2021 **Publication Details:** MDPI, Basel, Switzerland	**Research Question:** • How fraud detection in the fintech domain is conducted? • What machine learning techniques are used to perform financial fraud data classification?	An experiment workflow has been outlined as the following: • Case studies selection • Data collection • Data statistics and visualization • Feature engineering and selection • Algorithm selection • Evaluation	The results have shown that the current techniques' performance is not consistent when used across datasets. This indicates that identification of incorrect predictions across different datasets remains a difficult challenge. There's a need of more input parameters to make the results more reliable.	A review of the existing methods from ML domain into fintech industry have been thoroughly discussed. The ML methods for anomaly detection have been evaluated. From there, an experiment has been conducted with these ML techniques which give benefits of ML in detecting anomalies in complex datasets. In addition, using ML techniques can improve the security of fintech systems.	There are more technical challenges in detecting suspicious behaviors in fintech domain. First, the techniques are heavily dependent on initial configurations. Second, there are tradeoffs when adjusting the performance of various algorithms, resulting in false positives. Financial crime is often regarded as minor crime that is considered unserious consequences. All responsible authorities in the society needs to engage together to identify the perpetrators of the crime.
Author: Muhammad Muzammal Quang Qu Bulat Nasrulin	**Research Questions:** • What are the current limitations of	Describe the proposed system CHAINSQL with the following sections:	When CHAINSQL system is deployed, the operations are like current Structured Query Language (SQL).	CHAINSQL system has demonstrated the use cases mentioned and showed its feasibility	Some considerations can be done in the future, such as Big Data analytics and

Article Title: Renovating blockchain with distributed databases: An open-source system **Journal Title:** Future Generation Computer Systems **Publication Details:** Elsevier	the existing database systems? • How does the proposed system address the problem of database systems with blockchain technology?	Highlights Significance Datastore Consensus Validation Network Security and Privacy Features/Use cases Along with demonstrations of the system using illustrations and tables to evaluate the performance.	The performance of the system is like Ripple, due to the consensus protocol.	some business such as finance and supply chain.	complex indexing for query optimization.
Author: Steve Huckle Riptuparna Bhattacharya Martin White Natalia Beloff **Article Title:** Internet of Things, Blockchain and Shared Economy Applications **Journal Title:** Future Generation Computer Systems **Publication Details:** ScienceDirect, Elsevier	**Research Questions:** • What are the existing exploitations in blockchain in Internet of Things (IoT)? • What are the concepts for building IoT applications based on blockchain?	A proposed IoT application that based on service orientation approach, which used a few key IoT and blockchain technologies	A few blockchain integrated IoT scenarios are made to describe how the prototype works in real life scenario, such as AutoPay, John's International Tour, Digital Rights Management, and others.	The prototype describes the application scenarios with blockchain integration. Current research team has explored the IoT research of Sussex's Shared Things.	Deepen research interest in mobile financials and IoT in sharing economy, integrate wearable technology, develop the prototype in real life with codes in a summer hackathon.
Author: P. Monamo, V. Marivate and B. Twala **Article title**: Unsupervised learning for robust Bitcoin fraud detection **Journal title**: 2016 Information Security for South Africa (ISSA),	Research Question: how to effectively detect potential fraud in the Bitcoin network using unsupervised learning techniques. Hypotheses: • Trimmed k-means will be more effective at detecting fraudulent transactions than similar	The methodology of this article involves the use of unsupervised learning techniques, specifically trimmed k-means clustering, to detect potential fraud in the Bitcoin network. The authors used a dataset of Bitcoin transactions and extracted various features from the data. They then applied the trimmed k-means algorithm to cluster the transactions and identify potential outliers that may indicate fraudulent activity. The authors evaluated the effectiveness of their approach by	The proposed approach using trimmed k-means clustering was effective at detecting potential fraud in the Bitcoin network. The authors found that their approach detected more fraudulent transactions than similar studies or reports on the same dataset. They also found that the adoption of recent technological developments, such as BIP and HDM, when coupled with good performing fraud	• The proposed approach using trimmed k-means clustering was effective at detecting potential fraud in the Bitcoin network. • The approach detected more fraudulent transactions than similar studies or reports on the same dataset.	• Future research should focus on advanced feature extraction to improve the effectiveness of fraud detection algorithms. • Evaluation of algorithmic performance regarding flagged suspicious activities should be improved.

Publication details: Johannesburg, South Africa. pp. 129-134, Doi: 10.1109/ISSA.2016.7802939	studies or reports on the same dataset .	comparing it to similar studies or reports on the same dataset.	detection algorithms, enhanced financial security of users on the peer-to-peer network. However, the authors noted that there is still more work to be done in terms of advanced feature extraction and evaluation of algorithmic performance regarding flagged suspicious activities. Additionally, they acknowledged that the unavailability of labels made it difficult to evaluate algorithmic performance regarding flagged suspicious activities.	• Adoption of recent technological developments, such as BIP and HDM, when coupled with good performing fraud detection algorithms, enhanced financial security of users on the peer-to-peer network. • There is still more work to be done in terms of advanced feature extraction and evaluation of algorithmic performance regarding flagged suspicious activities. The unavailability of labels made it difficult to evaluate algorithmic performance regarding flagged suspicious activities.	• The development of more accurate and efficient fraud detection algorithms is necessary to enhance financial security in the Bitcoin network. • Future research should explore the use of other unsupervised learning techniques for fraud detection in the Bitcoin network. Practice should adopt recent technological developments, such as BIP and HDM, when coupled with good performing fraud detection algorithms, to enhance financial security of users on the peer-to-peer network.
Author: P. Monamo, V. Marivate and B. Twala **Article title**: Unsupervised learning for robust Bitcoin fraud detection **Journal title**: 2016 Information Security for South Africa (ISSA),	Research Question: how to effectively detect potential fraud in the Bitcoin network using unsupervised learning techniques. Hypotheses: • Trimmed k-means will be more effective at detecting fraudulent transactions than similar	The methodology of this article involves the use of unsupervised learning techniques, specifically trimmed k-means clustering, to detect potential fraud in the Bitcoin network. The authors used a dataset of Bitcoin transactions and extracted various features from the data. They then applied the trimmed k-means algorithm to cluster the transactions and identify potential outliers that may indicate fraudulent activity. The authors evaluated the effectiveness of their approach by	The proposed approach using trimmed k-means clustering was effective at detecting potential fraud in the Bitcoin network. The authors found that their approach detected more fraudulent transactions than similar studies or reports on the same dataset. They also found that the adoption of recent technological developments, such as BIP and HDM, when coupled with good performing fraud	• The proposed approach using trimmed k-means clustering was effective at detecting potential fraud in the Bitcoin network. • The approach detected more fraudulent transactions than similar studies or reports on the same dataset.	• Future research should focus on advanced feature extraction to improve the effectiveness of fraud detection algorithms. • Evaluation of algorithmic performance regarding flagged suspicious activities should be improved.

Publication details: Johannesburg, South Africa. pp. 129-134, Doi: 10.1109/ISSA.2016.7802939	studies or reports on the same dataset .	comparing it to similar studies or reports on the same dataset.	detection algorithms, enhanced financial security of users on the peer-to-peer network. However, the authors noted that there is still more work to be done in terms of advanced feature extraction and evaluation of algorithmic performance regarding flagged suspicious activities. Additionally, they acknowledged that the unavailability of labels made it difficult to evaluate algorithmic performance regarding flagged suspicious activities.	• Adoption of recent technological developments, such as BIP and HDM, when coupled with good performing fraud detection algorithms, enhanced financial security of users on the peer-to-peer network. • There is still more work to be done in terms of advanced feature extraction and evaluation of algorithmic performance regarding flagged suspicious activities. • The unavailability of labels made it difficult to evaluate algorithmic performance regarding flagged suspicious activities.	• The development of more accurate and efficient fraud detection algorithms is necessary to enhance financial security in the Bitcoin network. • Future research should explore the use of other unsupervised learning techniques for fraud detection in the Bitcoin network. • Practice should adopt recent technological developments, such as BIP and HDM, when coupled with good performing fraud detection algorithms, to enhance financial security of users on the peer-to-peer network.
Author: Thai T. Pham and Steven Lee **Article** Title: Anomaly Detection in the Bitcoin System - A Network Perspective **Journal Title**: IEEE International Conference on Intelligence and Security Informatics (ISI)	Research Question: How to detect anomalous behavior in the Bitcoin transaction network using Network Analysis techniques?	Involves using Network Analysis techniques to detect anomalous behavior in the Bitcoin transaction network. The authors describe their methods in Section 3, which includes data collection and parsing, feature extraction, and mathematical descriptions of the network analysis techniques. They then discuss evaluation methods in Section 4 and present	• The authors used Network Analysis techniques to detect anomalous behavior in the Bitcoin transaction network. • They represented the data with two focuses: users and transactions. • They used three main social network	• The authors investigated the Bitcoin network and used Network Analysis techniques to detect anomalous behavior. • They were able to detect one known case of theft out of	• The authors suggest that their methods could be useful for detecting suspicious behavior in other types of networks beyond the Bitcoin

Publication Details: 2015 IEEE International Conference on Intelligence and Security Informatics (ISI), Baltimore, MD, 2015, pp. 61-66.		the results they obtained by running these techniques on the network-type data set they generated in Section 5. Finally, they evaluate their methods and results in Section 6 and mention future works in Section 7. The study concludes with a summary of their findings in Section 8.	techniques to detect anomalies, which are potential anomalous users and transactions. • While the agreement metrics were not high, they were able to detect one known case of theft out of the 30 known cases they had. The authors worked on a bigger problem of anomaly detection in networks, and these techniques could apply in other network settings.	the 30 known cases they had. • While the agreement metrics were not high, these techniques could apply in other network settings. • The authors worked on a bigger problem of anomaly detection in networks. • They concluded that their methods could be useful for detecting suspicious behavior in the Bitcoin transaction network and other types of networks.	transaction network. • They note that their results were not highly accurate, and there is room for improvement in terms of agreement metrics. • The authors suggest that future research could focus on improving the accuracy of Network Analysis techniques for detecting anomalous behavior. • They also suggest that future research could explore other types of data sets and network structures to see how well these techniques perform in different contexts. • Finally, the authors suggest that their work has practical implications for detecting fraud and other types of suspicious behavior in

					financial transactions.
Author: Thai T. Pham and Steven Lee **Article** Title: Anomaly Detection in Bitcoin Network Using Unsupervised Learning Methods **Journal Title**: 2015 IEEE International Conference on Intelligence and Security Informatics (ISI) **Publication Details**: 2015 IEEE International Conference on Intelligence and Security Informatics (ISI), Baltimore, MD, 2015, pp. 61-66.	**Research Question**: How to detect anomalies in the Bitcoin transaction network using unsupervised learning methods?	• Data collection and parsing: The authors collected data from the Bitcoin transaction network and parsed it into two graphs, one with users as nodes and the other with transactions as nodes. • Feature extraction: The authors extracted various features from the graphs, including degree centrality, clustering coefficient, and PageRank. • Unsupervised learning techniques: The authors applied three unsupervised learning techniques (k-means clustering, Mahalanobis distance-based method, and Unsupervised Support Vector Machines) to the feature vectors of each node in the graphs to detect anomalies. • Evaluation methods: The authors evaluated the effectiveness of each technique using precision-recall curves and F1 scores. Results analysis: The authors analyzed the results of their experiments to determine which technique was most effective at detecting suspicious behavior in the Bitcoin network.	• The authors applied three unsupervised learning techniques (k-means clustering, Mahalanobis distance-based method, and Unsupervised Support Vector Machines) to detect anomalies in the Bitcoin transaction network. • They extracted various features from the graphs, including degree centrality, clustering coefficient, and PageRank. • The authors evaluated the effectiveness of each technique using precision-recall curves and F1 scores. • The results showed that all three techniques were effective at detecting anomalies in the Bitcoin network. • However, k-means clustering performed slightly better than the other two techniques in terms of F1 score. • The authors also found that certain features were more important than others for detecting anomalies in the network. For example, degree centrality was a strong	• The authors successfully applied unsupervised learning techniques to detect anomalous behavior in the Bitcoin transaction network. • All three techniques (k-means clustering, Mahalanobis distance-based method, and Unsupervised Support Vector Machines) were effective at detecting anomalies, with k-means clustering performing slightly better. • Certain features, such as degree centrality, were strong indicators of suspicious behavior in the network. • The study demonstrates the potential of unsupervised learning techniques for detecting financial crimes in networks like Bitcoin. • Future work could explore the use of supervised learning techniques and other	• The study highlights the potential of unsupervised learning techniques for detecting financial crimes in networks like Bitcoin, and future research could explore the use of these techniques in other types of financial networks. • The authors suggest that supervised learning techniques could be used to improve the accuracy of anomaly detection in financial networks. • The study also highlights the importance of certain features, such as degree centrality, for detecting suspicious behavior in financial networks. Future research could explore other

			indicator of suspicious behavior. Overall, the study demonstrates that unsupervised learning techniques can be effective at detecting anomalous behavior in financial networks like Bitcoin.	types of financial networks.	features that may be important for anomaly detection. • The authors note that their study is limited by the size and scope of their dataset, and future research could use larger datasets to further validate their findings. • Finally, the study has practical implications for law enforcement agencies and financial institutions that are interested in detecting financial crimes in Bitcoin and other digital currencies.
Author: Weili Chen, Zibin Zheng, Jiahui Cui, Edith Ngai, Peilin Zheng, and Yuren Zhou **Article title:** Detecting Ponzi Schemes on Ethereum: Towards Healthier Blockchain Technology **Journal/Conference title:** WWW 2018: The 2018 Web Conference **Publication details:** ○ **Date: April 23–27, 2018** ○ **Location: Lyon, France** ○ **Publisher:** ACM, New York, NY, USA ○ **Article number:** 4 ○ **Pages:** 10 pages	**Research question:** Can we detect Ponzi schemes on Ethereum using machine learning techniques? **Hypothesis:** By analyzing the bytecode of smart contracts and extracting relevant features, we can train a classification model to accurately identify Ponzi schemes on Ethereum.	• Data collection: 1382 verified smart contracts were collected from http://etherscan.io, including 131 Ponzi scheme contracts and 1251 non-Ponzi scheme contracts. • Data labeling: The smart contracts were manually inspected to determine whether they were Ponzi schemes or not, and the results were used as ground truth data for the model. • Feature extraction: Two categories of features were extracted from the smart	• The machine learning model achieved an accuracy of 96.5% in detecting Ponzi schemes on Ethereum. • The precision, recall, and F1 score of the model were also reported, with precision being the highest among the three metrics. • The analysis of the results revealed that Ponzi schemes on Ethereum share common	• The study demonstrated that machine learning techniques can be effective in detecting Ponzi schemes on Ethereum. • The analysis of the results revealed common characteristics of Ponzi schemes on Ethereum, which can be used to develop more effective detection methods.	• The study provides a foundation for future research on detecting fraudulent activities on blockchain platforms using machine learning techniques. • The analysis of the results can be used to develop more effective detection methods and improve user

DOI: https://doi.org/10.1145/3178876.3186046		contract data: opcode frequency and control flow graph (CFG) features. • Model training and evaluation: A machine learning model was trained on the extracted features using a random forest classifier. The model was evaluated using various metrics, including accuracy, precision, recall, and F1 score. • Results analysis: The results of the model were analyzed to identify key characteristics of Ponzi schemes on Ethereum and potential solutions to address this issue. • Data collection: 1382 verified smart contracts were collected from http://etherscan.io, including 131 Ponzi scheme contracts and 1251 non-Ponzi scheme contracts. • Data labeling: The smart contracts were manually inspected to determine whether they were Ponzi schemes or not, and the results were used as ground truth data for the model. • Feature extraction: Two categories of features were extracted from the smart contract data: opcode frequency and control flow graph (CFG) features. • Model training and evaluation: A machine learning model was trained	characteristics, such as high returns, short contract duration, and a large number. • The study also proposed several potential solutions to address the issue of Ponzi schemes on Ethereum, including regulatory measures and improved user education. Overall, the study demonstrated that machine learning techniques can be effective in detecting fraudulent activities on blockchain platforms like Ethereum.	• The study also highlighted the need for regulatory measures and improved user education to address the issue of Ponzi schemes on blockchain platforms like Ethereum. • Overall, the study contributes to the development of healthier blockchain technology by identifying and addressing fraudulent activities on these platforms.	education to prevent Ponzi schemes on blockchain platforms like Ethereum. • The study highlights the need for regulatory measures to address fraudulent activities on blockchain platforms, which can be further explored in future research. • Overall, the study contributes to the development of healthier blockchain technology by identifying and addressing fraudulent activities on these platforms, which can have significant implications for both research and practice.

		on the extracted features using a random forest classifier. The model was evaluated using various metrics, including accuracy, precision, recall, and F1 score. Results analysis: The results of the model were analyzed to identify key characteristics of Ponzi schemes on Ethereum and potential solutions to address this issue.			
Author: Eunjin (EJ) Jung, Marion Le Tilly, Ashish Gehani, and Yunjie Ge. **Article** Title: Data Mining-based Ethereum Fraud Detection **Conference Title**: 2019 IEEE International Conference on Blockchain (Blockchain) **Publication Details**: Atlanta, GA, USA	**Research question**: How to use data mining techniques to detect Ponzi schemes on the Ethereum blockchain, and improve upon prior work in this area	• The researchers used data mining techniques to develop a detection model for Ponzi schemes on Ethereum. • They built a dataset of likely benign Ethereum smart contracts, in addition to known Ponzi scheme smart contracts, and designed features based on their compiled code and transactions. • They used Weka to benchmark several classification algorithms to obtain models that achieve high precision and high recall. They analyzed top-strength features to gain novel perspectives on Ponzi scheme behavior.	• 3 model has been made, which are: • 0-day: Only use features that are available as soon as the smart contract is in the database. • Behavior based model: Check current performance compared to prior base models. In summary, there are some improvements in J48 model, and Random Forest is better than previous works. SGD has highest precision but worse in F-score. Full feature model: Using whole feature of the dataset into the modelling process. Little improvement over the 0-day models. Average amount of investment is useful on first day. Lifetime of non-Ponzi is longer than Ponzi transactions.	• Ponzi smart contracts on the blockchain are presented as high-return investment opportunities, while the risks involved are barely mentioned. • The authors-built data mining-based Ponzi detection models, which achieved high precision and recall and improved over prior work. • The 0-day model can be used to flag Ponzi smart contracts as soon as they are uploaded to the blockchain. • The full-feature model can be used to flag suspicious contracts so potential investors know about their risk. • The data mining-based Ponzi	• Data mining-based Ponzi detection models can be effective in detecting fraudulent smart contracts on the blockchain. This can serve as the first line of defense against scammers who present Ponzi smart contracts as high-return investment opportunities. The models developed in this research can be used to flag suspicious contracts and warn potential investors about the risks involved. The results of the study also suggest that the behavior-based features can be useful in

				detection can serve as the first line of defense against fraudulent smart contracts.	detecting active attackers who manipulate the smart contracts to evade the detection.
Author(s): Binjie Chen, Fushan Wei, and Chunxiang Gu **Article title:** Bitcoin Theft Detection Based on Supervised Machine Learning Algorithms **Journal title:** Security and Communication Networks Volume 2021 **Publication details:** Article ID 6643763, 10 pages	**Research Questions**: • How effective are supervised machine learning algorithms in detecting Bitcoin theft events? • Can supervised machine learning algorithms accurately predict Bitcoin theft events before they occur? • Which supervised machine learning algorithms perform the best in detecting Bitcoin theft events? • How can feature engineering improve the performance of supervised machine learning algorithms in detecting Bitcoin theft events? • Can supervised machine learning algorithms be trained to detect new types of Bitcoin theft events that were not previously identified? What is the impact of the size and quality of the training dataset on the performance of	• The study mainly used five supervised learning methods and three unsupervised learning methods. • The authors labeled the data based on reported Bitcoin theft events and utilized five supervised methods to detect Bitcoin theft. • The authors extracted features from the dataset, including transaction amount, transaction fee, number of inputs/outputs, etc. • The authors evaluated the performance of the supervised methods using recall and precision metrics. The authors compared the performance of supervised and unsupervised methods to demonstrate the advantages of using supervised algorithms.	• The authors found that the KNN, RF, and AdaBoost algorithms achieved good results on their dataset, with F1 values exceeding 80%. • The RF algorithm achieved recall, precision, and F1 values of 95.9%. • The authors compared the performance of supervised and unsupervised methods and found that supervised methods significantly improved recall and precision metrics. The authors concluded that using supervised machine learning algorithms can effectively detect Bitcoin theft events and provide warnings to prevent further losses.	• The authors used supervised machine learning algorithms to detect Bitcoin theft events and found that the KNN, RF, and AdaBoost algorithms achieved good results on their dataset. • The authors concluded that using supervised algorithms can effectively improve recall and precision metrics compared to unsupervised methods. • The authors suggested that their approach could be used to provide warnings about Bitcoin theft events and prevent further losses.	• The authors suggested that their approach could be extended to other cryptocurrencies and financial systems to detect fraudulent activities. • The authors recommended that future research should focus on improving the accuracy of detection algorithms and reducing false positives. • The authors suggested that their approach could be used by financial institutions, regulators, and law enforcement agencies to prevent financial crimes. • The authors also recommended that future research should explore the use of

	supervised machine learning algorithms in detecting Bitcoin theft events?				blockchain technology to enhance the security and transparency of financial transactions.
Author: Safa Alsafari, T.A., R.K., A.S.Y., S.A. (Sheraz Aslam), S.A. (Safa Alsafari), A.T.A., and I.A.H. **Article title**: A Machine Learning and Blockchain Based Efficient Fraud Detection Mechanism **Journal title:** Electronics **Publication details:** Volume 10, Issue 22, November 2021, Article number 2769	**Research Questions**: • Can machine learning algorithms (XGBoost and random forest) be used to accurately classify fraudulent and integrated transactions? • Can a blockchain-based system improve the security and efficiency of fraud detection in e-banking and online transactions? How does the proposed fraud detection mechanism compare to traditional methods in terms of accuracy, efficiency, and security?	• The authors collected a dataset of e-banking and online transaction records from a financial institution. • They used two machine learning algorithms (XGBoost and random forest) to classify transactions as fraudulent or integrated based on patterns in the dataset. • The authors developed a blockchain-based system to securely store and verify transaction records. They compared the performance of their proposed mechanism to traditional fraud detection methods using metrics such as accuracy, efficiency, and security.	• The authors found that both machine learning algorithms (XGBoost and random forest) were effective at classifying fraudulent and integrated transactions, with XGBoost achieving slightly higher accuracy. • The proposed blockchain-based system was found to improve the security and efficiency of fraud detection compared to traditional methods. • The authors reported high accuracy rates for their proposed mechanism, with an F1 score of 0.99 for XGBoost and 0.98 for random forest. • They also found that their mechanism was able to detect previously undetected fraudulent transactions in the dataset. Overall, the authors concluded that their proposed mechanism offers a promising approach to improving fraud detection	• The authors proposed a machine learning and blockchain-based mechanism for fraud detection in e-banking and online transactions. • They found that their proposed mechanism was effective at detecting fraudulent transactions with high accuracy rates. • The authors concluded that their mechanism offers a promising approach to improving the security and efficiency of fraud detection in the financial sector. • They suggested that future research could	• The authors suggested that future research could explore the use of other machine learning algorithms or additional features to further improve the accuracy of fraud detection. • They also noted that their proposed mechanism could be integrated into existing financial systems to improve their security and efficiency. • The authors highlighted the importance of ongoing research and development in fraud detection mechanisms as financial fraud continues to evolve and become more sophisticated. • They suggested that practitioners

			in e-banking and online transactions.	explore the use of other machine learning algorithms or additional features to further improve the accuracy of fraud detection.	in the financial sector should consider adopting blockchain-based systems and machine learning algorithms to improve their fraud detection capabilities.
Author: MacQueen, J. B. **Article title:** Some Methods for classification and Analysis of Multivariate Observations **Journal title:** Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability **Publication details:** 1967	**Research Questions**: • What kind of pellicular properties of the users that make them stand out to Bitcoin fraud? How the robbery of the Bitcoin transaction happened in previous notable event(s)?	• Perform k-means clustering on the Bitcoin dataset until July 13, 2011. • Perform feature extraction and building k-means model. Feature scaling and data normalization to separate good and bad users.	• K-means on all features: Unable to classify rogue users from the good users, resulted in one big cluster (99%). K-means on selected features: Successfully classified good and bad users into 2 clusters. Able to achieve 76.5% in clustering rogue users.	• The authors have proposed a model comprised of six unique features that can help identify potential rogue users in the Bitcoin network. • The authors acknowledge that the model is not perfect, but they believe it is a good starting point for further analysis. • The authors also mention that the model was able to separate good users from rogue users in their tests based on reported robberies. • Additionally, they were able to achieve a 76.5 percent accuracy rate in identifying synthetically generated rogue users using their model	• The proposed model may be useful in detecting potential rogue users in the Bitcoin network, which could help prevent Bitcoin theft and improve the overall security of the network. •
Author: Ostapowicz, Michał, Żbikowski, Kamil	**Research question**: Can supervised learning algorithms be used to	• Data collection: The authors downloaded data from the Etherscan API and	• Analysis: The authors analyzed the performance of three	• The authors proposed a machine	• Future research could focus on developing more

Article title: Detecting Fraudulent Accounts on Blockchain: A Supervised Approach **Journal title:** Web Information Systems Engineering – WISE 2019 **Publication details:** pp 18–31, vol 11881. Springer, Cham	detect fraudulent accounts on the Ethereum blockchain? Hypothesis: By using a supervised learning approach and training on a set of labeled data, it is possible to accurately identify fraudulent accounts on the blockchain.	aggregated it to create 13 variables. • Supervised learning algorithms: The authors used three different supervised learning algorithms (Random Forest, Support Vector Machines, and k-Nearest Neighbors) to classify accounts as either fraudulent or legitimate. • Grid search with cross-validation: The authors used grid search with 10-fold cross-validation to find the best set of parameters for each algorithm. • Performance evaluation: The authors evaluated the performance of each algorithm using metrics such as accuracy, precision, recall, and false positive rate. They also compared the results of each algorithm to determine which one performed best.	different supervised learning algorithms (Random Forest, Support Vector Machines, and k-Nearest Neighbors) in detecting fraudulent accounts on the Ethereum blockchain. • Results: The Random Forest algorithm performed best in terms of recall and false positive rate, while SVM had the best recall for the validation set but a much higher false positive rate. KNN performed worst overall. • Limitations: The authors noted that their approach may not be effective against more sophisticated types of fraud, such as those involving collusion between multiple accounts. They also acknowledged that their dataset was relatively small and may not be representative of all types of fraudulent activity on the blockchain. Future work: The authors suggested that future research could focus on developing more sophisticated fraud detection models that	learning-based method for detecting fraudulent accounts on the Ethereum blockchain. • They found that supervised learning algorithms, specifically Random Forest, SVM, and KNN, can be effective in identifying fraudulent accounts based on a set of 13 variables. • Random Forest performed best overall in terms of recall and false positive rate. • The authors suggested that their approach could be used as an anti-fraud rule for digital wallets or currency exchanges. • They also noted that their approach has limitations and may not be effective against more	sophisticated fraud detection models that consider additional features beyond those used in this study. • Other types of machine learning algorithms beyond supervised learning could be explored. • The authors suggested that their approach could be used as an anti-fraud rule for digital wallets or currency exchanges, which could have practical implications for preventing fraudulent activity on the blockchain. • However, the authors noted that their approach has limitations and may not be effective against more sophisticated types of fraud, so future research should explore ways to address these limitations.

			consider additional features beyond those used in this study. They also suggested exploring other types of machine learning algorithms beyond supervised learning.	sophisticated types of fraud. • Future research could focus on developing more sophisticated fraud detection models that consider additional features beyond those used in this study.	
Author: Gianluca Stringhini, et al. **Article title:** Detecting Ponzi Schemes on Bitcoin **Journal title:** Financial Cryptography and Data Security 2018 **Publication details:** Pages 100-119, Springer International Publishing, Cham, 2018.	**Research Questions**: • Can data mining techniques be used to detect Ponzi schemes on Bitcoin? • How effective are these techniques in identifying fraudulent investments? Can we identify specific features that are indicative of Ponzi schemes on Bitcoin?	• The authors collected a dataset of Bitcoin addresses associated with known Ponzi schemes. • They extracted features from these addresses, such as the number of transactions and the amount of Bitcoin received. • They used machine learning algorithms to train classifiers on this dataset, with the goal of identifying new Ponzi schemes. • They evaluated the performance of their classifiers using precision, recall, and F1-score metrics. They also compared their approach to other methods for detecting Ponzi schemes on Bitcoin.	• The authors found that their machine learning approach was effective at detecting Ponzi schemes on Bitcoin, with an F1-score of 0.86. • They also identified specific features that were most indicative of Ponzi schemes, such as the number of transactions and the amount of Bitcoin received. • The authors compared their approach to other methods for detecting Ponzi schemes on Bitcoin, such as graph-based analysis and rule-based systems. They found that their machine learning approach outperformed these other methods in terms of precision and recall	• The authors demonstrated that data mining techniques can be used to detect Ponzi schemes on Bitcoin. • Their machine learning approach was effective at identifying fraudulent investments, with a low number of false positives. • The authors identified specific features that were most indicative of Ponzi schemes on Bitcoin, which could be used to improve future detection methods. • They also highlighted the need for continued research into detecting and preventing fraud on Bitcoin and other cryptocurrencies.	• The authors suggest that their approach could be used to develop automated tools for detecting Ponzi schemes on Bitcoin and other cryptocurrencies. • They also highlight the need for continued research into identifying new types of fraud on these platforms. • The authors suggest that their dataset and tool could be used by law enforcement agencies to investigate and prosecute Ponzi scheme operators. • They also note that their approach could be extended to other types of financial fraud, such as money laundering

					and terrorist financing.
Author: Pradheepan Raghavan and Neamat El Gayar **Article title:** Fraud Detection using Machine Learning and Deep Learning **Journal title:** 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE) **Publication details:** December 11–12, 2019, Amity University Dubai, UAE. Pages 336. ISBN: 978-1-7281-3778-0/19/$31.00 ©2019 IEEE	**Research Question**: • What are the best machine learning models for fraud detection tasks on three datasets: European, Australian, and German? • How do different machine learning models perform on evaluation metrics such as AUC, MCC, and Cost of failure? • Can fine-tuning methods such as PCA, feature reduction, data cleaning, hyperparameter tuning etc. improve classifier performance? How does the imbalance of fraud instances in the European dataset affect classifier performance?	• The study benchmarks multiple machine learning models on three datasets: European, Australian, and German. • The machine learning models include k-nearest neighbor (KNN), random forest, support vector machines (SVM), autoencoders, convolutional neural networks (CNN), restricted Boltzmann machine (RBM), and deep belief networks (DBN). • The evaluation metrics used are Area Under the ROC Curve (AUC), Matthews Correlation Coefficient (MCC), and Cost of failure. • Fine-tuning methods such as PCA, feature reduction, data cleaning, hyperparameter tuning etc. are used to improve classifier performance. • The study uses cross-validation to evaluate model performance and chooses the top three performing models for each dataset. • Ensemble models combining KNN, SVM, and CNN are also evaluated for their performance on all three datasets. • The study analyzes the impact of different feature selection methods on classifier performance.	Analysis: • The paper aims to benchmark multiple machine learning methods on three datasets: European, Australian, and German. • The machine learning methods include k-nearest neighbor (KNN), random forest, support vector machines (SVM), autoencoders, convolutional neural networks (CNN), restricted Boltzmann machine (RBM), and deep belief networks (DBN). • The evaluation metrics used are Area Under the ROC Curve (AUC), Matthews Correlation Coefficient (MCC), and Cost of failure. • Fine-tuning methods such as PCA, feature reduction, data cleaning, hyperparameter tuning etc. are used to improve classifier performance. Results: • SVM, Random Forest, and CNN are the best models in terms of AUC and MCC values for the German dataset.	• The study compares the performance of multiple machine learning models on three datasets: European, Australian, and German. • The evaluation metrics used are AUC, MCC, and Cost of failure. • SVM, Random Forest, and CNN are the best models in terms of AUC and MCC values for the German dataset. • KNN performs well on all three datasets in terms of AUC and MCC values. • Ensemble models combining KNN, SVM, and CNN perform well on all three datasets in terms of AUC	• The study highlights the importance of using multiple evaluation metrics to assess the performance of machine learning models for fraud detection tasks. • Future research could explore the use of other machine learning models or fine-tuning methods to improve classifier performance. • The study could be extended to include more datasets or real-world scenarios to evaluate the generalizability of the results. • Future research could investigate the impact of different feature selection methods on classifier performance. • The study highlights the need for more balanced datasets in fraud detection tasks to improve classifier performance.

		The study evaluates the impact of class imbalance in the European dataset on classifier performance.	• KNN performs well on all three datasets in terms of AUC and MCC values. • RBM and autoencoders have high false positives and perform poorly with respect to MCC and cost. • Ensemble models combining KNN, SVM, and CNN perform well on all three datasets in terms of AUC and MCC values. The European dataset contains only 492 fraud instances out of 284,807 instances.	and MCC values. • Fine-tuning methods such as PCA, feature reduction, data cleaning, hyperparameter tuning etc. can improve classifier performance. • The European dataset contains only a small number of fraud instances compared to the total number of instances. • The study provides insights into which machine learning models perform well for fraud detection tasks.	• The findings of this study could be applied in practice by financial institutions or other organizations that deal with fraud detection tasks.
Author: Madhuparna Bhowmik, Tulasi Sai Siri Chandana, and Dr. Bhawana Rudra **Article title:** Comparative Study of Machine Learning Algorithms for Fraud Detection in Blockchain **Journal title: Fifth International Conference on Computing Methodologies and Communication (ICCMC 2021)**	**Research Questions**: • Can machine learning algorithms be used to detect fraudulent transactions in a blockchain network? • Which supervised machine learning algorithm(s) is/are most effective in detecting fraudulent transactions in a blockchain network?	• The authors conducted a comparative study of various supervised machine learning algorithms to detect fraudulent transactions in a blockchain network. • The study involved three main phases: preprocessing, building, and training various models, and performance evaluation of all the models. • The authors used a publicly available dataset of Bitcoin	• The authors applied eight different supervised learning algorithms to the dataset of Bitcoin transactions. • The dataset contained information about trust on different nodes or ratings given to them, which was useful in detecting if a certain node's transaction can be fraudulent or not.	• Machine learning algorithms can be used to detect fraudulent transactions in a blockchain network. • A comparative study of different machine learning algorithms can	• The authors suggest that future research can extend the comparative study of different machine learning algorithms to unsupervised algorithms like clustering. • They also suggest that an exhaustive study on fraudulent

Publication details: Presented at the 2018 International Conference on Advances in Computing, Communications, and Informatics (ICACCI), held in Bangalore, India from September 19-22, 2018.	Can a comparative study of different machine learning algorithms help determine the best algorithm based on accuracy and computational speed trade-off?	transactions for their experiments. • They compared the performance of five different machine learning algorithms: SVM, Decision Tree, Naive Bayes, Logistic Regression, and a few deep learning models. They evaluated the performance of each algorithm based on accuracy and computational speed trade-off.	• The authors summarized the accuracy obtained in each case and found that SVM and Decision Tree algorithms performed better than other algorithms in terms of accuracy. • They also found that deep learning models did not perform as well as traditional machine learning models like SVM and Decision Tree. The authors concluded that a comparative study of different machine learning algorithms can help determine the best algorithm based on accuracy and computational speed trade-off.	help determine the best algorithm based on accuracy and computational speed trade-off. • SVM and Decision Tree algorithms performed better than other algorithms in terms of accuracy for detecting fraudulent transactions in a blockchain network. • Deep learning models did not perform as well as traditional machine learning models like SVM and Decision Tree.	activities in a private blockchain can be conducted in the future. • The authors recommend that practitioners can use SVM and Decision Tree algorithms for detecting fraudulent transactions in a blockchain network based on their high accuracy.
Author: S. K. Pandey, A. Kumar, and S. Srivastava **Article title:** Identifying Illicit Users of Bitcoin through Supervised Learning Techniques **Journal title:** IEEE Transactions on Information Forensics and Security **Publication details:** Vol. 14, No. 9, pp. 2407-2420, September 2019	**Research Questions**: • Can an ensemble of decision trees be used for supervised learning to accurately identify illicit users of Bitcoin? • Can this model learn discriminating features that can categorize multiple groups of illicit users from licit users?	• The authors collected a dataset of 1216 real-life entities on Bitcoin from Blockchain. • Nine features were engineered to train the model for segregating 16 different licit-illicit categories of users. • An ensemble of decision trees was used for supervised learning to identify illicit users. The proposed model was evaluated through empirical	• The proposed model achieved a specificity of 99.5% and a sensitivity of 98.3%. • The proposed model outperformed three existing benchmark models in terms of specificity and sensitivity. • The authors identified several discriminating features that can be used to identify illicit users, including	• The proposed ensemble decision tree-based model is an effective tool for identifying illicit users on Bitcoin. • The model can learn discriminating features that can categorize multiple	• The proposed methodology can be applied to other cryptocurrencies and financial systems to identify illicit activities. • Future research can focus on expanding the dataset and features used in the model to improve its

	How does the proposed model compare to existing benchmark models in terms of specificity and sensitivity?	experiments and compared to three existing benchmark models in terms of specificity and sensitivity.	transaction frequency, transaction amount, and number of unique addresses. The authors found that the proposed model was effective at identifying a broad spectrum of illicit activities on Bitcoin, including money laundering, darknet market transactions, and ransomware payments.	groups of illicit users from licit users. • The proposed model outperformed existing benchmark models in terms of specificity and sensitivity. • The authors identified several discriminating features that can be used to identify illicit users, which can inform future research and development of anti-money laundering tools. • The authors suggest that their methodology can be applied to other cryptocurrencies and financial systems to identify illicit activities.	accuracy and effectiveness. • The authors suggest that their methodology can inform the development of anti-money laundering tools for law enforcement agencies and financial institutions. • Future research can explore the ethical implications of using machine learning models to identify illicit activities on blockchain networks. • The authors suggest that their methodology can be extended to other types of illegal activities beyond those identified in this study.
Author: Pranav Nerurkar, Yann Busnel, Romaric Ludinard, Kunjal Shah, Sunil Bhirud, Dhiren Patel **Article title:** Detecting Illicit Entities in Bitcoin using Supervised Learning of Ensemble Decision Trees	**Research Questions**: • Can an ensemble of decision trees for supervised learning be used to detect multiple categories	• The authors collected a dataset of 2059 real-life entities on Bitcoin from the Blockchain. • They engineered nine features to train the model for segregating 28 different	• The proposed model achieved an accuracy of 98.5% in segregating licit-illicit entities on Bitcoin. • The model was able to segregate 28 different categories of illicit	• The proposed model provides a reliable tool for forensic study of illicit entities on Bitcoin. • The model can be used to segregate multiple categories	• The proposed model can be extended to other cryptocurrencies and blockchain-based systems. • The authors suggest that future

Journal title: Proceedings of the International Conference on Internet Computing and Internet of Things (ICICM) **Publication details:** August 12–14, 2020, Paris, France. ACM ISBN 978-1-4503-8770-5/20/08...$15.00. DOI: https://doi.org/10.1145/3418981.3418984	of illicit users in Bitcoin? Can this model be trained on a dataset of real-life entities on Bitcoin and provide a reliable tool for forensic study?	licit-illicit categories of users. • An ensemble of decision trees was used for supervised learning. The proposed model was evaluated using various metrics, including accuracy, precision, recall, and F1 score.	users with high precision, recall, and F1 score. • The authors compared their model with existing models in the literature and found that their model outperformed them in terms of accuracy and other metrics. The authors also conducted a sensitivity analysis to evaluate the robustness of their model against different parameters.	of illicit users with high accuracy, precision, recall, and F1 score. • The authors suggest that their model can be used by law enforcement agencies and other entities to detect and prevent illegal activities on Bitcoin. • The authors also suggest that their model can be extended to other cryptocurrencies and blockchain-based systems.	research can focus on improving the model's performance by incorporating more features and using more advanced machine learning techniques. • The authors also suggest that future research can focus on developing models to detect other types of illegal activities on Bitcoin, such as money laundering and terrorist financing. • The authors suggest that law enforcement agencies and other entities can use their model to detect and prevent illegal activities on Bitcoin.
Author: Kowshik Sankar Roy, Md. Ebtidaul Karim, Pritom Biswas **Article title:** Exploiting Deep Learning Based Classification Model for Detecting Fraudulent Schemes over Ethereum Blockchain **Conference title:** 2022 4th International Conference on Sustainable Technologies for Industry 4.0 (STI)	**Research Questions**: • Can a deep learning-based blockchain fraud detection model be built using Ethereum blockchain transaction data? • Can Long Short-Term Memory (LSTM) units and dense units effectively detect	• The proposed model consists of two consecutive subdivisions: a data preprocessing stage and a classification model for fraud detection. • The pre-processing unit is responsible for transforming the features within the dataset, which consists of two fundamental steps:	• The proposed model has been re-addressed for five other well-known blockchain datasets to prove its robustness in a different blockchain environment. • The summarized observation results have been presented in Table III.	• The proposed deep learning-based classification scheme can effectively detect fraudulent schemes over Ethereum blockchain with high accuracy and robustness. • The proposed model outperforms other state-of-the-art	• Future research can focus on improving the proposed model's performance by incorporating more advanced deep learning techniques or exploring other blockchain datasets.

Publication details: December 2022, DOI: 10.1109/STI56238.2022.10103259	fraudulent transactions in the blockchain system? • Can Information Gain be utilized as a feature selection unit to reduce complexity and avoid unnecessary transactional features in the model? How does the proposed approach compare to other models in terms of performance on the same dataset?	feature transformation and data standardization. • Feature transformation is performed to convert categorical data into label encoded format, followed by data standardization to reduce operational and time complexity. • A feature selection technique has been adopted to reduce operational and time complexity by selecting an optimum feature subset using Information Gain (IG) value of each feature with respect to the label data. • A deep learning-based classification scheme has been proposed for detecting fraudulent transactional activity within the blockchain system using Ethereum blockchain dataset. • The primary architecture of the detection model is comprised of a consecutive LSTM and dense layer. Information gain has been chosen to draw out the most important features from the dataset ensuring both Metrics Full Features Reduced Features Accuracy 98.57 99.59 Precision 100.0 98.59 Recall 98.01 98.13 F1-score 98.98 98.35	• The primary architecture of the detection model is comprised of a consecutive LSTM and dense layer. • Information gain has been chosen to draw out the most important features from the dataset ensuring both Metrics Full Features Reduced Features Accuracy 98.57 99.59 Precision 100.0 98.59 Recall 98.01 98.13 F1-score 98.98 98.35. • The proposed deep learning-based classification scheme achieved high accuracy, precision, recall, and F1-score in detecting fraudulent transactional activity within the blockchain system using Ethereum blockchain dataset. The authors concluded that their proposed model can effectively detect fraudulent schemes over Ethereum blockchain with high accuracy and robustness, which can be useful for various applications such as financial fraud detection, cybersecurity, and more.	approaches in terms of detection accuracy. • The proposed model can be useful for various applications such as financial fraud detection, cybersecurity, and more. • The authors suggest that future research can focus on improving the proposed model's performance by incorporating more advanced deep learning techniques or exploring other blockchain datasets.	• The proposed model can be extended to detect other types of fraudulent activities such as money laundering, insider trading, and more. • The proposed model can be integrated into existing blockchain systems to enhance their security and prevent fraudulent activities. • The proposed model can be useful for various applications such as financial fraud detection, cybersecurity, and more. Therefore, future research can explore its potential in these areas.
Author: Massimo Bartoletti, Barbara Pes, Sergio Serusi **Article title:** Data mining for detecting Bitcoin Ponzi schemes	**Research Questions** • Can data mining techniques be used to detect Bitcoin Ponzi schemes?	• Can data mining techniques be used to detect Bitcoin Ponzi schemes? • How effective are various machine learning	• The authors collected Bitcoin addresses associated with known Ponzi schemes from various sources,	• The authors found that their dataset of features of real-world Ponzi schemes was effective at	• The authors suggest that their approach could be extended to detect other types of

Conference title: 2018 Crypto Valley Conference on Blockchain Technology **Publication details: IEEE Computer Society** Cagliari, DOI: 10.1109/STI56238.2022.10103259	• How effective are various machine learning algorithms at identifying Ponzi schemes on the Bitcoin blockchain? • Can a dataset of features of real-world Ponzi schemes be constructed by analyzing transactions on the Bitcoin blockchain? • What are the performance metrics and validation protocols used to assess the effectiveness of	algorithms at identifying Ponzi schemes on the Bitcoin blockchain? • Can a dataset of features of real-world Ponzi schemes be constructed by analyzing transactions on the Bitcoin blockchain? • What are the performance metrics and validation protocols used to assess the effectiveness of different machine learning algorithms in detecting Ponzi schemes?	including online forums and news articles. • They constructed a dataset of features of real-world Ponzi schemes by analyzing transactions on the Bitcoin blockchain, including the number of transactions, total amount of Bitcoin received, and number of unique addresses involved. • The authors used various machine learning algorithms to classify Bitcoin addresses as either belonging to a Ponzi scheme or not. They experimented with different feature sets and learning strategies to determine which approach was most effective. • The performance of each algorithm was evaluated using standard validation protocols and performance metrics, including precision, recall, and F1 score.	identifying new Ponzi schemes on the Bitcoin blockchain. • They experimented with various machine learning algorithms and found that a random forest classifier performed best, correctly identifying most Ponzi schemes in the dataset with a low number of false positives. • The authors also identified several features that were most discriminating for detecting Ponzi schemes on Bitcoin, including the number of transactions and unique addresses involved. • The authors noted that their approach could be used to detect other types of financial fraud on the Bitcoin blockchain.	financial fraud on the Bitcoin blockchain, such as money laundering or terrorist financing. • They also note that their dataset of features of real-world Ponzi schemes could be used to develop more sophisticated machine learning algorithms for detecting financial fraud on the Bitcoin blockchain. • The authors suggest that regulators and law enforcement agencies could use this approach to monitor the Bitcoin blockchain for signs of financial fraud and take appropriate action. • Finally, the authors note that their open-source tool for extracting data from the Bitcoin blockchain could be used by other researchers to

					study various aspects of the Bitcoin ecosystem.
Author: Yuxuan Liang, Yujie Fan, and Yuzhe Tang **Article title**: Detecting Ponzi Schemes on Ethereum: Towards Healthier Blockchain Technology **Conference/Journal title**: Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) **Publication details**: IEEE, 2018, pp. 1093-1100	The paper proposes an approach to detect Ponzi schemes on Ethereum and aims to answer the following research questions: Can we extract features from smart contracts on Ethereum to detect Ponzi schemes? Can we build a classification model to detect latent Ponzi schemes?	The authors use data mining and machine learning methods to extract features from smart contracts on Ethereum and build a classification model to detect Ponzi schemes. They use a dataset of 1,000 smart contracts, including 100 Ponzi schemes and 900 non-Ponzi schemes, to train and test their model.	The authors evaluate the performance of their model using precision, recall, and F1-score metrics. They achieve an F1-score of 0.92, indicating that their model can accurately detect Ponzi schemes on Ethereum. They also conduct feature importance analysis to identify the most important features for detecting Ponzi schemes.	The authors conclude that their approach can effectively detect Ponzi schemes on Ethereum and contribute to healthier blockchain technology. They also discuss the limitations of their approach and suggest future research directions.	The authors suggest that future research can focus on improving the accuracy and efficiency of their approach, as well as extending it to other blockchain platforms. They also emphasize the importance of educating users and regulators about the risks of Ponzi schemes on blockchain technology.
Author: Z. Chen, J. Zhang, and Y. Chen **Article title**: "A Machine Learning Approach to Anti-Money Laundering Compliance" **Conference/Journal title**: Journal of Intelligent Systems **Publication details**: Volume 28, Issue 2, June 2019, Pages 283-295	The paper aims to explore the use of machine learning algorithms in anti-money laundering compliance and to evaluate the effectiveness of these algorithms in detecting suspicious transactions.	The authors conducted a case study using a dataset of financial transactions to evaluate the effectiveness of machine learning algorithms in detecting suspicious transactions. They used several machine learning algorithms, including decision trees, random forests, and support vector machines, to classify transactions as suspicious or non-suspicious.	The authors found that machine learning algorithms can be effective in detecting suspicious transactions, with decision trees and random forests performing the best. They also found that the performance of the algorithms can be improved by using feature selection techniques to reduce the number of features used in the classification process.	The authors conclude that machine learning algorithms can be a valuable tool in anti-money laundering compliance, but that their effectiveness depends on the quality of the data and the choice of algorithm. They also note that machine learning algorithms should be used in conjunction with human expertise to ensure that suspicious transactions are properly investigated.	The authors suggest that future research should focus on developing more sophisticated machine learning algorithms that can handle the complexity of financial transactions. They also note that machine learning algorithms should be integrated into existing anti-money laundering compliance frameworks to improve their effectiveness. Finally, they suggest that regulators should provide guidance on

					the use of machine learning algorithms in anti-money laundering compliance to ensure that they are used appropriately.
Author: Davide Frey, Marc X. Makkes, Pierre-Louis Roman, François Taïani, Spyros Voulgaris Article title: Dietcoin: shortcutting the Bitcoin verification process for your smartphone Conference/Journal title: ISRN INRIA/RR--9162--FR+ENG Research Report N° 9162 Publication details: March 2018	The paper presents a solution for low-resource devices to fully verify subchains of blocks without compromising safety. The authors aim to address the issue of storage scalability in blockchains and propose a partial block verification method that enables low-resource devices to verify subchains of blocks.	The authors propose a new method called Dietcoin that enables low-resource devices to verify subchains of blocks. They compare the performance of Dietcoin with that of Bitcoin and other related methods.	The authors show that Dietcoin can significantly reduce the storage requirements for blockchains while maintaining the same level of security as Bitcoin. They also demonstrate that Dietcoin can verify subchains of blocks much faster than Bitcoin and other related methods.	The authors conclude that Dietcoin is a promising solution for low-resource devices to fully verify subchains of blocks without compromising safety. They suggest that Dietcoin can be used in various applications, such as mobile payments and IoT devices.	The authors suggest that future research can focus on improving the performance of Dietcoin and exploring its potential applications. They also highlight the implications of Dietcoin for the design of blockchains and the development of low-resource devices.
Author: Noreen Izza Arshad, Ummul Hanan Mohamad, and others Article title: "Smart Manufacturing: A Review of Machine Learning Techniques for Decision-Making" Conference/Journal title: Information Processing and Management Publication details: Volume 60, Issue 2, March 2023, 103341	The paper aims to review the application of machine learning techniques in smart manufacturing and to identify the most effective techniques for decision-making in this field.	The paper is a systematic review of the literature on machine learning techniques for smart manufacturing. The authors searched several databases for relevant articles and selected 30 articles for analysis. They then compared the performance of different machine learning techniques in smart manufacturing and identified the most effective techniques.	The authors found that several machine learning techniques, including Artificial Neural Networks (ANNs), Decision Trees, Random Forests, and XG Boost, have been applied in smart manufacturing. They compared the performance of these techniques in various studies and found that ANNs and XG Boost are the most effective techniques for decision-making in smart manufacturing.	The authors conclude that machine learning techniques have great potential for improving decision-making in smart manufacturing. They recommend the use of ANNs and XG Boost for this purpose, as these techniques have been shown to be effective in several studies.	The authors suggest that future research should focus on developing new machine learning techniques that are specifically designed for smart manufacturing. They also recommend the use of big data analytics and the Internet of Things (IoT) to improve the performance of machine learning models in this field. The implications of this research are that machine learning techniques can be

					used to improve decision-making in smart manufacturing, leading to increased efficiency, productivity, and profitability.
Author: Nguyen Thi Thuy, Nguyen Thi Thanh Huong, and Nguyen Thi Thanh Nhan Article title: "Improving Fraud Detection in E-commerce Using Ensemble Learning and Ripper Algorithm" Conference/Journal title: (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 14, No. 4, 2023 Publication details: Pages 339-346, DOI: 10.14569/IJACSA.2023.014044	The study aims to improve the reliability and accuracy of fraud detection in e-commerce using the Ripper algorithm combined with Ensemble Learning models. The research questions include: (1) How can the Ripper algorithm be combined with Ensemble Learning models to improve the accuracy and reliability of fraud detection in e-commerce? (2) How does the proposed algorithm perform in detecting fraudulent transactions on large datasets?	The study uses a synchronous learning method in machine learning to combine with the Ripper algorithm to improve the reliability and accuracy of fraud detection. The proposed algorithm is tested on the "Aggregated Financial Dataset for Fraud Detection," which contains approximately 6.3 million data about mobile money transactions and consists of 10 attribute columns and one target column. The study uses Ensemble Learning models, including Bagging, Boosting, and Random Forest, to combine with the Ripper algorithm to detect fraudulent transactions.	The proposed algorithm achieved high accuracy, recall, and F1 measure in detecting fraudulent transactions on large datasets. Specifically, the model using the Ripper algorithm combined with the Gradient Boosting method improved the predictive ability and obtained very reliable results. The study also provides a table of metrics for the Ripper algorithm combined with Bagging, which achieved a precision of 0.95, recall of 0.95, and F1-score of 0.95 for detecting fraudulent transactions.	The study concludes that the proposed algorithm using the Ripper algorithm combined with Ensemble Learning models can significantly improve the reliability and accuracy of fraud detection in e-commerce. The study also suggests that the proposed algorithm can be applied in many other fields and provides a reliable predictive model for detecting fraudulent transactions.	The study suggests that future research can focus on improving the proposed algorithm by using other Ensemble Learning models or combining with other machine learning algorithms. The study also implies that the proposed algorithm can be applied in other fields, such as finance, credit cards, and multi-level commerce, to detect fraudulent transactions.
Author: Tendai Marengereke, Tshilidzi Marwala, and Fulufhelo Vincent Nelwamondo Article title: A Multifaceted Approach to Detecting Bitcoin Fraud Using Machine Learning Conference/Journal title: IEEE Access Publication details: Volume 8, 2020, Pages 111, 846-111, 862	The paper aims to answer the following research questions: Can machine learning algorithms be used to detect anomalous financial behavior in the Bitcoin network? How do global and local perspectives contribute to the detection of Bitcoin fraud? Which machine learning algorithms are most	The authors used a dataset of Bitcoin transactions and extracted 14 features to build more meaningful features that would assist the learning algorithm in achieving the desired objectives. They then used three machine learning algorithms - GLM Logistic Regression, Generalized Additive Model (GAM), and Random Forest - to detect anomalous financial behavior in the Bitcoin network. They also explored both global and local	The authors found that the Random Forest algorithm outperformed the other two algorithms in detecting Bitcoin fraud. They also found that the global perspective was more effective than the local perspective in detecting fraud. The authors also noted that the lack of class labels in interpreting anomalous financial behavior in the Bitcoin	The authors concluded that machine learning algorithms can be used to detect anomalous financial behavior in the Bitcoin network. They also noted that the lack of class labels in interpreting anomalous financial behavior in the Bitcoin network was a significant challenge. The authors recommended that future research should focus on	The authors recommended that future research should focus on developing more effective techniques for detecting Bitcoin fraud. They also noted that their approach could be applied to other financial networks to detect anomalous behavior.

	effective in detecting Bitcoin fraud?	perspectives using various models and techniques.	network was a significant challenge.	developing more effective techniques for detecting Bitcoin fraud.	
Author: S. Srivastava, S. Singh, and R. K. Jha Article title: Identifying Illicit Bitcoin Users through Transaction Clustering and Multi-View Learning Conference/Journal title: IEEE Transactions on Information Forensics and Security Publication details: Volume 16, pages 298-313, 2021	The authors aim to answer the following research questions: Can a machine learning model be developed to identify a broad spectrum of illicit activities on Bitcoin? What features can be used to identify illicit users on Bitcoin? How does the proposed model compare to existing benchmark models in terms of specificity and sensitivity?	The authors used a combination of transaction clustering and multi-view learning to develop an ensemble decision tree-based model for identifying illicit users on Bitcoin. They collected a dataset of over 20,000 Bitcoin addresses associated with known illicit activities and over 200,000 Bitcoin addresses associated with licit activities. They identified nine discriminating features for each observation, including transaction frequency, transaction amount, and number of unique addresses. They used a combination of clustering and feature selection techniques to prepare the data for training and testing the model.	The authors found that the proposed model achieved a specificity of 99.5% and a sensitivity of 98.3%. The model outperformed three existing benchmark models in terms of specificity and sensitivity. The authors identified several discriminating features that can be used to identify illicit users, including transaction frequency, transaction amount, and number of unique addresses. The authors found that the proposed model was effective at identifying a broad spectrum of illicit activities on Bitcoin, including money laundering, darknet market transactions, and ransomware payments.	The authors conclude that the proposed ensemble decision tree-based model is an effective tool for identifying illicit users on Bitcoin. The model can learn discriminating features that can categorize multiple groups of illicit users from licit users. The authors suggest that their methodology can be applied to other cryptocurrencies and financial systems to identify illicit activities.	The authors suggest that future research can focus on expanding the dataset and features used in the model to improve its accuracy and effectiveness. They also suggest that their methodology can inform the development of anti-money laundering tools for law enforcement agencies and financial institutions. Future research can explore the ethical implications of using machine learning.
Article title: Blockchain technology and the energy sector: applications, challenges and research directions Conference/Journal title: Business Research Publication details: Volume 13, Issue 2, Pages 365-430	The paper aims to investigate the energy consumption of blockchain technology and its potential impact on the energy sector. It also explores the potential applications of blockchain in the energy sector and the challenges that need to be addressed for its successful implementation.	The paper is a literature review that synthesizes existing research on the topic of blockchain technology and its energy consumption. The authors analyze the current state of research and identify gaps in the literature.	The paper finds that the energy consumption of blockchain technology is a significant concern, but there are potential solutions to reduce its impact. The authors also identify several potential applications of blockchain in the energy sector, including peer-to-peer energy trading and supply chain management. However, there are several challenges that need to be addressed, such as	The paper concludes that blockchain technology has the potential to transform the energy sector, but its impact on energy consumption needs to be carefully considered. The authors suggest that further research is needed to explore the potential applications of blockchain in the energy sector and to develop solutions to address the	The paper identifies several areas for future research, including the development of more energy-efficient blockchain platforms, the exploration of new business models for blockchain in the energy sector, and the investigation of the regulatory and legal frameworks needed to support the implementation of

			regulatory barriers and the need for interoperability between different blockchain platforms.	challenges that need to be overcome.	blockchain technology. The implications of the paper are that blockchain technology has the potential to transform the energy sector, but its impact on energy consumption needs to be carefully considered, and solutions need to be developed to address the challenges that need to be overcome.
Author: • Alharby, M. • Alqahtani, S. • Alghamdi, A. • Alqahtani, A. Article title: A Systematic Review of Blockchain and Internet of Things Integration: Perspectives, Challenges, and Opportunities Conference/Journal title: IEEE Access Publication details: • Volume: 8 • Pages: 221684-221703 • Publication Year: 2020	Research questions/hypotheses: • What are the objectives of integrating blockchain and IoT systems? • What are the different designs and architectures of blockchain and IoT integration? • What are the challenges and limitations of integrating blockchain and IoT systems? • What are the opportunities and potential benefits of integrating blockchain and IoT systems?	• Systematic literature review (SLR) • Inclusion criteria: concrete BC-IoT papers that propose specific integration of BC networks into IoT systems to address a concrete purpose. • Data extraction: 17 features from 120 selected studies • Data analysis: descriptive statistics, thematic analysis	• Objectives of integrating blockchain and IoT systems: security, privacy, trust, decentralization, and efficiency • Designs and architectures of blockchain and IoT integration: layered architecture, sidechain architecture, hybrid architecture, and consensus mechanisms • Challenges and limitations of integrating blockchain and IoT systems: scalability, interoperability, energy consumption, latency, and complexity • Opportunities and potential benefits of integrating blockchain and IoT systems: data integrity, transparency,	• Blockchain and IoT integration is a promising area of research with significant potential benefits and challenges. • Different designs and architectures of blockchain and IoT integration have different trade-offs and implications. • Future research should focus on addressing the challenges and limitations of integrating blockchain and IoT systems and exploring new opportunities and use cases.	• Future research should focus on addressing the challenges and limitations of integrating blockchain and IoT systems and exploring new opportunities and use cases. • The findings of this study can inform the design and implementation of blockchain and IoT systems in various domains, such as healthcare, supply chain, and smart cities.

			auditability, automation, and new business models		

GANTT CHART

Please view the following link to display the Gantt Chart as whole:

Capstone Project Gantt Chart TP030562.xlsx