Syrian Arab Republic
Syrian Private University
Faculty of Informatics Engineering
Department of Informatic Engineering
Software Engineering & Information Systems

AI – Enhanced Blockchain network for cryptocurrency Exchange

A Senior (2) Project Completed the requirements for obtaining a bachelor's degree in Informatics Engineering - Software Engineering and Information Systems

Prepared by:

Farah Fares – Ahmad Shekha

Supervised by:

Dr. Christine Zenieh

Eng. Rana Bu Hamdan

July 2024-2025

Supervised Certification

I certify that the preparation of this project entitled

[**AI-Enhanced blockchain network for Cryptocurrency Exchange**] prepared by Ahmad Shekha and Farah Fares, was made under my supervision at the Faculty of Computer, Informatics & Communication Engineering in partial fulfillment of the Requirements for the Degree of Bachelor of Software and Information System Engineering.

Name: Signature: Date:

Abstract

Blockchain technology has transformed financial transactions by removing the need for third parties or centralized authorities. However, despite its decentralized promise, blockchain networks—particularly those following the Bitcoin model—continue to exhibit vulnerabilities, especially within their consensus mechanisms. Meanwhile, artificial intelligence (AI) has emerged as a powerful tool across domains such as healthcare, science, and information technology, offering new opportunities for optimization and governance.

This project proposes an AI-enhanced framework to address the increasing centralization observed in blockchain networks. To facilitate full control and experimentation, we developed a blockchain system from scratch inspired by Bitcoin's architecture. This custom network allowed us to bypass transaction fees, enable direct protocol modifications, and conduct AI-driven experimentation.

As a foundational step, we conducted a comprehensive analysis of Bitcoin's decentralization trends over ten years, employing four established decentralization metrics to evaluate miner concentration and network fairness. Building on this analysis, we applied unsupervised AI techniques—including anomaly detection and clustering—to identify behavioral patterns associated with centralizing miner activity. These insights were translated into interpretable rules through decision tree learning, which were then integrated into the consensus logic of our network. The system was evaluated through simulated mining scenarios, demonstrating measurable improvements in decentralization without altering the core consensus mechanism.

The AI-generated rules significantly improved network fairness across all four-decentralization metrics. Notably, the Nakamoto Coefficient increased from 1 to 3, Shannon Entropy rose by 56%, and both Gini and HHI metrics showed marked reductions in inequality, proving the effectiveness of our approach in enhancing decentralization through interpretable, data-driven constraints.

الملخص

أحدثت تقنية البلوك تشـــين تحولًا جذريًا في عالم المعاملات المالية من خلال إلغاء الحاجة إلى الوســـطاء أو الســـلطات المركزية. ومع ذلك، لا تزال هذه التقنية تعاني من بعض الثغرات، خاصـــة تلك المتعلقة بآلية الإجماع المعتمدة داخل الشـبكة. في المقابل، شـهد الذكاء الاصـطناعي تطورًا سريعًا وأصـبح يُسـتخدم على نطاق واسـع لتحسين مجالات متعددة مثل الطب، والعلوم، والتكنولوجيا.

يهدف هذا المشروع إلى تقديم إطار جديد يعتمد على الذكاء الاصطناعي لمعالجة مشكلة المركزية المتزايدة داخل شـبكات البلوك تشـين. لتحقيق هذا الهدف، قمنا ببناء شـبكة بلوك تشـين من الصـفر، مسـتوحاة من بنية شـبكة البيتكوين، بهدف امتلاك الســـيطرة الكاملة على البيئة التجريبية، وتجاوز رســـوم المعاملات، وتمكين التعديلات المباشرة على بروتوكول الشبكة لدعم تكامل تقنيات الذكاء الاصطناعي.

في المرحلة التمهيدية من المشـــروع، أجرينا تحليلًا شـــاملًا لشـــبكة البيتكوين من خلال تطبيق أربعة مؤشرات رياضـــية لقياس درجة اللامركزية، معتمدين على بيانات تم جمعها من عام 2014 حتى 2024. بعد ذلك، استخدمنا خوارزميات غير خاضعة للإشراف للكشف عن أنماط السلوك التي ترتبط بظواهر التمركز، وتم تصنيف الكتل بناءً على نشـاط المعدنين. لاحقًا، تم اسـتخراج قواعد تفسـيرية باسـتخدام خوارزمية شـجرة القرار، ومن ثم دمج هذه القواعد داخل منطق الإجماع لشبكتنا الخاصة.

أظهرت نتائج المحاكاة تحســـنًا ملحوظًا في مؤشرات اللامركزية بعد تطبيق القواعد المســـتخرجة. على ســـبيل المثال، ارتفع معامل Nakamoto Coefficient من 1 إلى 3، وازدادت قيمة Shannon Entropy بنســـبة 56%، في حين انخفض كل من معامل Gini Coefficient ومؤشر HHI)) بشـــكل واضـــح. تؤكد هذه النتائج فعالية النهج المقترح في تعزيز اللامركزية من خلال استخدام قواعد واضحة ومبنية على تحليل بياني دقيق.

Table of Contents

Chapter 1 Introduction ***1***

1.1 Introduction *2*
1.3 Project's goal *2*
Chapter 2 Reference study ***3***

2.1 Introduction *4*
2.2 Key components and Basic concepts *4*
2.2.1 Blockchain Technology *4*
2.2.2 Consensus mechanism *5*
2.2.3 Smart contract: *7*
2.2.4 Artificial intelligence *7*
2.3 Reference Study and Related Work *8*
2.3.1 Bitcoin System: *8*
2.3.2 Related work *13*
2.4 Conclusion *15*
Chapter 3 Proposed System ***16***

3.1 Introduction *17*
3.2 Workflow and Project Phases: *17*
3.3 Blockchain Network Development *17*
3.3.1 Block structure *18*
3.3.2 Transaction structure *18*
3.3.3 Rules for replacing the chain: *18*
3.3.4 Rules for valid transactions: *18*
3.3.5 Network workflow and consensus mechanism: *19*
3.4 Identifying Centralization Risks and Proposed Solution: *19*
3.5 Data collection: *20*
3.6 Centralization Assessment: *21*
3.7 Feature Extraction: *23*
3.8 Model Preparation and Rule Extraction: *24*
3.8.1 Anomaly detection: *25*
3.8.2 Clustering: *25*
3.8.3 Decision Tree to extract interpretable rules: *26*
3.9 Implementation and Testing: *27*
3.9.1 Rule integration: *27*
3.9.2 Simulation design: *28*
3.9.3 Comparing strategy: *28*
3.10 Use case diagram *29*
3.11 Requirements analysis *29*
3.12 Use case specification *30*
3.13 UML Class Diagram: *32*
3.14 Sequence diagram *33*
3.15 Unit test list: *33*
3.16 Conclusion *36*
Chapter 4 Implementation ***37***

4.1 Introduction *38*
4.2 Tools and technologies: *38*
4.3 Blockchain Network Implementation and Testing: *38*
4.3.1 Main Structure: *38*

4.3.2 Main classes *39*
4.3.3 Unit Testing results: *40*
4.3.4 Integration test: *46*
4.4 Bitcoin Decentralization Analysis: *47*
4.4.1 Nakamoto Coefficient: *47*
4.4.2 Shannon Entropy: *48*
4.4.3 Gini coefficient: *48*
4.4.4 Herfindahl-Hirschman Index (HHI): *49*
4.5 AI Model Implementation: *50*
4.5.1 Data Preparation: *50*
4.5.2 Anomaly Detection Model Implementation: *51*
4.5.3 Clustering: *52*
4.5.4 Decision tree for interpretable rules: *55*
4.6 Rule Integration and Simulation Setup *58*
4.6.1 Rule Integration into Consensus Logic *59*
4.6.2 Simulation Design and Parameters *59*
4.6.3 Evaluation Metrics *60*
4.6.4 Comparative Analysis of Simulation Results *60*
4.7 conclusion *61*
Chapter 5 Conclusion ***62***
5.1 introduction *63*
5.2 Summary of Results *63*
5.3 Project Contributions *63*
5.4 Limitations *64*
5.6 Final Remarks *65*
Chapter 6 References ***66***

List of Tables:

Table 2.1 POW, POA comparison 6

Table 3.1 Features Description 24

Table 3.2 Create wallet specification 30

Table 3.3 Send Transaction 31

Table 3.4 Validation chain specification 31

Table 3.5 View balance specification 32

Table 3.6 Block unit test 34

Table 3.7 Blockchain unit test 34

Table 3.8 Transaction unit test 35

Table 3.9 Wallet unit test 35

Table 3.10 Transaction pool unit test 35

Table 4.1 Block test result 41

Table 4.2 Blockchain test result 43

Table 4.3 Transaction test result 44

Table 4.4 Wallet test result 45

Table 4.5 Transaction pool test result 45

Table 4.6 Integration test result 46

Table 4.7 threshold 0.2 and 0.3 data comparison 55

Table 4.8 Comparison between two sets of rules 57

Table 4.9 Summarize the simulation outcomes 60

List of Figures

FIGURE 2.1 MERKLETREE 9
FIGURE 2.2 BITCOIN BLOCK STRUCTURE. 10
FIGURE 2.3 BITCOIN TRANSACTION STRUCTURE 11
FIGURE 2.4 BITCOIN WORKFLOW 11
FIGURE 2.5 P2P EXPLAINATION 13
FIGURE 3.1 Centralization Risk 19
FIGURE 3.2 Isolation Forest 25
FIGURE 3.3 K-means 26
FIGURE 3.4 DECISION TREE 27
FIGURE 3.5 Use case diagram 29
FIGURE 3.6 UML class Diagram 32
FIGURE 3.7 Sequence Diagram 33
FIGURE 4.1 Main Structure 38
FIGURE 4.2 Block class 39
FIGURE 4.3 Transaction class 39
FIGURE 4.4 Wallet CLASS 40
FIGURE 4.5 Overall test results 40
FIGURE 4.6 Block test 41
FIGURE 4.7 Blockchain network test 42
FIGURE 4.8 Transaction test 43
FIGURE 4.9 Wallet Test 44
FIGURE 4.10 TRANSACTION Pool test 45
FIGURE 4.11 Integration test 46
FIGURE 4.12 Nakamoto Coefficient 47
FIGURE 4.13 Shannon Entropy 48

FIGURE 4.14 Gini Coefficient ……49

FIGURE 4.15 HHI ……49

FIGURE 4.16 0.3 threshold outlier detection ……51

FIGURE 4.17 0.2 threshold outlier detection ……52

FIGURE 4.18 0.3 threshold K-means ……53

FIGURE 4.19 0.2 threshold K-means ……53

FIGURE 4.20 0.2 threshold Decision tree ……55

FIGURE 4.21 0.3 threshold Decision tree ……56

FIGURE 4.22 Decentralization metrics for 2020 ……57

FIGURE 4.23 Decentralization metrics for 2024……58

List of Equations

FIGURE 3.1 Gini coefficient *21*

FIGURE 3.2 Nakamoto coefficient. *21*

FIGURE 3.3 HHI *22*

FIGURE 3.4 Shannon Entropy *22*

Table of abbreviations

Abbreviations	Definitions
BTC	Bitcoin
PoW	Proof of work
PoA	Proof of Authority
P2P	Peer to Peer
ETH	Ethereum
EVM	Ethereum Virtual Machine
Mempool	Memory pool
Txs	Transactions
AI	Artificial intelligence
ML	Machine learning
API	Application Programming Interface
JSON	JavaScript Object Notation
Dapp	Decentralized app
IF	Isolation forest
CPU	Central processing unit

Chapter 1
Introduction

1.1 Introduction

This chapter provides an overview of the project's background and its main objectives. The primary goal of this project is to address the growing challenges faced by blockchain networks, particularly the issue of centralization, which can negatively impact the security and transparency of the network. With the rapid development of artificial intelligence (AI) technologies, there are new opportunities to improve the performance of blockchain systems by analyzing data and identifying patterns that could help enhance decentralization. In this chapter, we will discuss the fundamental concept of blockchain, its underlying mechanisms, and the key challenges it faces in the context of centralization. Additionally, we will explore how AI can be leveraged to provide innovative solutions aimed at mitigating these challenges.

1.2 Problem definition

Through a comprehensive study of the Bitcoin system, we identified a critical centralization trend in mining. As shown by Romiti et al. (2019), three major pools—BTC.com, AntPool, and ViaBTC—jointly exceeded 50% of the network's hash rate, with ≤20 actors within each pool receiving over half of the payouts [21]. This undermines Bitcoin's decentralization and increases systemic risk.

1.3 Project's goal

The primary goal of this project is to investigate and address the centralization trend in Bitcoin mining through a controlled, AI-enhanced simulation framework. To achieve this, we set the following objectives:

1. Develop a custom blockchain network from scratch to ensure full control over nodes, mining logic, and consensus rules—serving as a flexible simulation platform.

2. Demonstrate Bitcoin's centralization trend using empirical data and recognized decentralization metrics, as supported by Romiti et al. (2019).

3. Apply AI techniques to analyze Bitcoin block and mining reward distributions to detect hidden patterns and key centralization drivers.

4. Integrate AI-driven enhancements into our custom network and simulate miner behavior under various rule sets to evaluate their effect on decentralization and network health.

Chapter 2
Reference study

2.1 Introduction

In this section, we will explore the fundamental aspects of the well-known blockchain platform, Bitcoin. We will break down its essential components and characteristics, providing an in-depth understanding of the network's operation and the transaction cycle from user involvement to the recording and storage of transactions in a block.

We will then conduct a thorough analysis of scientific literature across multiple domains relevant to our project. This includes research from both blockchain and artificial intelligence fields that can guide and support our methodology. By integrating insights from these areas, we aim to strengthen the foundation and effectiveness of our proposed solutions.

This section will be the foundation for designing our system and our Ai solutions.

2.2 Key components and Basic concepts

2.2.1 Blockchain Technology

We will conduct an in-depth study of the main components of blockchain to help us build our blockchain network from scratch.

Blockchain technology: it is core to peer-to-peer distributed ledger that is cryptographically secure, append-on immutable (extremely hard to change), and updateable only via consensus of agreement among peers. [4]

Distributed ledger: This ledger is distributed among its participants and spread across multiple sites or organizations. This type can either be private or public. [3]

Consensus mechanism: A consensus mechanism is a set of steps are taken by all, or most nodes in order to agree on a proposed state or value. of the ledger. [4]

Block: A block is composed of multiple transactions and some other elements such as the previous hash (hash pointer), timestamp, and more. [4]

Cryptographic Hashes: Each block in the chain is linked to the previous one through a reference which is the hash of the previous block, hash functions ensure the integrity of data and provide a unique identifier for each block because for any different input for hash function we get a new and different output. [1]

Transaction: A transaction is the fundamental unit of a blockchain. A transaction represents a transfer of value from one address to another. [4]

Smart contract: These programs run on top of blockchain and encapsulate business logic to be executed when certain condition is met. The smart contract feature is not available in all blockchains but is now becoming a very desirable feature due to the flexibility and power it provides to the blockchain applications. [1]

Public and Private key: Public and private key pairs are used for secure transactions and user authentication, the public key is shared openly, while the private key is kept secret and used for digital signatures. [4]

Digital signatures: A cryptographic mechanism that leverages a user's private key to generate a unique and mathematically linked signature for a specific set of data, such as a transaction. This signature, when combined with the user's public key, verifies the authenticity of the sender and ensures that the associated data has not been tampered with during transmission. [4]

Private blockchain network: Permissioned blockchains restrict access, and participants are typically known and verified. [4]

Public blockchain network: Permissionless blockchains, like Bitcoin, are open to anyone, and participants can join and leave the network freely. [4]

Node: Nodes are individual computers that participate in the blockchain network, they validate transactions, maintain a copy of the blockchain, and contribute to the consensus process. [4]

Decentralization: This is a core of concept and benefit of blockchain. There is no need for a trusted third party or intermediary to validate transaction; instead, a consensus mechanism is used to agree on the validity of transactions.[18]

Immutability: Once the data has been written to the blockchain, it is extremely difficult to change it back. It is not truly immutable but, due to the fact that changing data is extremely difficult and almost impossible this is seen as a benefit to maintaining an immutable ledger of transactions. [18]

Transparency: As blockchains are shared and everyone can see what is on the blockchain, this allows the system to be transparent and as a result is established. This is more relevant is scenarios such as the disbursement of funds or benefits where personal discretion should be restricted. [19]

2.2.2 Consensus mechanism

Consensus mechanisms are a set of rules and protocols that ensure all nodes in a blockchain network agree on the validity of blocks, the order of transactions, and the distributed ledger state. Different blockchain networks use different consensus mechanisms, such as Proof-of-Work (PoW) and Proof-of-

Authority (PoA). The following table provides an in-depth analysis of PoW and PoA, highlighting their respective characteristics and making it easy to compare

Table 2.1 POW, POA comparison

Consensus mechanism	POW	POA
Mechanism	Miners try to Solve Puzzles to add block	Selected nodes takes authority of adding block
scalabilty	Less scalability	More scalability
Energy Efficiency	Energy intencive	Most energy efficient
Decentralized	Most decentralized no entity controls the	More centerlized some nodes control system
Cost	high	low
Usage	Public blockchain network	Private blockchain network
Security	Better	worse
Rewards	Yes	NO

Based on previous comparisons:

It is clear that the main problem with Proof of Work (PoW) is its high energy consumption. This is because miners need to perform extensive computational work to solve the puzzle required to generate a hash that meets the required difficulty level. Additionally, PoW is less scalable than other mechanisms due to the time it takes to solve the puzzle.

However, PoW is the most decentralized and secure mechanism. This is because miners are incentivized to maintain the system's integrity by earning rewards for mining valid blocks, and losing the computational work they spent on invalid blocks.

On the other hand, Proof of Authority (PoA) is more energy-efficient and scalable as no mining or puzzle-solving is required to add a block. Only trusted nodes have the authority to add blocks.

However, this makes PoA less decentralized, as some nodes control the system and can add blocks whenever they wish. This leads to another problem with PoA, which is that it is less secure than other mechanisms.

Therefore, finding a mechanism that is energy-efficient and scalable like PoA, and more decentralized like PoW, would result in the perfect mechanism. This project aims to propose its own mechanism in the upcoming sections.

2.2.3 Smart contract:

It is important to understand the main principles of smart contracts because we will be implementing a smart contract in a way that achieves all of its principles.

Smart contracts are digital agreements that are self-executing and encoded in computer programs. They utilize blockchain technology to automate, verify, and enforce the terms of a contract, making transactions more secure and trustworthy. Ethereum, a blockchain platform, is known for pioneering smart contracts. Unlike Bitcoin, Ethereum offers a versatile platform where developers can create and deploy smart contracts. These contracts have several benefits, including automation of actions based on predefined conditions, elimination of intermediaries, and immutability, which means that once deployed, smart contracts cannot be changed. [1]

Ethereum follows a simple principle to execute smart contracts. It converts the Solidity programming language, which is one of the most famous smart contracts languages, into bytecode. This bytecode is a string of hexadecimal code that is published on the Ethereum network and assigned a unique address. This process is called deploying. When a user wants to run a smart contract, they send a transaction to the bytecode address. After validating the transaction, each node in the network runs the bytecode using its Ethereum Virtual Machine (EVM) and implements the code on the current state of the network that each node has.[6]

2.2.4 Artificial intelligence

We will briefly explain AI because we plan to use it in our project to address the weaknesses in blockchain technology**.**

Artificial Intelligence (AI): Refers to the simulation of human intelligence in machines, enabling them to perform tasks that typically require human intelligence. This encompasses various capabilities, including learning, problem-solving, perception, and language understanding. [12]

Machine Learning (ML): Is an AI subset that enables systems to learn and improve from experience without being explicitly programmed. ML algorithms can analyze patterns and make predictions, enhancing blockchain data analysis. [16]

AI model: Refers to a computational system designed to simulate intelligent decision-making processes. It is crafted to analyze data, recognize patterns, and make informed predictions or decisions without explicit programming.[16]

Fraud detection: the use of one's occupation for personal enrichment through the deliberate misuse or application of the employing organization's resources or assets [22]

Anomaly detection: the identification of unique items, observations or events that elevate suspicion by being significantly different from the majority of the data [22]

Unsupervised Learning: Unsupervised learning is an important class of learning algorithms that helps uncover the inherent structure of the data without any external supervision (as in supervised learning). [20]

Clustering: Clustering is an unsupervised learning technique that groups similar data points into clusters based on their features. [24]

Outlier: An outlier is defined as an observation that "appears" to be inconsistent with other observations in the data set. An outlier has a low probability that it originates from the same statistical distribution as the other observations in the dataset. [23]

Decision Tree: A decision tree is a machine learning algorithm used for classification and regression, where data is split based on sequential decision rules. [14]

2.3 Reference Study and Related Work

2.3.1 Bitcoin System:

2.3.1.1 Bitcoin Overview [1] [4]

Bitcoin, introduced in a 2008 whitepaper by the pseudonymous Satoshi Nakamoto, marked the inception of blockchain technology. It is a decentralized, peer-to-peer digital currency that operates without a central authority. Bitcoin transactions are recorded on a public ledger known as the blockchain, maintained by a distributed network of nodes.

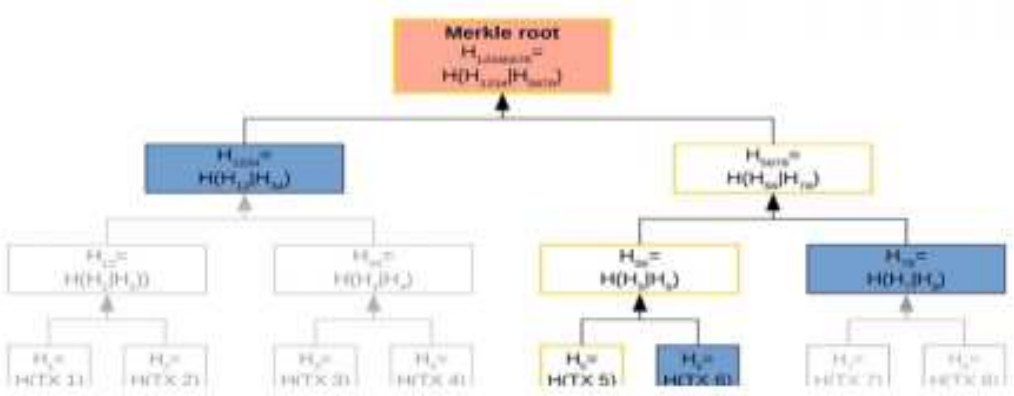
Bitcoin's impact extends beyond finance, influencing discussions on the future of money, privacy, and decentralization. It has sparked innovations in blockchain technology, inspiring the development of numerous alternative cryptocurrencies and blockchain projects.

2.3.1.2 Bitcoin block structure

Timestamp: The timestamp is a field in the block header that represents the approximate time when the block was mined. [1]

Nonce: The nonce is a 32-bit field in the block header used in the mining process. Miners repeatedly change the nonce in an attempt to find a hash that meets the current difficulty criteria. It's a variable to adjust the output hash.[1]

Merkle Root Hash: The Merkle Root is a hash of all the transaction hashes within the block. It is a crucial part of the block header and ensures the integrity of all transactions in the block. If any transaction is altered, the Merkle Root changes.[1]



PrevBlockHash: short for Previous Block Hash, is a 256-bit hash that uniquely identifies the preceding block in the blockchain. It ensures the chronological order and integrity of the blockchain.[2]

Bits: The "bits" field is a 32-bit value in the block header that represents the current target difficulty for mining a new block. It is a compact representation of the target threshold that the block's hash must meet for it to be considered valid.[2]

Transaction: Transactions represent the fundamental units of data in a block. They include information about the transfer of bitcoins from one address to another. A block contains multiple transactions.[1]

The following figure summarizes the Bitcoin block structure:

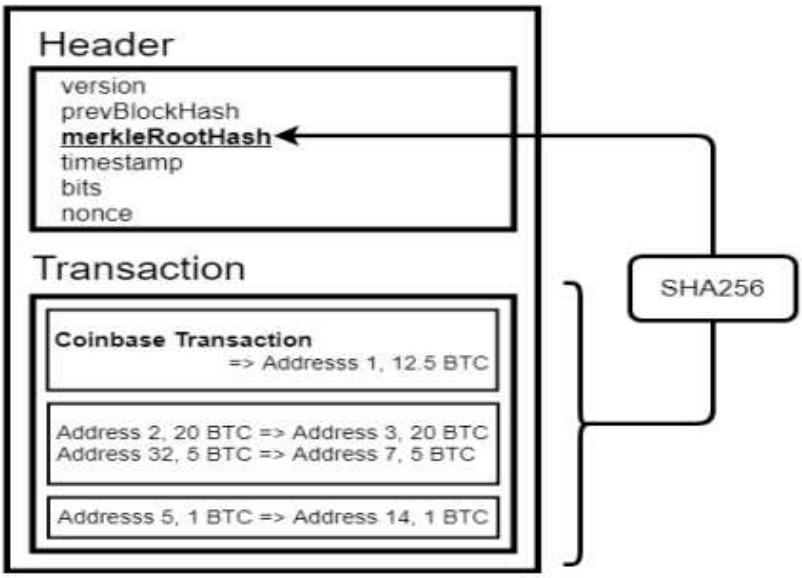


Figure 2.2 Bitcoin Block Structure

2.3.1.3 Bitcoin transaction's structure

Transaction Identifier (TxID): A unique identifier for the transaction, generated by hashing the transaction data. The TxID is used to reference and locate a specific transaction in the blockchain. [1]

Input: Each transaction consumes one or more Unspent Transaction Outputs (UTXOs) from previous transactions. The input contains references to these UTXOs, unlocking them for use in the current transaction.[1]

Output: The output specifies the recipients of the transferred value. It includes the amount of Bitcoin being sent and the recipient's address. Outputs that are not spent in subsequent transactions become UTXOs available for future transactions.[1]

Scripts and Signatures: Bitcoin transactions use scripts, often written in a scripting language called Script. These scripts define the conditions under which the recipient can spend the received funds. It's conceded of scriptSig and scriptPubKey, scriptSig prove that the sender is owner of previous transaction output, and scriptPubKey is a condition that receiver should meet in the future to spend the utxos he received [1],[4]

Amount: value of bitcoin sender wants to send, and it should be equal or less than user referenced bitcoin in the input.[1]

The following figure represents the bitcoin transaction structure.

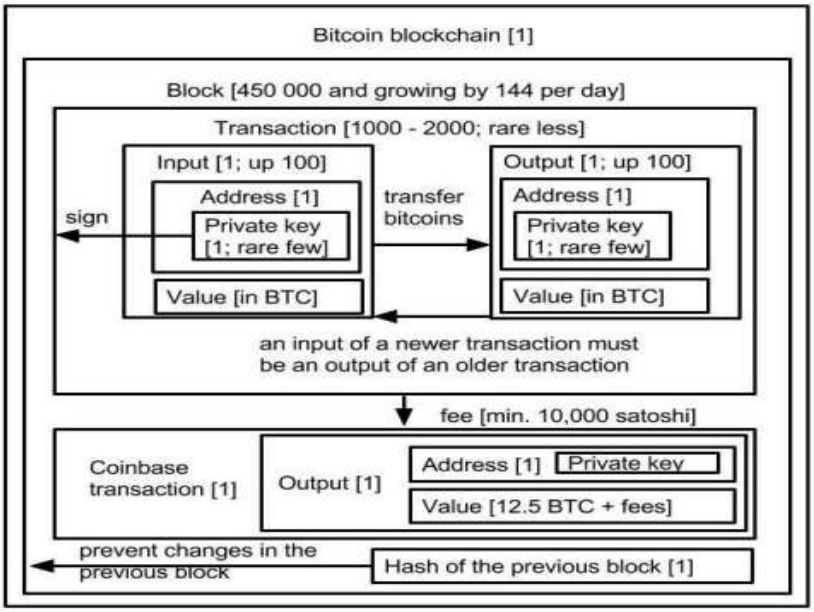


Figure 2.3 Bitcoin Transaction Structure

2.3.1.4 Bitcoin network workflow: [1], [4].

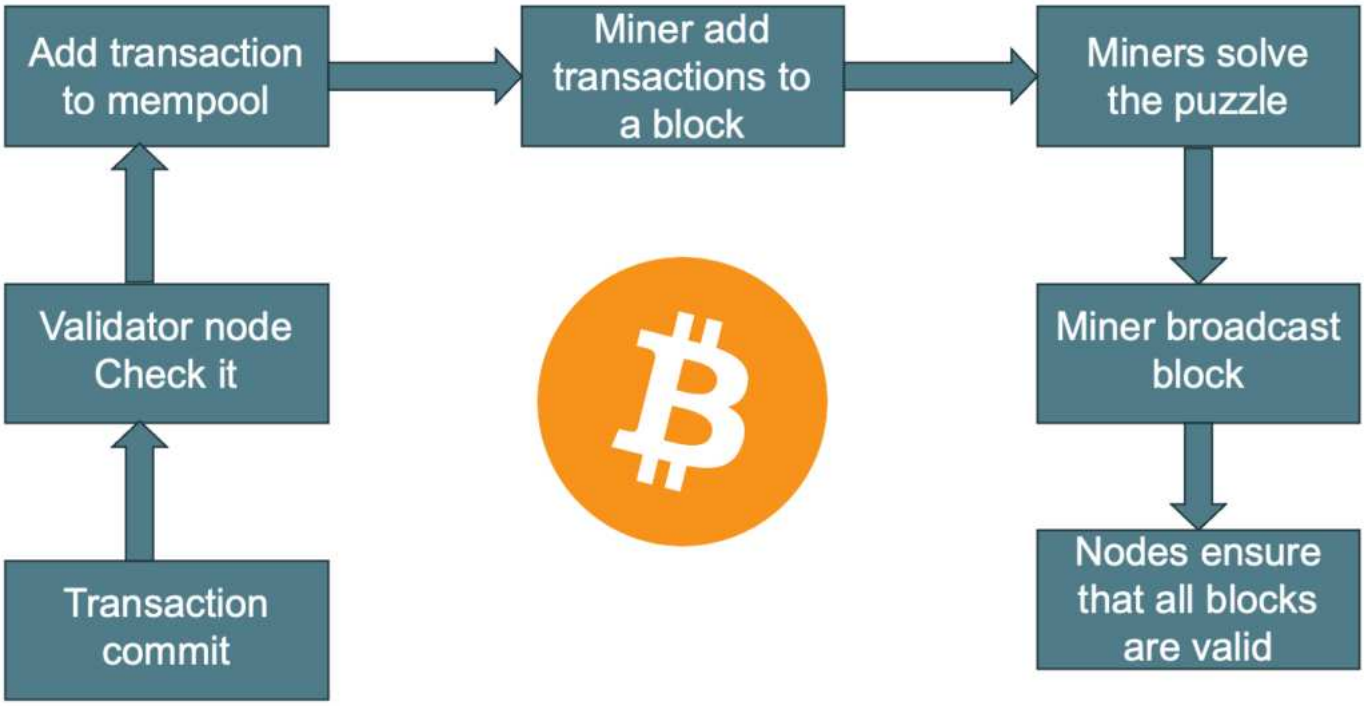


Figure 2.4 Bitcoin Workflow

Let’s get deep in each step:

Transaction Commit: Users initiate transactions, providing details such as inputs, outputs, and amounts using Bitcoin wallet software.

Validator node check: Validators in the network examine the syntax of the transactions to ensure they adhere to the prescribed format.

Mempool Addition: Valid transactions are added to the mempool, a pool of unconfirmed transactions maintained by nodes. This mempool serves as a temporary storage for transactions awaiting confirmation.

Miners add TXs to block: Miners select transactions from the mempool to include in a new block. This selection is strategic, often based on maximizing transaction fees.

Miners solve puzzles (PoW): Miners engage in the Proof of Work process, solving a computationally intensive puzzle by incrementing the nonce until a valid block hash is found. This energy-intensive task adds a new block to the blockchain and secures the network.

Block Broadcasting: The successful miner broadcasts the newly mined block, containing validated transactions, to the entire network. This broadcasting ensures that all nodes receive the latest updates.

Block Validation: Nodes in the network validate the received block, ensuring it adheres to consensus rules and that the included transactions are indeed valid.

2.3.1.5 Bitcoin smart contract: [1], [4]

Smart contracts are contracts with defined conditions that are directly written into code. They are self-executing and are often associated with platforms like Ethereum. There is a common misconception that Bitcoin does not support smart contracts, which is not entirely accurate.

Bitcoin actually implements smart contract functionalities through its scripting language. This is particularly evident in Pay-to-Public-Key (P2PK) transactions. Pay-to-Public-Key (P2PK) is a specific type of ScriptPubKey that is used to lock bitcoin to a public key. This means that the bitcoin can only be spent by the person who owns the private key that corresponds to the public key used in the script.

For instance, if Alice sends 1 BTC to Bob in a P2PK transaction, she includes Bob's public key in the transaction. When Bob wants to spend the bitcoin he received from Alice, he must sign his transaction with the private key that corresponds to the public key Alice included in her transaction. In simpler terms, when someone sends a coin, they should provide a digital signature to prove that they own it.

They should also lock the sent coin with the recipient's public key so that only the recipient can spend it by providing their own digital signature. All of this information will be recorded in the transaction.

When a new transaction is received, each node will validate the signature to ensure that it is valid. If it is, the money will be transferred from the sender to the recipient on the node's ledger.

This figure explains the process of P2PK in the transaction:

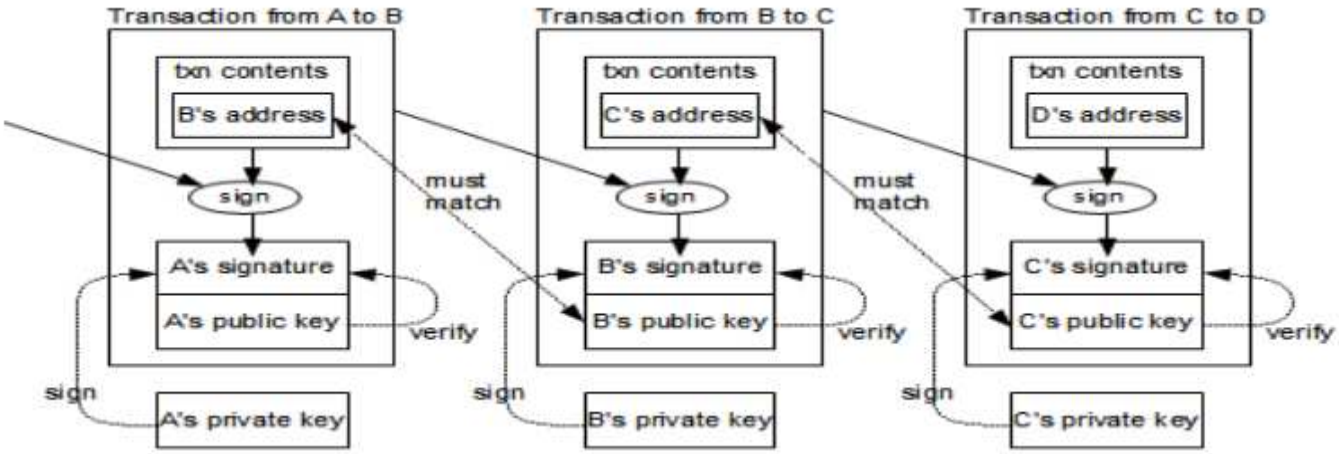


Figure 2.5 P2P explanation

2.3.1.6 Calculate user balance

Bitcoin maintains a record of all the transactions associated with a user's address and calculates their current balance by adding up all the unspent transaction outputs (UTXOs) that belong to them.

UTXOs are basically individual units of value that have been received by the user's address but have not been spent yet. These UTXOs are like pieces of coins that are held in reserve until they are needed for a new transaction.

By keeping track of all UTXOs, Bitcoin can accurately determine a user's current balance, taking into account both incoming and outgoing transactions.

2.3.2 Related work

A substantial body of research has been conducted on the blockchain industry, addressing diverse aspects ranging from data collection to decentralization analysis, network behavior, and simulation frameworks. This section outlines key studies relevant to our project, categorized by their thematic focus.

2.3.2.1 Data Collection

Effective blockchain analysis relies heavily on robust data collection mechanisms. Lin and Li [13] utilized Google BigQuery, a cloud-based platform that offers an interactive Application Programming Interface (API), to access public datasets of Bitcoin blocks. This service supports data retrieval from eight major Proof-of-Work (PoW) blockchains, enabling large-scale analytics and historical tracking. This data source forms the backbone of our blockchain data acquisition process.

2.3.2.2 Decentralization Metrics

Assessing decentralization in blockchain networks is a crucial research area. Lin and Li [24], as well as Grandjean and Heimbach [10], proposed a set of well-established metrics to quantify decentralization. These include:

- Shannon Entropy: A metric originating from information theory, proposed by Shannon in 1948, which measures the uncertainty or randomness in data distributions. In blockchain, it reflects how evenly mining power or block generation is distributed [13].

- Nakamoto Coefficient: Indicates the minimum number of entities that must be compromised to disrupt the consensus mechanism of the blockchain [10].

- Gini Coefficient: An economic inequality measure adapted to blockchain mining, where values range from 0 (perfect equality) to 1 (extreme inequality) in the distribution of mining power [10].

- Herfindahl-Hirschman Index (HHI): A concentration metric from economics used to assess centralization by summing the squares of market shares. In blockchain, a high HHI suggests domination by few mining pools, which poses a threat to decentralization [10].

2.3.2.3 Centralization Drivers in Mining

Several studies have analyzed factors contributing to centralization within the Bitcoin network. BEIKVERDI [5] and Zolotavkin [26] highlighted the emergence and growth of large mining pools as a key centralizing trend. These pools often accumulate disproportionate control over block generation, thereby reducing network diversity. Zolotavkin [26] also warned of short-term dominance spikes, where temporary surges in mining power could threaten consensus stability. Furthermore, Long [15] introduced the concept of the “previous block advantage ,” wherein miners who successfully mine one block gain a temporary edge in mining the next due to latency and network propagation benefits. These insights are fundamental to our feature extraction strategy for quantifying centralization risk.

2.3.2.4 Unsupervised Learning and Rule Extraction

To interpret blockchain behavior in an explainable manner, Deng and Wang [7] proposed

a framework that leverages unsupervised learning. Their approach involves using Isolation Forest for anomaly detection in blockchain activity, followed by decision tree algorithms to extract human-interpretable rules.

These rules can then be integrated into network monitoring systems to detect abnormal behaviors or deviations from expected patterns. Our project builds upon this technique

to enhance interpretability in decentralized environments.

2.3.2.5 Simulation and Testing Frameworks

Simulation is essential for validating hypotheses and observing the impact of new rules or configurations on a blockchain network. Babur and Shafiq [3] introduced a comprehensive simulation framework that models blockchain nodes and their interactions. They proposed three structured test cases to evaluate network performance under different rule sets.

This simulation model serves as a baseline for our comparative evaluation before and after applying modified rule sets.

2.4 Conclusion

As we conclude our study, we delved into the intricate workings of Bitcoin and learned about its fundamental concepts. We explored complex ideas such as consensus mechanisms and smart contracts, which have set the foundation for our very own project.

Then we explored multiple scientific studies that together will be the ground for our solutions.

Chapter 3
Proposed System

3.1 Introduction

This chapter presents the design and methodology of our proposed system, which aims to investigate and address centralization trends in the Bitcoin network using artificial intelligence techniques and a custom-built blockchain simulation platform. We begin by outlining the high-level system architecture, followed by a detailed explanation of each phase of the project. These include data collection, decentralization assessment, AI model training, and simulation of improvement strategies within our .own network

3.2 Workflow and Project Phases:

Our project is structured into a series of well-defined phases, each building upon the previous to address the centralization issue in Bitcoin. The phases are as follows:

1. Blockchain Network Development: Design and implement a blockchain network from scratch, modeled after Bitcoin. This custom network will serve as a controlled simulation environment for testing our proposed enhancements.

2. Identifying Centralization Risks and Proposed Solution: Conduct a detailed analysis of Bitcoin's operational workflow to identify the specific stages contributing to centralization.

3. Data Collection: Gather Bitcoin block data to be used in both statistical analysis and AI-based modeling.

4. Centralization Assessment: Apply established decentralization metrics to the collected data to empirically demonstrate Bitcoin's trend toward centralization.

5. Feature Extraction: Analyze the data to identify and extract key factors influencing decentralization. These features will be used in training AI models.

6. **Model Preparation and Rule Extraction:** Train an anomaly detection model to identify blocks or miner behaviors that may negatively impact network decentralization, and use a decision tree classifier to extract interpretable rules from these anomalies. These rules will inform protocol-level constraints aimed at mitigating centralization trends**.**

7. **Implementation and Testing:** Apply the extracted rules to our custom blockchain network to evaluate their effectiveness in enhancing decentralization.

In this chapter, each phase will be examined in depth, including the methodology, tools, and rationale behind our approach

3.3 Blockchain Network Development

To explore and validate our AI-driven enhancements, it was essential to create a blockchain network from the ground up. While Bitcoin served as our reference model, developing a custom network gave us full control over consensus rules, block structures, mining logic, and transaction handling. This

environment functions as a simulation platform where we can safely experiment with changes and measure their effects without limitations.

The implementation preserves the core mechanics of Bitcoin while enabling the integration of artificial intelligence techniques aimed at improving decentralization. The following subsections outline the architectural and technical foundations of this custom-built network.

3.3.1 Block structure

The block structure closely mirrors that of Bitcoin (as shown in **Figure 2.2**), with one key addition: a **Miner ID** field. This field allows us to track miner activity and analyze their behavior, which is essential for evaluating decentralization metrics and applying AI models effectively.

3.3.2 Transaction structure

The transaction structure implemented in our network follows the format previously illustrated in **Figure 2.3.** It retains Bitcoin's fundamental principles while ensuring compatibility with our rule-based verification logic and future AI enhancements.

3.3.3 Rules for replacing the chain:

Once a trusted node appends a new block to its chain, it broadcasts the updated chain to other nodes. To ensure network stability and prevent exploitation, we define the following rules for accepting and replacing chains:

1. The incoming chain must be longer than the current chain.

2. The incoming chain blocks must be connected with hashes successfully, so that each block stores the hash of the previous block.

3. The difference in difficulty between two blocks in a row should not be more than 1 to prevent difficulty malformations or alterations.

4. The incoming chain should not contain a block with a duplicated transaction.

5. The incoming chain should not contain a block with a miner's reward higher than 50. 6. The incoming chain should not have a block with an invalid transaction.

3.3.4 Rules for valid transactions:

A transaction in our network is considered valid if it meets all of the following conditions:

The amount being sent does not exceed the sender's balance. The total input value equals the total output value.

The sender provides a valid digital signature using their private key to prove ownership of the coins.

The transaction includes the recipient's public key, ensuring that only the intended recipient can later authorize spending of the coins via their own signature.

3.3.5 Network workflow and consensus mechanism:

Our network follows the same fundamental workflow as the Bitcoin system, as illustrated in **Figure 2.4**. This includes processes such as transaction validation, block formation, and block broadcasting across the network. However, to address the trend toward centralization, we enhance this standard workflow by incorporating a set of additional rules. These rules are not predefined but will be extracted through artificial intelligence models trained on Bitcoin block data. By integrating these AI-derived rules into the consensus mechanism, we aim to guide the network toward a more decentralized .behavior without altering the core structure of the consensus process itself

3.4 Identifying Centralization Risks and Proposed Solution:

Building on previous research that suggests Bitcoin is becoming increasingly centralized, this step focuses on analyzing Bitcoin's workflow to identify specific operational mechanisms that may be contributing to this trend, particularly in mining, where a critical centralization pattern has been identified by Romiti et al. [21].

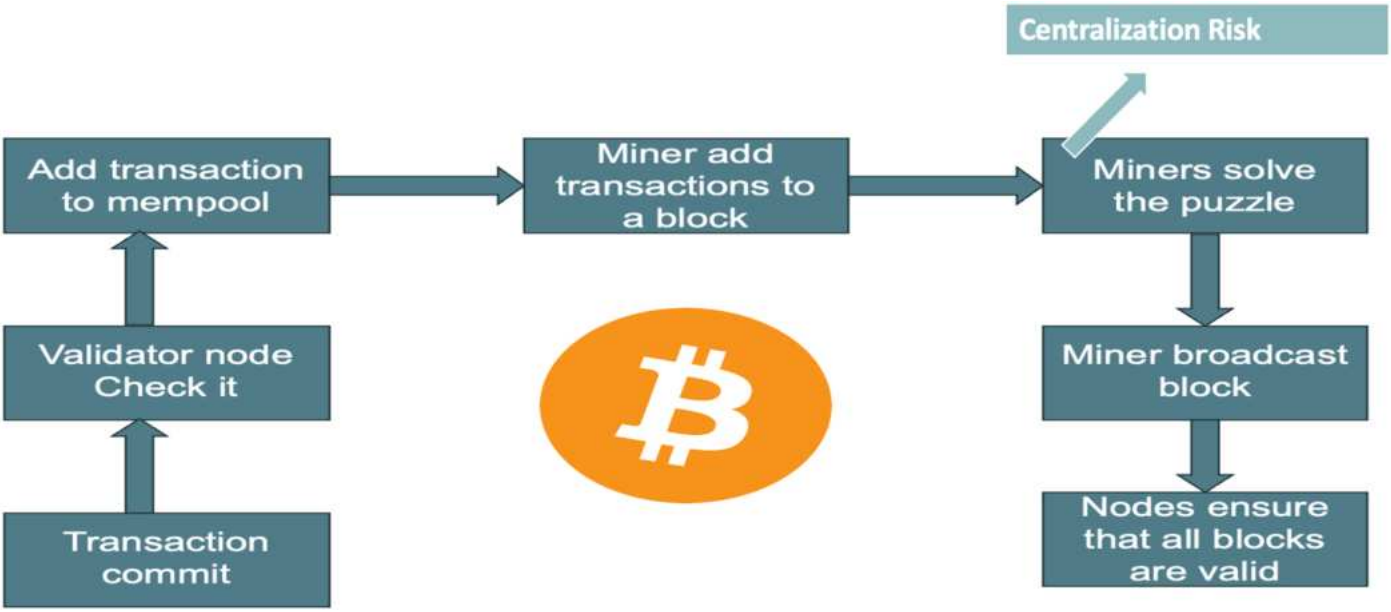


Figure 3.1 Centralization Risk

AI-Enhanced Consensus Mechanism, our analysis confirms that the centralization issue is rooted in the Proof-of-Work (PoW) consensus mechanism—the step where miners compete to solve a complex cryptographic puzzle to add the next block.

Over time, the computational resources required for PoW have become so demanding that only a few large mining pools dominate this process. This concentration of mining power undermines Bitcoin's original vision of decentralization and introduces systemic risk, where a handful of entities can influence the network disproportionately.

To address this, we explored Proof-of-Authority (PoA) as an alternative. Unlike PoW, PoA relies on a limited set of trusted nodes to validate new blocks, offering greater energy efficiency and easier governance. However, PoA comes with its own risks—mainly that it can introduce bias if trusted authorities are not carefully chosen or regulated.

Our proposed solution is to create a hybrid approach, using PoA principles while introducing AI-based oversight. Specifically, we use artificial intelligence to:

- Analyze mining behavior patterns in existing Bitcoin block data.
- Identify high-activity miners or mining pools contributing to centralization.
- Learn the rules or patterns these miners follow.
- Automatically generate constraints that limit how often these miners can add new blocks.

These AI-generated rules will then be integrated into our custom blockchain's consensus logic. By placing data-driven limits on dominant miners, our system aims to preserve decentralization without sacrificing the integrity of the consensus mechanism.

In short, AI serves as a watchdog, continuously monitoring network behavior and enforcing fair participation. This PoA-inspired, AI-enhanced model represents a novel approach to solving one of Bitcoin's most pressing structural issues**.**

3.5 Data collection:

We utilized the Bitcoin block dataset available in the public datasets section of Google BigQuery. This dataset is publicly accessible and is referenced in various studies. It contains records of all Bitcoin blocks from 2009 to the present. For our project, we focused on data from 2014 to 2024 to avoid potential misinterpretations of the early years. From each block, we extracted the following information: timestamp, difficulty, year the block was added, reward value, miner pool name, and the address that received the block reward. We have compiled data for 55929 Bitcoin blocks to be used in our study.

3.6 Centralization Assessment:

To empirically validate the trend of centralization in Bitcoin, we employed a set of established decentralization metrics to analyze the historical block data collected in Section 3.5. These metrics were chosen for their prominence in prior research and their capacity to quantify miner concentration, influence, and network inequality. By examining Bitcoin's block production over time through these metrics, we aim to provide evidence-based confirmation of the shift towards centralization and identify the peak of decentralization in Bitcoin's history as a foundation for further analysis. The following four metrics were utilized in our assessment:

Gini Coefficient: Used by [10] to measure decentralization in Ethereum PoS and Bitcoin PoW, this metric quantifies inequality in resource distribution (mining/staking power).

It is an inequality measure; mathematically, the Gini coefficient is calculated as follows:

$$G = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|x_i - x_j|}{2n^2\bar{x}}$$

Equation 3.1 Gini coefficient

where n is the number of entities, and x_i denotes the wealth of person i. Note that in the context of PoW, we take a miner's block publishing contribution as their wealth.

A low Gini coefficient then indicates that wealth is distributed equally among miners [10].

It ranges from:

- 0 = Perfect equality (all participants have equal power, more decentralization)
- 1 = Maximum inequality (single entity controls all resources, less decentralization)

Nakamoto Coefficient: Applied in [10] to assess blockchain resilience, this metric represents the number of independent entities needed to disrupt the blockchain [10]. For aPoW blockchain, generally ,more than 50% of the mining power is required to disrupt the network [10]. Mathematically, the Nakamoto Coefficient is calculated as follows:

$$N = min\{k \in [1, \cdots, K] : \sum_{i=1}^{k} p_i \geq 0.51\}$$

Equation 3.2 Nakamoto coefficient

- Where: Higher values indicate better decentralization.

Herfindahl-Hirschman Index (HHI): Utilized by [10] to evaluate market concentration. Mathematically, it is expressed as

$$HHI = \sum_{i=1}^{n} s_i^2,$$

Equation 3.3 HHI

where n is the number of entities in the market, and si denotes the market share of entity i as a fraction. This economic metric ranges:

- 0 = Perfect competition (many equal participants, more decentralization)
- 1 = Monopoly (single dominant entity, less decentralization)

Shannon Entropy: Employed by [13] to measure unpredictability in block producer selection, Shannon pointed out that information has redundancy, and the amount of redundancy depends on the distribution of probabilities, or uncertainty, within the information, which could be quantified by the notion of Shannon entropy [13] We adopt the Shannon entropy as the fourth measurement metric to measure the degree of randomness and disorder of the distribution of the number of blocks mined by miners.

Mathematically, the Shannon Entropy is calculated as follows:

$$E = -\sum_{i=1}^{n} p_i log_2 p_i$$

Equation 3.4 Shannon Entropy

where bi denotes the number of blocks generated by miners and n denotes the range of miners which entropy is computed for.

where:

- Higher values = More decentralized (power is distributed, more decentralization)
- Lower values = More centralized (few entities dominate, less decentralization)

These four metrics will be calculated annually from 2014 to 2024 to observe and analyze the long-term centralization trend within the Bitcoin network. To obtain a comprehensive and accurate picture of decentralization, we apply each metric in two distinct contexts:

Mining Pool Level – Grouping blocks by known mining pool names (e.g., AntPool, F2Pool, ViaBTC) to assess how much of the network is controlled by major institutional players.

Unique Miner Address Level – Grouping blocks based on the unique Bitcoin addresses that received the block rewards, which may reflect individual or sub-pool level mining behavior.

This dual analysis enables us to capture both the organizational and individual aspects of mining activity, providing a more nuanced understanding of how decentralization in Bitcoin is evolving. The results of these measurements will be presented and discussed in detail in the following chapter.

3.7 Feature Extraction:

Following the centralization assessment in the previous section, the next step is to extract meaningful features from the Bitcoin block data that can help identify patterns contributing to centralization. Feature extraction is a critical phase in the machine learning pipeline, as the quality and relevance of selected features directly affect the performance of AI models. In our project, we aim to isolate factors related to miner behavior, block difficulty, reward distribution, and temporal patterns that may signal a miner's disproportionate influence over the network. These extracted features will serve as input for the anomaly detection and rule extraction models in the following phases.

Several studies have analyzed factors contributing to centralization within the Bitcoin network. BEIKVERDI [5] and Zolotavkin [26] emphasized the rise of large mining pools as a significant driver of centralization. These pools accumulate vast computational resources and often control a substantial portion of block production. To quantify this effect, we include a **feature that measures the number of blocks mined per day by each pool**, allowing us to detect excessive activity that may indicate centralized power.

Zolotavkin [26] also highlighted the threat of short-term dominance spikes, where specific entities suddenly control block production for brief intervals, destabilizing consensus. To capture this behavior, we introduce a **feature that counts how many blocks each miner contributed within the last 10 blocks** of the chain, reflecting burst dominance.

In addition, long [15] introduced the "previous block advantage", a phenomenon in which miners who successfully mine a block may gain a temporary advantage due to reduced latency or better network positioning. This behavior is often associated with selfish mining strategies, which intentionally delay block propagation to gain a competitive edge. To track such patterns, we extract a **feature that detects consecutive blocks mined by the same entity,** as repeated success without network-wide competition can signal centralization risk.

Finally, to assess the diversity of miner participation, we include two additional features:

A binary feature indicating whether the miner **is unique within the last 10 blocks**, helping us understand how varied recent participation is.

A daily uniqueness flag, which checks whether a miner appears only once in all blocks mined during a single day, provides insight into whether mining activity is broadly distributed or dominated by a few entities each day.

This table summarizes all the features that will be used in the models:

Table 3.1 Features Description

Feature name	Description
blocks_per_pool_per_day	Counts the daily block contribution for miner till this block
blocks_per_pool_last_10_blocks	Counts the block contribution for miner from the last 10 blocks till this block
Is_consecutive_block	A binary feature that takes 1 if the block is a consecutive block for same miner pool
is_unique_miner_in last_24_hour	A binary feature takes 1 if is the first block for the miner in this day
is_unique_miner_in last_10_block	A binary feature takes 1 if is the first block for the miner from the last 10 blocks

These features, grounded in prior research and directly derived from historical block data, provide a robust foundation for identifying and modeling behaviors that contribute to centralization in the Bitcoin network.

3.8 Model Preparation and Rule Extraction:

In this phase, we prepare and train AI models to detect centralization-related anomalies in the Bitcoin network and extract interpretable rules that explain these behaviors. The process consists of two key steps:

(1) Identifying anomalous blocks or miner behaviors using unsupervised learning.
(2) Clustering Anomalies to Focus on Relevant Patterns.
(3) Training a decision tree model to summarize those anomalies into human-readable rules.

To achieve this, we adopt a methodology inspired by Deng et al. [7], who proposed a framework for outlier summarization through interpretable rules. Their approach highlights the limitations of anomaly detectors that function as black boxes and underscores the need to transform low-level anomaly outputs into actionable, human-interpretable patterns. To achieve this, they trained a decision tree on the detected anomalies with Isolation Forest, enabling the extraction of clear and explainable rules that summarize the underlying outlier behavior. In our project, once anomalous blocks are identified using an unsupervised model (e.g., Isolation Forest), we apply decision tree learning to extract rules that

reflect structural behaviors contributing to centralization. These rules can then be translated into constraints applied within our blockchain simulation to test their impact on network decentralization.

3.8.1 Anomaly detection:

In this stage, we aim to identify blocks or miner behaviors that significantly deviate from typical patterns observed in the Bitcoin network. Anomaly detection is a critical component of our pipeline, as it allows us to isolate outlier behavior that may contribute to centralization, such as short-term mining spikes or repeated block dominance.

We employ an anomaly detection algorithm, which is **Isolation Forest (iForest)** this model is well-established in the field of unsupervised anomaly detection.

Isolation Forest is based on the principle that anomalies are few and different, and therefore easier to isolate. As described by Liu et al. [8], anomaly detection using iForest is a two-stage process. In the first stage (training), the algorithm builds an ensemble of binary trees (isolation trees) using random subsets of the training data. In the second stage (testing), each data point is passed through these trees, and an anomaly score is computed based on the average path length needed to isolate it. Anomalous instances are typically isolated with fewer splits, resulting in higher scores. iForest is especially effective for high-dimensional datasets and can handle cases where the training set does not contain any labeled anomalies [8].

Its robustness, efficiency, and ability to handle irrelevant features make it well-suited for detecting unusual mining behaviors within the Bitcoin network [8]. The following figure illustrates the way IF works:

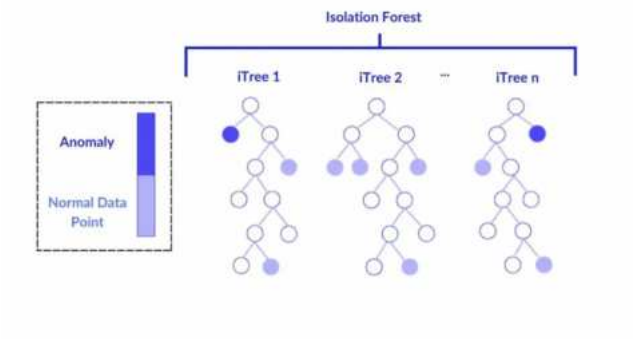


Figure 3.2 Isolation Forest

3.8.2 Clustering:

After applying unsupervised anomaly detection, we observed that the flagged anomalies were **heterogeneous**, representing different behavioral patterns rather than a single, unified type of outlier.

To better understand and organize these behaviors, we introduced a clustering step to group the anomalies based on their feature similarities.

We applied the **K-Means clustering algorithm** to the anomaly subset, using the same features extracted in Section 3.7. K-Means is an unsupervised algorithm that partitions data into *k* clusters by minimizing the intra-cluster variance. It starts by randomly initializing *k* centroids, then iteratively assigns each data point to the nearest centroid and updates the centroids based on the mean of assigned points until convergence. This results in compact and well-separated clusters that help reveal structure within the anomaly set. The following figure illustrates the way k-means works:

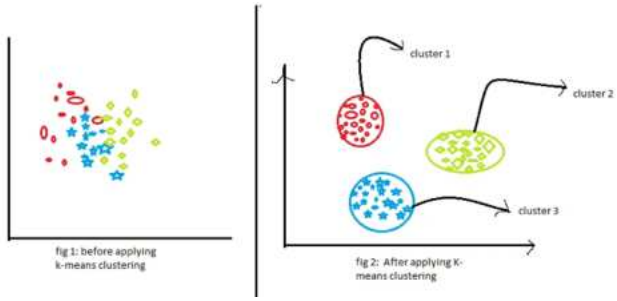


Figure 3.3 K-means clustering algorithm

In our context, the objective was not to evaluate cluster quality or assign semantic labels, but to **segregate distinct behavioral patterns** and isolate the subset of anomalies most relevant to our centralization problem.

By selecting only, the anomaly cluster(s) aligned with our decentralization risk indicators, we ensure that the **decision tree model in the next step is trained on data that is both focused and interpretable**, maximizing its practical value for simulation and rule enforcement.

3.8.3 Decision Tree to extract interpretable rules:

After identifying and filtering anomalies using clustering, we apply a decision tree to the selected anomaly subset to derive **interpretable rules** that explain why those blocks deviate from typical mining behavior.

Decision tree learning is a classical classification technique where the learned function can be represented by a decision tree. It classifies instances by sorting them down the tree from the root to the leaf node, which could predict the label of this instance. Each node in the tree denotes the test of the specific attribute, and the instance is classified by moving down the tree [8].

Decision trees are particularly suited for this task due to their transparency and rule-based logic. The model was trained using the features outlined in Section 3.7 and the resulting model outputs clear conditions (e.g., “IF miner appears in 3 of last 10 blocks AND is not unique today → anomaly”),

which can be directly translated into constraints for improving network decentralization. The following figure illustrates the way a decision tree works:

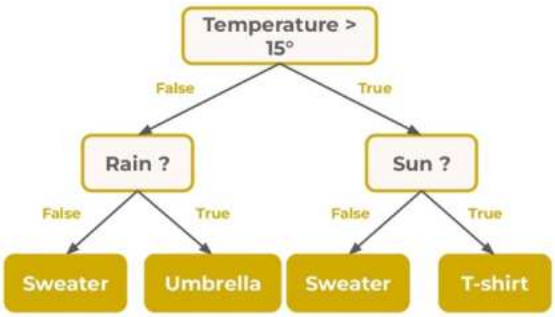


Figure 3.4 Decision Tree

In this section, we presented our model preparation pipeline, which consists of anomaly detection, clustering, and rule extraction. We first detected abnormal mining behaviors using unsupervised models, then clustered them to isolate the most relevant patterns. Finally, we applied a decision tree to extract interpretable rules for guiding decentralization-focused interventions. Results will be considered in the next chapter.

3.9 Implementation and Testing:

In the final stage of our design pipeline, we outline how the proposed system will be implemented and how its effectiveness will be evaluated. This includes integrating the decision rules extracted in the previous phase into our custom-built blockchain network and conducting controlled simulations to assess their impact on decentralization. In this section, we describe the planned rule integration, simulation setup, and comparison strategy that will guide the evaluation process in the next chapter.

3.9.1 Rule integration:

The extracted rules will be incorporated into the consensus mechanism layer of our blockchain network, specifically within the logic that nodes use to validate incoming chains. Each node will verify whether the miner of each incoming block complies with the predefined decentralization rules. Blocks that violate any of the rules will be rejected and excluded from the accepted chain.

Additionally, miners will be prevented from successfully mining new blocks if their behavior does not align with the extracted constraints (e.g., exceeding daily block limits or violating miner uniqueness). This integration ensures that both block validation and mining activity across the network strictly follow the rules designed to promote decentralization.

3.9.2 Simulation design:

To evaluate the system effectively, we will simulate the addition of 1,000 blocks in a controlled environment consisting of 10 nodes with varying computational power. The simulation will be conducted in two phases: one **before** the application of the decision rules (baseline), and one **after** the rules are enforced (enhanced).

By leveraging our custom blockchain network—over which we maintain full control—we can systematically test how the AI-derived rules influence network behavior under repeatable and measurable conditions. This controlled environment is essential for isolating the effects of the rule-based enhancements.

3.9.3 Comparing strategy:

The two simulation scenarios (pre- and post-rule enforcement) will be compared using multiple evaluation criteria. First, we will assess decentralization performance using the four metrics introduced in Section 3.6: Gini coefficient, Nakamoto coefficient, Herfindahl-Hirschman Index, and Shannon entropy.

Second, we will compare operational factors such as the total time required to mine 1,000 blocks and the progression of mining difficulty throughout the simulation. Together, these metrics will provide a comprehensive picture of the system's behavior and allow us to assess the effectiveness of our rule-based enhancements.

3.10 Use case diagram

The following figure shows the use case diagram and the stakeholders that interact with it.

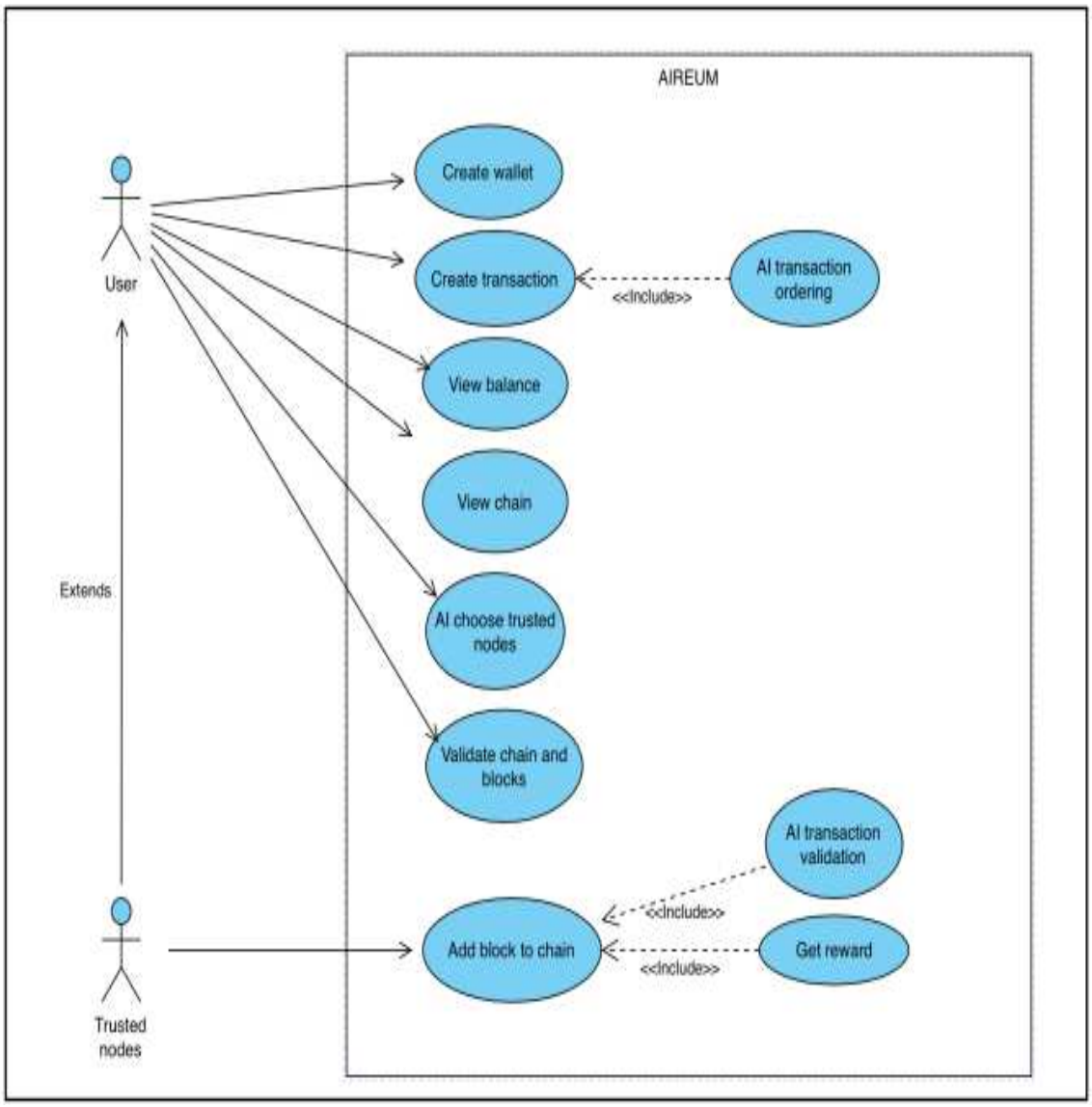


Figure 3.5 Use Case

3.11 Requirements analysis

- The system should Set up the overall blockchain application with all main components.

- The system should create the fundamental blockchain class.

- The system should build the genesis block

- to get the blockchain going for the first time.

- The system should create the core transaction miner users.

- The system should create the core wallet class for the cryptocurrency.

- The system initiates a new user and gives him a wallet.

- The system should calculate the balance for users.

- The system should develop functionality to validate the blockchain, to allow users to sync and replace the new chain.

- The system should implement the proof of work system.

- The system should create transactions when users want.

- The system should build a mempool to store transactions.

- The system should choose trusted nodes to get the authority to add blocks.

- The system should prioritize transactions in the mempool.

- The system should analyze user behavior.

- The system should provide interfaces for users to use the system.

3.12 Use case specification

Specification of the create wallet use case:

Table 3.2 Create Wallet specification

Use case ID : 01
Use case Name: Create wallet
Created by: Ahmad Shekha and Farah Fares
Priority: high
Primary actors: User
Description: The create wallet case allows users to create a new wallet on a blockchain network. The wallet includes a public address and a private key. The public address can be used to send funds to the wallet, and the private key can be used to access the funds stored in the wallet.
Basic flow 1-The user initiates the case by clicking on a button or link. 2- The system create a key Pair a public and private key 3- The system print the hash of public and private key and ask user to save them and keep private key safe.

Specification of the Send transaction use case

Table 3.3 Send Transaction Use case

Use case ID: 02
Use case Name: Create Transaction
Created by: Ahmad Shekha and Farah Fares
Priority: high
Primary actors: User
Description: The create transaction use case allows users to create a new transaction on a blockchain network.
Basic flow 1- The user initiates the case by clicking a button or link to create a new transaction. 2- The user enters transaction information the recipient and the amount. 3 - The system checks the validation of the transaction by checking the amount and recipient. 4-The system signs the transaction with the private key of the sender and broadcasts the transaction to the network 5 – Each node on the system checks the transaction syntax and adds the transaction to its Mempool waiting to be added to the block.
Alternative flow: **Alt Step 3:** The amount exceeds the sender's balance; the system tells the sender to enter a valid amount. **Alt Step 4:** The sender didn't give the recipient's address; the system tells the sender to put the recipient's address.

Specification of validating block and chain use case:

Table 3.4 Validate Chain specification

Use case ID: 03
Use case Name: Validate chain and block
Created by: Ahmad Shekha and Farah Fares
Priority: high
Primary actors: User and miner
Description: This use case initiates whenever a node receives a new chain
Basic flow 1- The user receives a new chain to replace it with the current one. 2- The system checks if the received chain is longer than the current. 3 - The system checks if the blocks are connected well with hashes. 4-The system checks if there is a block with more than one reward transaction. 5- The system checks if there is an invalid transaction in any block of the chain. 6- The system checks if there is a transaction with an amount that exceeds the sender's balance. 7- The system checks if there is a duplicate transaction in any block. 8- The system replaces the user the current chain with the received chain.
Alternative flow: **Alt Step 2,3,4,5,6,7:** If any check fails, the system doesn't replace the chain with the received chain.

Specification of View balance use case:

Table 3.5 View balance specification

Use case ID: 04
Use case Name: view balance
Created by: Ahmad Shekha and Farah Fares
Priority: high
Primary actors: User
Description: describes the process of a user viewing their account balance in a blockchain.
Basic Flow: 1-The system receives a request from the user to view their balance. 2-The system calculates the balance of the user by calculating received transaction outputs that are associated with the user's address 3- The system keeps doing this process till it finds a transaction the user sent before, he just adds the output of that transaction to the value it had before to get the last balance.
Alternative Flow: **Alt Step 2:** The system didn't find any transaction associated with the user address; The system returns the starting value.

3.13 UML Class Diagram:

The following diagram shows the Classes in our system with each one attributes and methods:

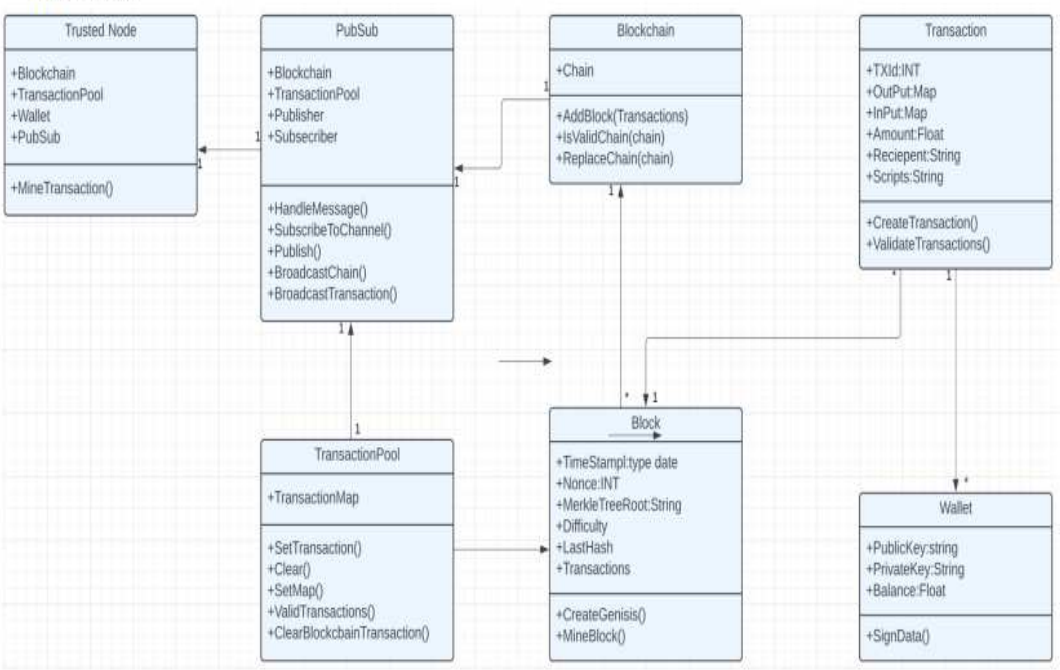
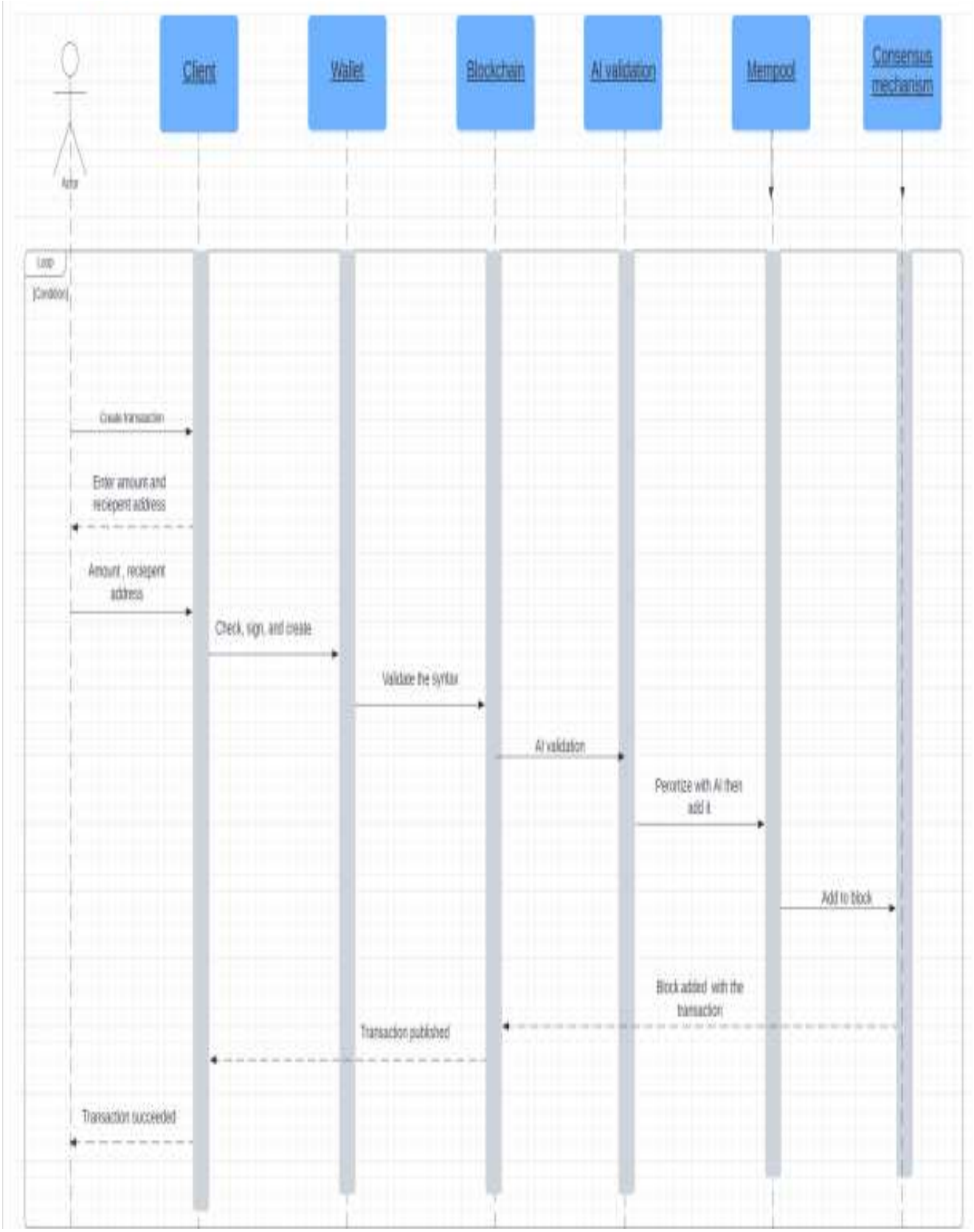


Figure 3.6 UML Class Diagram

3.14 Sequence diagram



3.15 Unit test list:

1- Block Unit Test:

Figure 3.7 Sequence diagram

Table 3.6 Block unit test

Test ID	Test Name	Test Modification(s)	Result
T-01	Block creation	Initiate the block with all its component	
T-02	Genesis block	Initiate genesis block as first block	
T-03	MerkleTree	Each block should have a Merkle tree for transactions in it and save its root hash	
T-04	Block mining	Mining block after adding data to it	
T-05	Adjust difficulty	Adjust difficulty based on specific time difference	

2- Blockchain Unit Test:

Table 3.7 Blockchain Unit Test

Test ID	Test Name	Test Modification(s)	Result
T-01	Chain creation	Create chain with genesis block as first block	
T-02	Add block	Add new block to the chain and broadcast it	
T-03	New chain validate	Check if new chain is longer	
T-04	New chain validate	Check if new chain has invalid data	
T-05	New chain validate	Check if new chain has a block with more than on reward transaction	
T-06	New chain validate	Check if new chain has a block with invalid transaction	
T-07	New chain validate	Check if new chain has a transaction with invalid input amount	
T-08	New chain validate	Check if new chain has a block with duplicated transaction	

3 – Transaction Unit Test:

Table 3.8 Transaction Unit test

Test ID	Test Name	Test Modification(s)	Result
T-01	Transaction initiate	Initiate new transaction instance with the inputs and outputs	
T-02	Reward transaction	Initiate transaction reward only for trusted nodes	
T-03	Validate transaction	Validate transaction after being published, check transactions amounts and sign	

4 – wallet Unit Tests:

Table 3.9 Wallet unit test

Test ID	Test Name	Test Modification(s)	Result
T-01	Wallet creation	Initiate new wallet for user with balance and keyPair	
T-02	Transaction Sign	Wallet sign transaction and verify signature	
T-03	Create Transactions	Wallet creates and verify transaction and then sign it	
T-04	Calculate Balance	Wallet calculates real balance for each user	

5- Transaction Pool Unit Test:

Table 3.10 Transaction Pool Unit test

Test ID	Test Name	Test Modification(s)	Result
T-01	TransactionPool initiate	Initiate new transaction pool for each user	
T-02	Add Transactions	Transaction pool should add received transaction to the pool	
T-03	Sync poolMap real-time	Sync pool map with last copy when initiate	
T-04	Validate transaction in pool	Validate all transactions in the pool before start mining	
T-05	Clear pool after new block	Pool should be clear the Pool after new block added	

3.16 Conclusion

This chapter presented the full design and methodology of our proposed system to address centralization trends in the Bitcoin network using artificial intelligence. We began by building a custom blockchain platform modeled after Bitcoin, which allowed us to test protocol-level enhancements in a controlled environment. Centralization risks were identified through a detailed analysis of Bitcoin's workflow and assessed using four established decentralization metrics.

We then extracted miner behavior features, applied unsupervised anomaly detection, and used clustering to isolate relevant outliers. A decision tree model was trained to extract interpretable rules, which were prepared for enforcement within our consensus mechanism. The chapter concluded with the integration plan and simulation strategy to test the impact of our rules.

The next chapter focuses on implementing this design, running simulations, and analyzing whether our AI-driven rules enhance decentralization as intended

Chapter 4
Implementation

4.1 Introduction

In this chapter, we present the practical implementation of our proposed system. The process consists of the following key stages:

Blockchain Network Implementation and Testing – We built a custom blockchain platform from scratch and conducted unit and integration tests to validate its core functionality.

Bitcoin Decentralization Analysis – We applied four decentralization metrics on historical Bitcoin block data to empirically verify the trend toward centralization.

AI Model Implementation – We trained unsupervised models (Isolation Forest and OCSVM) to detect anomalies, clustered the results, and used a decision tree to extract interpretable rules.

Rule Integration and Simulation Setup – The extracted rules were implemented into the network's consensus logic, and simulation scenarios were designed to compare system behavior before and after applying these constraints.

These four stages represent the full implementation of our system and provide the basis for evaluating its effectiveness in promoting decentralization.

4.2 Tools and technologies:

- Frameworks: Express for the backend and React.js for the frontend.
- Programming Language: JavaScript (for both backend and blockchain backend) and Python for AI models.
- Communication Protocol: REST API for connecting the front-end with the back-end.
- Communication between nodes: We used NATS to create a Pub/Sub network, where each node can send messages to all nodes and receive messages from all nodes.
- JavaScript runtime environment: We used NodeJS as the JavaScript runtime environment.

4.3 Blockchain Network Implementation and Testing:

4.3.1 Main Structure:

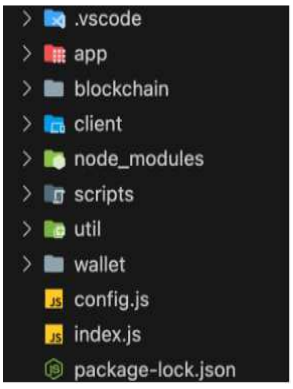


Figure 4.1 Main Structure

4.3.2 Main classes

The following figures illustrate the implementation of the main blockchain components that were conceptually defined in the design phase (see Section 3.3). These include the core classes responsible for block construction, transaction construction, and creating a unique wallet for user.

Block class:

```
class Block {
  constructor({ timestamp, lastHash, hash, data, nonce, difficulty , miner_id}) {
    this.timestamp = timestamp;
    this.lastHash = lastHash;
    this.hash = hash;
    this.data = data;
    this.nonce = nonce;
    this.difficulty = difficulty;
    this.miner_id = miner_id;
    const merkleTree = new MerkleTree(transactionStrings);
```

Figure 4.2 Block Class

Transaction class:

```
constructor({ senderWallet, recipient, amount, outputMap, input }) {
  this.id = uuid();
  this.outputMap = outputMap || this.createOutputMap({ senderWallet, recipient, amount });
  this.input = input || this.createInput({ senderWallet, outputMap: this.outputMap });
}

createOutputMap({ senderWallet, recipient, amount }) {
  const outputMap = {};

  outputMap[recipient] = amount;
  outputMap[senderWallet.publicKey] = senderWallet.balance - amount;

  return outputMap;
}

createInput({ senderWallet, outputMap }) {
  return {
    timestamp: Date.now(),
    amount: senderWallet.balance,
    address: senderWallet.publicKey,
    signature: senderWallet.sign(outputMap)
```

Figure 4.3 Transaction Class

Wallet class:

```
class Wallet {


  constructor(){
   this.balance = starting_balance;
   this.keyPair = ec.genKeyPair();
   this.publicKey = this.keyPair.getPublic().encode('hex');
    };
```

Figure 4.4 Wallet Class

4.3.3 Unit Testing results:

To ensure the reliability and correctness of the system, we utilized **Jest** to implement a comprehensive suite of unit tests covering the core functionalities of the blockchain. These tests were designed based on the test planning strategy outlined in **Section 3.15**. All test suites executed successfully, as shown in the figure below. The full test implementation is available in the project's GitHub repository [9]. In the following sections, we first present an overview of the test results, followed by a detailed examination of each test case and its outcome.

The overall tests result:

```
PASS  util/crypto-hash.test.js
PASS  blockchain/block.test.js
PASS  blockchain/index.test.js
PASS  wallet/index.test.js
PASS  wallet/transaction.test.js
PASS  wallet/transaction-pool.test.js (5.795 s)

Test Suites: 6 passed, 6 total
Tests:       75 passed, 75 total
Snapshots:   0 total
Time:        6.119 s
Ran all test suites.

Watch Usage: Press w to show more.
```

Figure 4.5 Overall result

Now, we proceed to examine the individual test suites corresponding to each core component defined in the system analysis (see Section 3.15).

```
PASS blockchain/block.test.js
  Block
    ✓ has a timestamp, lastHash, hash, and a data property (2 ms)
    genesis()
      ✓ returns a Block instance
    mineBlock
      ✓ returns a Block instance
      ✓ sets the `lastHash` to be the `hash` of the lastBlock
      ✓ sets the `data`
      ✓ sets a `timestamp` (1 ms)
      ✓ creates a SHA-256 `hash` based on the proper inputs
      ✓ sets a `hash` that matches the difficulty criteria
      ✓ adjusts the difficulty (1 ms)
    adjustDifficulty()
      ✓ raises the difficulty for a quickly mined block
      ✓ lowers the difficulty for a slowly mined block (1 ms)
      ✓ has a lower limit of 1

Test Suites: 1 passed, 1 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        0.314 s, estimated 1 s
```

Figure 4.6 Block test

4.3.3.1 Block test results:

The following figure presents the test results for the Block component that were executed.

Based on the figure above, the summarized test outcomes for the Block component are detailed in the following table

Table 4.1 Block test result

Test ID	Test Name	Test Modification(s)	Result
T-01	Block creation	Initiate the block with all its component	Passed
T-02	Genesis block	Initiate genesis block as first block	Passed
T-03	MerkleTree	Each block should have a Merkle tree for transactions in it and save its root hash	Passed
T-04	Block mining	Mining block after adding data to it	Passed
T-05	Adjust difficulty	Adjust difficulty based on specific time difference	Passed

4.3.3.2 Blockchain network test result:

The following figure presents the test results for the Blockchain component that were executed.

```
PASS blockchain/index.test.js
  Blockchain
    ✓ contains a `chain` Array instance (2 ms)
    ✓ starts with the genesis block (1 ms)
    ✓ adds a new block to the chain (1 ms)
    isValidChain()
      when the chain does not start with the genesis block
        ✓ returns false (1 ms)
      when the chain starts with the genesis block and has multiple blocks
        and a lastHash reference has changed
          ✓ returns false (3 ms)
        and the chain contains a block with an invalid field
          ✓ returns false (2 ms)
        and the chain does not contain any invalid blocks
          ✓ returns true (2 ms)
    replaceChain()
      when the new chain is not longer
        ✓ does not replace the chain
        ✓ logs an error (1 ms)
      when the new chain is longer
        and the chain is invalid
          ✓ does not replace the chain (2 ms)
          ✓ logs an error (3 ms)
        and the chain is valid
          ✓ replaces the chain (2 ms)
          ✓ logs about the chain replacement (2 ms)
      and the `validateTransactions` flag is true
        ✓ calls validateTransactionData()
    validTransactionData()
      and the transaction data is valid
        ✓ returns true (290 ms)
      and the transaction data has multiple rewards
        ✓ returns false and logs an error (274 ms)
      and the transaction data has at least one malformed outputMap
        and the transaction is not a reward transaction
          ✓ returns false and logs an error (105 ms)
        and the transaction is a reward transaciton
          ✓ returns false and logs an error (256 ms)
      and the transaction data has at least one malformed input
        ✓ returns false and logs an error (331 ms)
      and a block contains multiple identical transactions
        ✓ returns false and logs an error (403 ms)

Test Suites: 1 passed, 1 total
Tests:       20 passed, 20 total
Snapshots:   0 total
Time:        1.997 s, estimated 3 s
```

Figure 4.7 blockchain network test

Based on the figure above, the summarized test outcomes for the Blockchain component are detailed in the following table

Table 4.2 Blockchain test result

Test ID	Test Name	Test Modification(s)	Result
T-01	Chain creation	Create chain with genesis block as first block	Passed
T-02	Add block	Add new block to the chain and broadcast it	Passed
T-03	New chain validate	Check if new chain is longer	Passed
T-04	New chain validate	Check if new chain has invalid data	Passed
T-05	New chain validate	Check if new chain has a block with more than on reward transaction	Passed
T-06	New chain validate	Check if new chain has a block with invalid transaction	Passed
T-07	New chain validate	Check if new chain has a transaction with invalid input amount	Passed
T-08	New chain validate	Check if new chain has a block with duplicated transaction	Passed

4.3.3.3 Transaction test results:

The following figure presents the test results for the Transaction component that were executed.

```
PASS wallet/transaction.test.js
  Transaction
    ✓ has an `id` (133 ms)
    outputMap
      ✓ has an `outputMap` (113 ms)
      ✓ outputs the amount to the recipient (110 ms)
      ✓ outputs the remaining balance for the `senderWallet` (115 ms)
    input
      ✓ has an `input` (108 ms)
      ✓ has a `timestamp` in the input (108 ms)
      ✓ sets the `amount` to the `senderWawllet` balance (113 ms)
      ✓ sets the `address` to the `senderWallet` publicKey (114 ms)
      ✓ signs the input (278 ms)
    validTransaction
      when the transaction is valid
        ✓ returns true (266 ms)
      when the transaction is invalid
        and a transaction outputMap value is invalid
          ✓ returns false and logs an error (111 ms)
        and a transaction input Signature is invalid
          ✓ returns false and logs an error (368 ms)
    update()
      and the amount is invalid
        ✓ throws an error (121 ms)
      and the amount is valid
        ✓ outputs the amount to the next recipient (158 ms)
        ✓ subtracts the amount from the original sender output amount (170 ms)
        ✓ maintains a total output that matches the input amount (151 ms)
        ✓ re-signs the transaction (148 ms)
        and another update for the same recipient
          ✓ adds to the recipient amount (206 ms)
          ✓ subtracts the amount from the original sender output amount (199 ms)
    rewardTransaction()
      ✓ creates a transaction with the reward input (157 ms)
      ✓ creates one transaction for the miner with the `MINING_REWARD` (153 ms)

Test Suites: 1 passed, 1 total
Tests:       21 passed, 21 total
Snapshots:   0 total
Time:        3.693 s, estimated 5 s
```

Figure 4.8 Transaction test

Based on the figure above, the summarized test outcomes for the Transaction component are detailed in the following table

Table 4.3 Table Blockchain test result

Test ID	Test Name	Test Modification(s)	Result
T-01	Transaction initiate	Initiate new transaction instance with the inputs and outputs	Passed
T-02	Reward transaction	Initiate transaction reward only for trusted nodes	Passed
T-03	Validate transaction	Validate transaction after being published, check transactions amounts and sign	Passed

4.3.3.4 Wallet test results:

The following figure presents the test results for the Wallet component that were executed.

```
PASS wallet/index.test.js
  Wallet
    ✓ has a `balance` (68 ms)
    ✓ has a `publicKey` (55 ms)
    signing data
      ✓ verifies a signature (277 ms)
      ✓ does not verify an invalid signature (304 ms)
    createTransaction
      and the amount exceeds the balance
        ✓ throws an error (64 ms)
      and the amount is valid
        ✓ creates an instance of `Transaction` (107 ms)
        ✓ matches the transaction input with the wallet (107 ms)
        ✓ outputs the amount to the recipient (105 ms)
      and a chain is passed
        ✓ calls `Wallet.calculateBalance` (107 ms)
    calculateBalance()
      and there are no outputs for the wallet
        ✓ returns the `STARTING_BALANCE` (52 ms)
      and there are outputs for the wallet
        ✓ adds the sum of all outputs to the wallet balance (264 ms)
        and the wallet has made a transaction
          ✓ returns the output amount of the recent transaction (320 ms)
          and there are outputs next to and after the recent transaction
            ✓ includes the output amoutns in the returned balance (500 ms)

Test Suites: 1 passed, 1 total
Tests:       13 passed, 13 total
Snapshots:   0 total
Time:        2.637 s, estimated 3 s
```

Figure 4.9 Wallet test

Based on the figure above, the summarized test outcomes for the Wallet component are detailed in the following table

Table 4.4 Test Modifications

Test ID	Test Name	Test Modification(s)	Result
T-01	Wallet creation	Initiate new wallet for user with balance and keyPair	Passed
T-02	Transaction Sign	Wallet sign transaction and verify signature	Passed
T-03	Create Transactions	Wallet creates and verify transaction and then sign it	Passed
T-04	Calculate Balance	Wallet calculates real balance for each user	Passed

4.3.3.5 Transactions pool test result:

The following figure presents the test results for the Transaction Pool component that were executed.

```
PASS wallet/transaction-pool.test.js
  TransactionPool
    setTransaction()
      ✓ adds a transaction (129 ms)
    validTransactions()
      ✓ returns valid transactions (1877 ms)
      ✓ logs errors for the invalid transactions (1797 ms)
    clear()
      ✓ clears the transactions (107 ms)
    clearBlockchainTransactions()
      ✓ clears the pool of any existing blockchain transactions (734 ms)

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        4.959 s, estimated 6 s
```

Figure 4.10 Transaction pool test

Based on the figure above, the summarized test outcomes for the Transaction Pool component are detailed in the following table:

Table 4.5 Test Modification

Test ID	Test Name	Test Modification(s)	Result
T-01	TransactionPool initiate	Initiate new transaction pool for each user	Passed
T-02	Add Transactions	Transaction pool should add received transaction to the pool	Passed
T-03	Sync poolMap real-time	Sync pool map with last copy when initiate	Passed
T-04	Validate transaction in pool	Validate all transactions in the pool before start mining	Passed
T-05	Clear pool after new block	Pool should be clear the Pool after new block added	Passed

4.3.4 Integration test:

The following figure shows the result of the integration test designed in Section 3.16. The test was executed using a Python script to simulate node interactions through the system's public API endpoints.

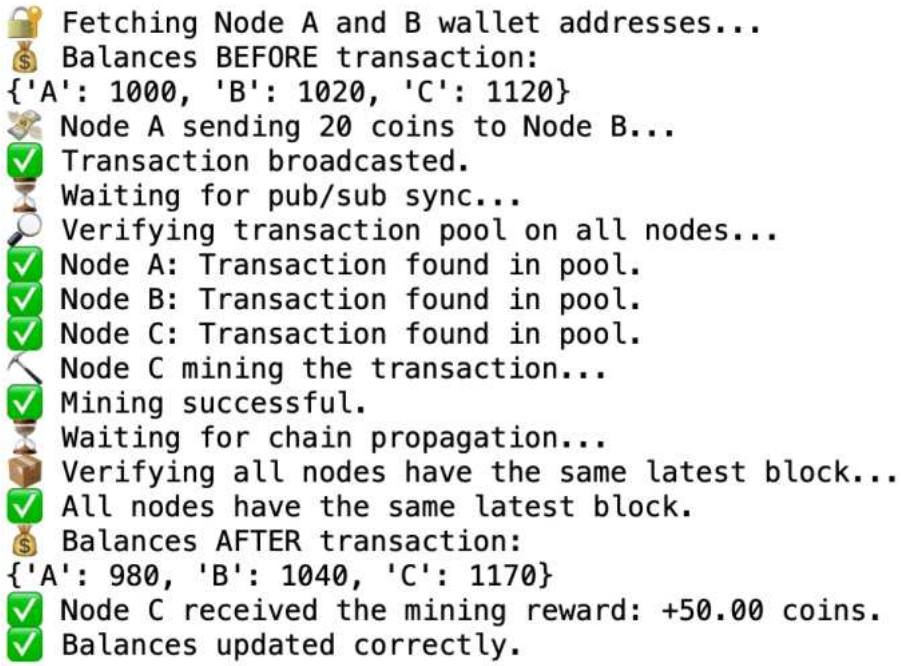


Figure 4.11 Integration test

Based on the output shown in the previous figure, the final result of the integration test is summarized in the following table, as designed in Section 3.15

Table 4.6 Final result of the integration

Step ID	Action	Node(s) Involved	Excepted Outcome	Result
1	Create three nodes	Nodes A, B, C	Three nodes created with unique port for each	Passed
2	Fetch wallet address for each	Nodes A, B, C	Obtain sender and recipient public keys and wallet balance for each node.	Passed
3	Send transaction (20 coins)	Nodes A → B	Transaction created to send 20 coins from node A to node B and broadcast to all nodes	Passed
4	Verify transaction pool	Nodes A, B, C	All nodes include transaction in their pool	Passed
5	Mine a block with transaction included	Node C	Transaction mined into new block and published to other nodes	Passed
6	Verify chain sync	Nodes A, B, C	All nodes have the same latest block with the sent transaction	Passed
7	Check updated balances	Nodes A, B, C	A: -20 (sent), B: +20 (received), C: +50 (mining reward).	Passed

4.4 Bitcoin Decentralization Analysis:

In this section, we analyze the historical decentralization trends of the Bitcoin network using the four key metrics defined in Section 3.6. These metrics—Gini Coefficient, Nakamoto Coefficient, Herfindahl-Hirschman Index (HHI), and Shannon Entropy—were applied to Bitcoin block data collected from 2014 to 2024. The aim is to provide empirical evidence supporting the hypothesis that Bitcoin has become increasingly centralized over time. Each metric was calculated separately for both the **mining pool level** and the **unique miner address level** to capture organizational and individual-level concentration patterns. The results are visualized and discussed in the following subsections.

4.4.1 Nakamoto Coefficient:

The Nakamoto Coefficient represents the minimum number of entities that together control more than 50% of the network's mining power. A higher value indicates stronger decentralization and network resilience, while a lower value reflects increased vulnerability to collusion or control by a small group. The following figure shows the Nakamoto Coefficient trend across the dataset:

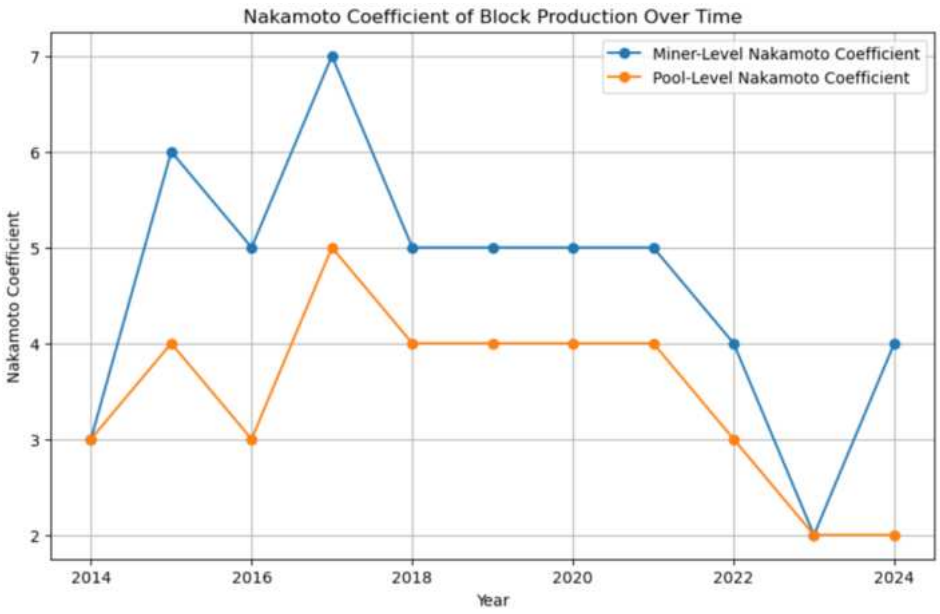


Figure 4.12 Nakamoto Coefficient of Bitcoin

As shown in the previous figure, the Nakamoto Coefficient declined from **6 in 2014** to **2 in 2023**, before slightly recovering to **4 in 2024** at the mining pool level. This indicates that by 2023, as few as two mining pools had the potential to compromise the network's consensus, highlighting a significant centralization trend—particularly evident after 2017.

4.4.2 Shannon Entropy:

Shannon Entropy measures the randomness or diversity in miner participation. Higher entropy values indicate a more uniform distribution of mining power, while lower values suggest that block production is dominated by fewer entities. The following figure shows the Shannon Entropy across the dataset:

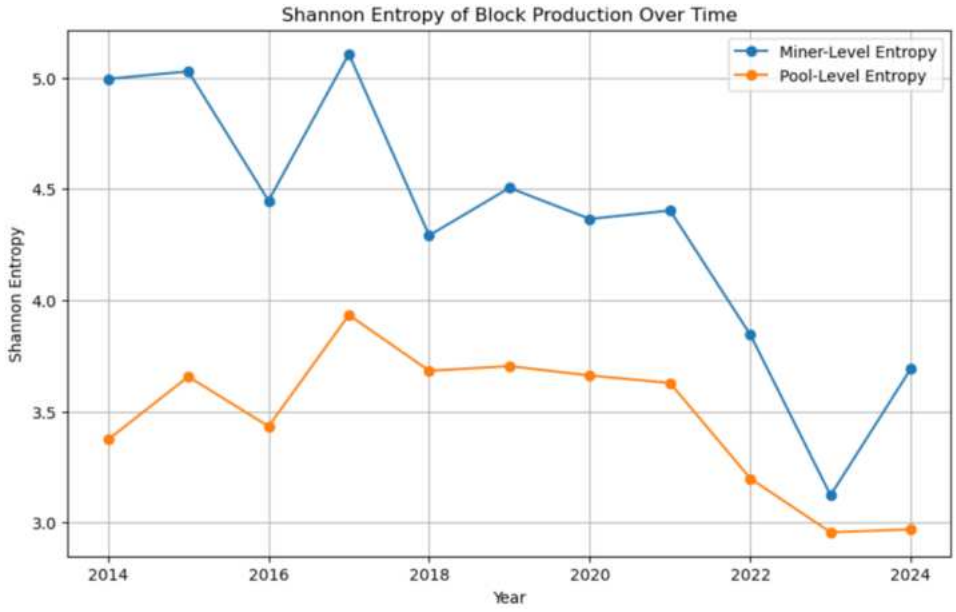


Figure 4.13 Shannon Entropy of Bitcoin

As shown in the previous figure, Shannon Entropy declined from **3.4 in 2014** to **2.9 in 2024**, with a more notable drop from its peak of **3.9 in 2017**. This reflects a gradual reduction in miner diversity, particularly after 2017. The results suggest that block generation has become increasingly concentrated among fewer unique participants, reinforcing the centralization trend observed through the other metrics.

4.4.3 Gini coefficient:

The Gini Coefficient was used to quantify the inequality in block production among miners. A higher Gini value indicates a greater concentration of mining power, while a lower value suggests a more even distribution. This analysis helps identify periods of rising centralization in the network.

As shown in the Next figure, the Gini coefficient increased steadily from **0.35 in 2020 to 0.82 in 2023 after it had a drop to 0.70** at the mining pool level, indicating a significant rise in inequality. The unique address level followed a similar trend. This supports the hypothesis that mining activity has become increasingly dominated by fewer large entities.

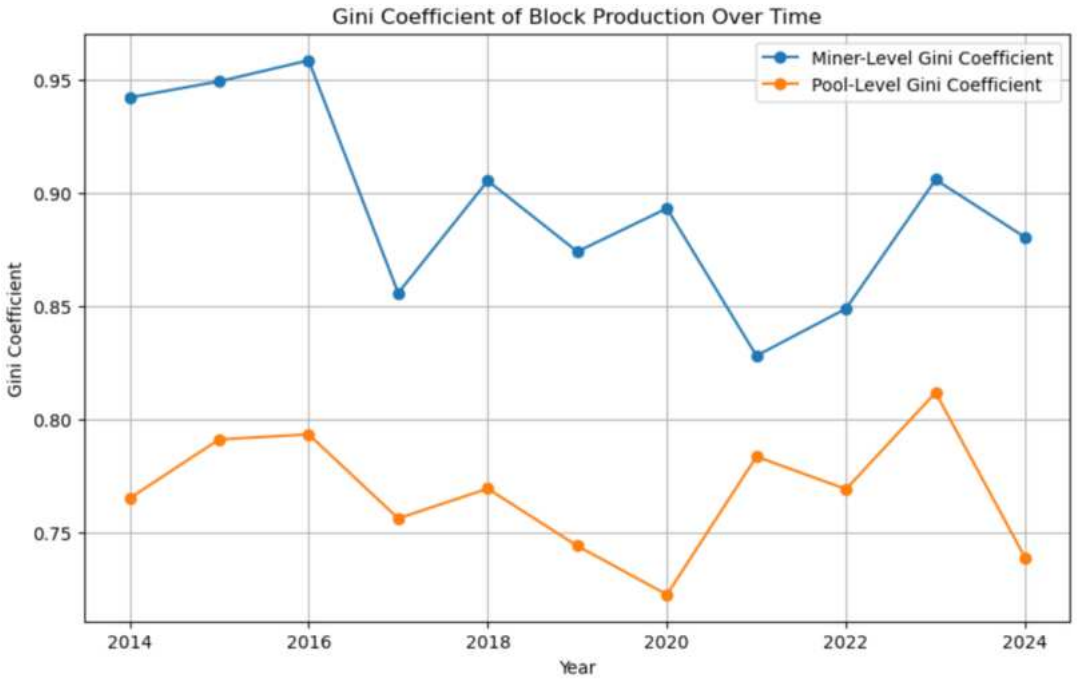


Figure 4.14 Gini Coefficient of Bitcoin

4.4.4 Herfindahl-Hirschman Index (HHI):

The Herfindahl-Hirschman Index (HHI) is a widely used economic metric that quantifies market concentration. In our case, it measures how concentrated block production is among mining entities. Values closer to 1 indicate monopoly-like dominance, while values closer to 0 imply a more competitive environment. The following figure shows the HHI across the dataset:

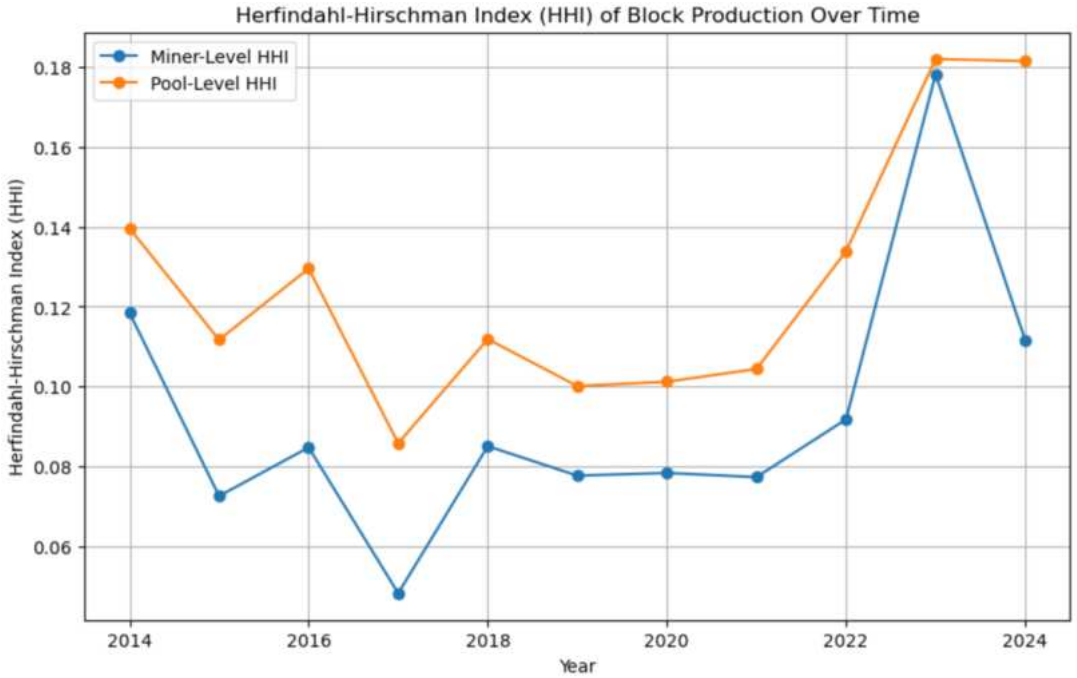


Figure 4.15 Herifindahl – Hirschman Index

According to the previous figure, the Herfindahl-Hirschman Index (HHI) increased from **0.14 in 2014** to **0.19 in 2024**, with a noticeable rise following its peak of **0.09 in 2017**. This upward trend indicates a

growing concentration of mining activity, particularly among large institutional mining pools, reflecting a gradual consolidation of power within the network.

In conclusion, all four-decentralization metrics consistently indicate that **Bitcoin reached its peak decentralization in 2017**, after which the network exhibited a clear trend toward increasing centralization. While the **Gini Coefficient** showed a delayed response—with the centralization effect becoming more apparent around **2020**—the **Nakamoto Coefficient**, **HHI**, and **Shannon Entropy** began reflecting this shift immediately after 2017. This trend was evident not only at the **mining pool level** but also at the **unique miner address level**, which closely mirrored the patterns observed among pools. These results confirm that Bitcoin's mining landscape has become increasingly concentrated over time, reinforcing the need for intervention to preserve decentralization.

4.5 AI Model Implementation:

In this section, we present the implementation of our AI-based approach for detecting centralization-related anomalies in the Bitcoin network and extracting interpretable rules that inform protocol-level improvements. The models were built upon the feature set described in Section 3.7 and reflect the methodology outlined in Section 3.8. The implementation involves three key stages: anomaly detection using unsupervised models, clustering anomalies into behavioral groups, and extracting human-readable rules using a decision tree classifier. Each stage is implemented and evaluated individually before being integrated into our blockchain simulation system.

4.5.1 Data Preparation:

To train and evaluate our AI models effectively, we focused on Bitcoin block data from the year **2017**, as identified in Section 4.4 as the **peak of decentralization**. This choice allows us to build the model using the most balanced and diverse period in the network's mining history. The dataset includes approximately **60,000 blocks**, collected using **Google BigQuery** and filtered to retain only those generated in 2017. We will focus our study solely on miner pools to provide a real-world analysis.

The features used in model training are the same as those defined in **Section 3.7**, which capture miner activity patterns, temporal distribution, and decentralization-related indicators. Before model training, the entire dataset was **normalized and scaled** to fall within the range of **0 to 1**, ensuring consistency across features and optimizing the performance of distance-based models such as **Isolation Forest**, **OCSVM**, and **K-Means**. This preprocessed dataset forms the input for the anomaly detection and rule extraction stages described in the following sections.

4.5.2 Anomaly Detection Model Implementation:

In this phase, we implemented the **Isolation Forest (iForest)** algorithm to detect blocks exhibiting anomalous mining behavior that may contribute to network centralization. iForest is particularly well-suited for unsupervised anomaly detection in high-dimensional data, as it isolates anomalies based on the principle that outliers are few and different and can be separated with fewer random partitions. This makes it ideal for identifying irregular miner activity without requiring labeled data.

To evaluate the sensitivity of the model and the implications of different anomaly definitions, we applied Isolation Forest using two contamination thresholds: **0.2** and **0.3**. These thresholds determine the proportion of data that the model treats as anomalous, and they directly influence the nature of the decision rules that will be extracted in the next phase.

- At **0.2**, the model is more **conservative**, labeling only the most extreme 20% of blocks as anomalies. This results in **more lenient criteria for being considered normal**, with only strong deviations being flagged.
- At **0.3**, the model becomes more **aggressive**, labeling 30% of blocks as anomalous. Consequently, the **criteria for normality become stricter**, since the model flags even mild deviations.

This tunable behavior makes the contamination parameter a useful tool for blockchain developers: **lower thresholds emphasize stability**, targeting only severe centralization risks, while **higher thresholds allow for earlier intervention**, even in cases of subtle miner dominance. In this study, both thresholds are evaluated to understand how rule strictness can be modulated in future consensus design. The figures below present the anomalies detected under both thresholds.

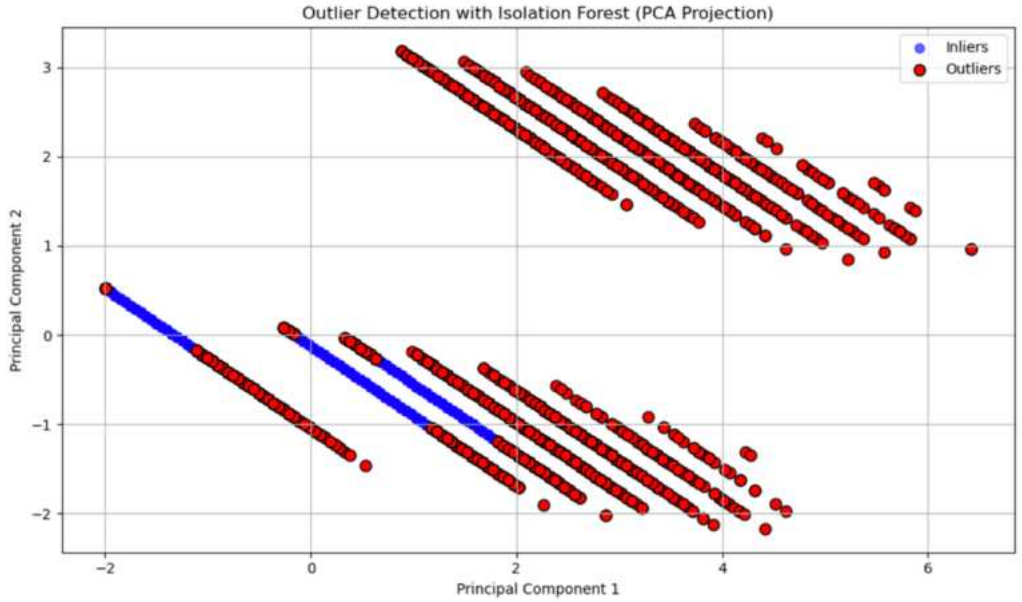


Figure 4.16 Outlier detection with 0.3 threshold

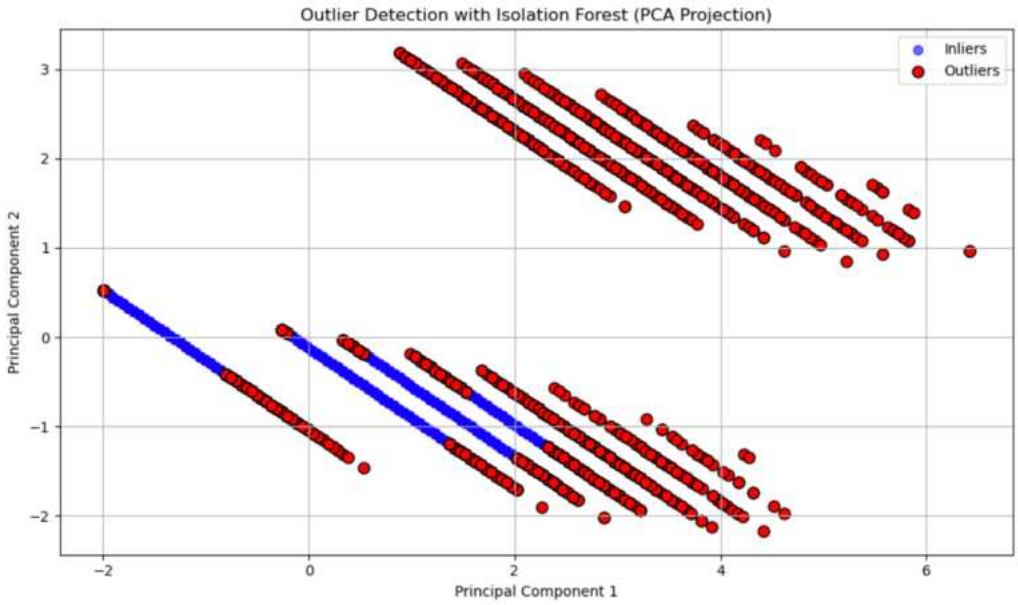


Figure 4.17 Outlier detection with 0.2 threshold

At a threshold of 0.2, the model labeled approximately **11,000 blocks** as anomalies, while the 0.3 threshold increased this number to **16,000 anomalies**. As shown in the previous figures, the lower threshold (0.2) results in a more conservative model, labeling a greater portion of blocks as normal. However, upon further inspection, it became evident that these anomalies were **not uniform** in nature—some reflected benign miner behavior, while others aligned more closely with **centralization risks**. To address this, we introduce a **clustering step** in the next section, allowing us to differentiate between types of anomalies and ensure that **only relevant behaviors** are used for training the decision tree.

4.5.3 Clustering:

Following the anomaly detection phase, we observed that the flagged anomalous blocks were **not homogeneous**—they reflected diverse patterns of unusual behavior. Some anomalies were relevant to centralization (e.g., dominance of a single miner over a short window), while others were unrelated to our target risk. To **better organize and interpret these behaviors**, we introduced a clustering step.

We applied the **K-Means** algorithm to the subset of blocks identified as anomalous by the Isolation Forest model. K-Means is an unsupervised clustering method that partitions data into *k* groups by minimizing intra-cluster variance. It begins by initializing *k* centroids, then iteratively assigns each data point to its nearest centroid and updates the centroids based on the mean of points in each cluster, until convergence.

In our case, the goal was **not to assign formal labels or evaluate cluster accuracy**, but to use clustering as a tool to:

- Identify distinct sub-patterns within the anomaly set
- **Isolate the cluster(s)** most indicative of centralizing miner behavior
- Filter out unrelated anomalies before rule extraction

This selective focus ensures that the **decision tree model** trained in the next phase operates only on **meaningful anomalies**, maximizing interpretability and relevance to our decentralization objectives.

The following figures illustrates the result of clustering applied to the detected anomalies, where each color represents a distinct behavioral pattern:

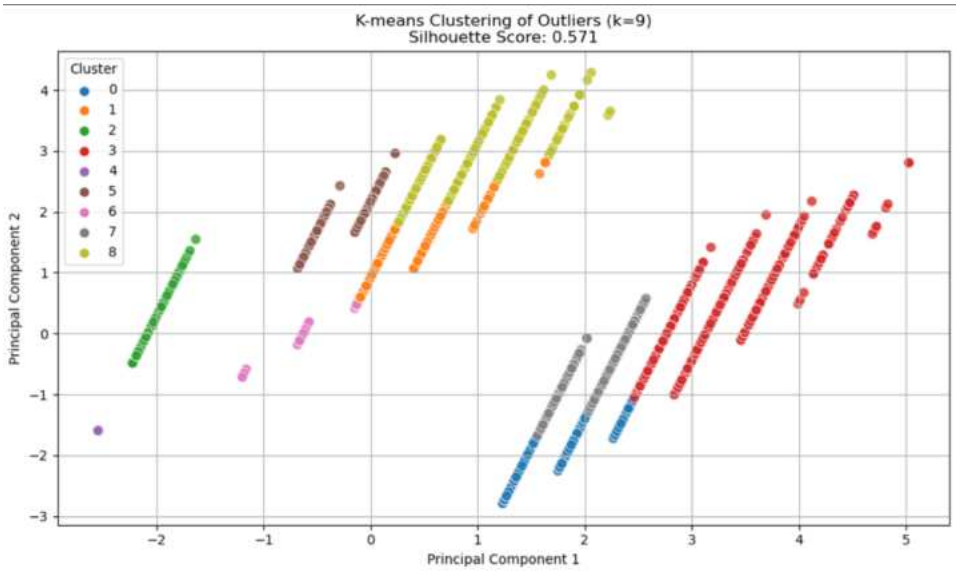


Figure 4.18 k means clustering with 0.3 threshold

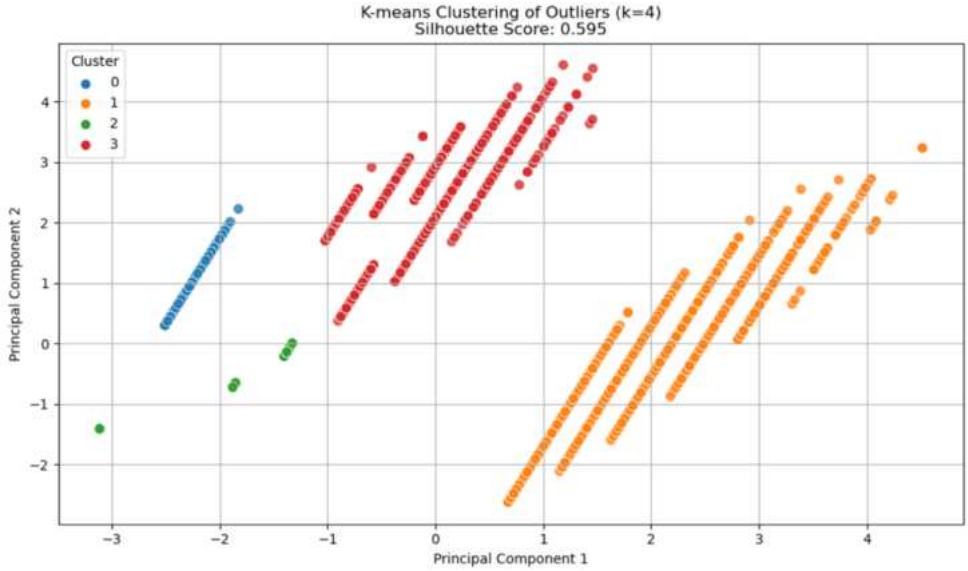


Figure 4.19 k means clustering with 0.2 threshold

Based on the clustering visualizations shown in the previous figures, the anomalous blocks were grouped using the K-Means algorithm. For the 0.3 contamination threshold, the blocks were divided into **9 clusters**, and for the 0.2 threshold, into **4 clusters**, with the number of clusters selected using the **Silhouette score** to ensure optimal separation.

We conducted a manual inspection of each cluster to interpret the behavioral patterns they represented. This analysis revealed that **not all anomalies were harmful** to decentralization. Specifically, some clusters contained blocks produced by **low-activity miners**—such as those who mined only one block per day or contributed minimally across recent windows. While these behaviors were statistically unusual, they do not pose a centralization risk. In fact, they reflect **healthy miner diversity** within the ecosystem.

For the **0.3 threshold**, benign low-activity behaviors were mainly grouped in **clusters 2, 4, and 6**. For the **0.2 threshold**, these behaviors were largely represented in **cluster 2**. We reclassified the blocks belonging to these clusters as **normal**, as they did not align with the centralization risks we aimed to target.

In contrast, the remaining clusters exhibited clear signs of **dominant miner behavior,** and consistent with behaviors known to threaten decentralization, and were retained as true anomalies for the next phase.

By filtering out benign clusters, we ensured that the **decision tree model** would be trained exclusively on blocks exhibiting centralization risk, thereby enhancing both the **relevance and interpretability** of the resulting rules.

After this filtering:

- For the **0.3 threshold**, we obtained **8,370 blocks labeled as outliers** (class 1) and **36,372 as normal** (class 0).
- For the **0.2 threshold**, we obtained **7,414 blocks labeled as outliers** and **37,328 as normal**.

These labeled datasets will be used in the next phase to train a **decision tree classifier**, which will generate **interpretable rules** for guiding decentralized network behavior. A summary comparison of the two thresholds, cluster filtering, and resulting anomaly counts is provided in Table X, highlighting the differences in sensitivity and retained outlier set:

Table 4.7 compares threshold 0.2 and 0.3

Parameter	Threshold 0.2	Threshold 0.3
Number of clusters	4	9
Silhouette score	0.59	0.57
Benign Clusters Reclassified	Cluster 2	Clusters 2, 4, 6
Outliers count (after filtering)	7414 blocks	8370 blocks
Normal Blocks (after filtering)	37,328 blocks	36,372 blocks

As summarized in the table above, the filtered datasets from both thresholds now contain only those anomalies that reflect meaningful decentralization risks. With this refined and labeled data, we are now ready to proceed to the next step—**training a decision tree model** to extract interpretable rules that can guide blockchain behavior toward greater decentralization.

4.5.4 Decision tree for interpretable rules:

Following the anomaly clustering phase in previous section, we now implement the final stage of our AI pipeline: extracting human-interpretable rules from the identified centralization-risk behaviors. This phase bridges our anomaly detection results with actionable protocol improvements for the blockchain network.

We trained a DecisionTreeClassifier from scikit-learn on the filtered anomaly datasets (7,414 anomalies at the 0.2 threshold and 8,370 at 0.3) using the following configuration:

- Criterion: 'gini' impurity measure
- Max depth: 4 (optimized for balance between interpretability and performance)

The model was trained separately for each contamination threshold to compare rule strictness. The figures below present the trained decision tree for each contamination threshold.

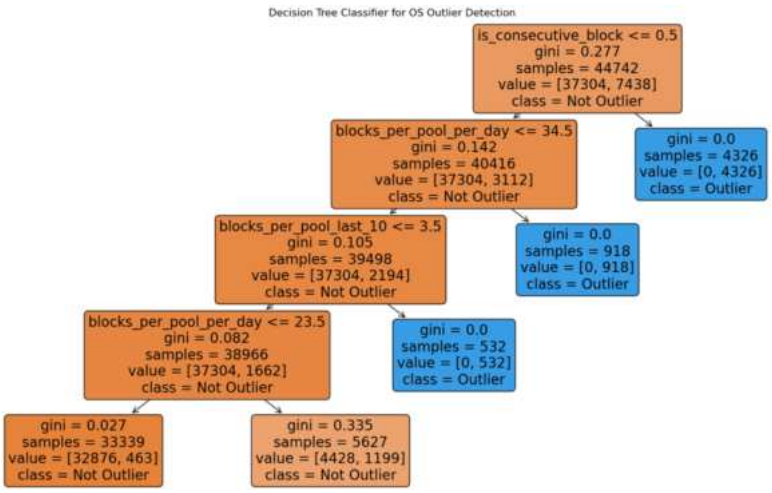


Figure 4.20 Decision tree with 0.2 threshold

The prior decision tree rules for 0.2 threshold are:

- **Consecutive Blocks**:

  IF is_consecutive_block > 0.5 → Anomaly

  (Flags all miners producing back-to-back blocks)
- **Short-Term Dominance (Last 10 Blocks)**:

  IF blocks_per_pool_last_10 > 3.5 → Anomaly

  (Allows ≤3 blocks in a 10-block window)
- **Long-Term Dominance (Daily Activity)**:

  IF blocks_per_pool_per_day > 34.5 → Anomaly

  *(Permits ≤34 blocks/day) *

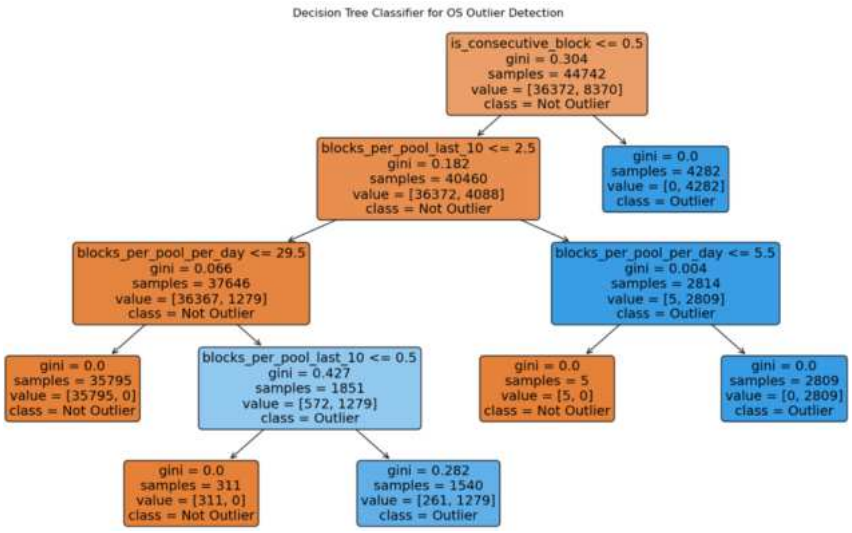


Figure 4.21 Decision tree with 0.3 threshold

The previous decision tree rulesfor 0.3 threshold are:

- **Consecutive Blocks**:

  IF is_consecutive_block > 0.5 → Anomaly

  (Same as 0.2)
- **Short-Term Dominance (Last 10 Blocks)**:

  IF blocks_per_pool_last_10 > 2.5 → Anomaly

  *(Flags ≥3 blocks in a 10-block window) *

- **Long-Term Dominance (Daily Activity)**:
  IF blocks_per_pool_per_day > 29.5 → Anomaly
  *(Tolerates ≤29 blocks/day) *

Based on the trained models, the following table represents a comparison between two sets of rules:

Table 4.8 Comparison between two sets of rules

Parameters	Threshold 0.2	Threshold 0.3
Consecutive Blocks	Immediate flag	**Immediate flag**
Short-Term Streak	3 Blocks allowed from last 10	**2 Blocks allowed from last 10**
Long-Term Streak	34 daily blocks allowed daily	**29 blocks allowed daily**

As shown above, the 0.3 threshold yields **stricter rules** for what is considered normal mining behavior, making it more suitable for enforcing tighter decentralization standards.

To evaluate the generalizability of the decision tree model, we applied the extracted rules to block data from two additional years—**2020** and **2024**. The objective was to verify whether the decision rules could improve decentralization when retrospectively applied to real data. The following figure shows the four-decentralization metrics after applying the model on 2020 year.

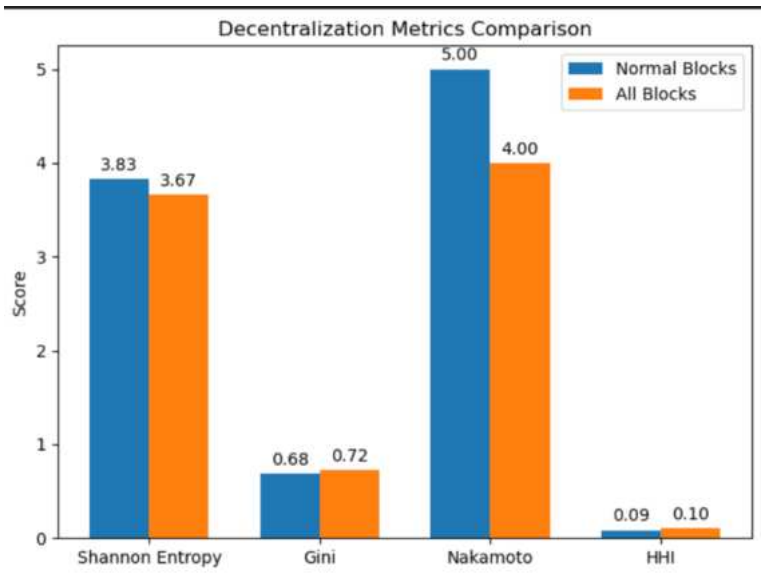


Figure 4.22 four decentralization metrics

Based on the previous figure, all four-decentralization metrics showed measurable improvement for 2020.

The following figure shows the same metrics for 2024:

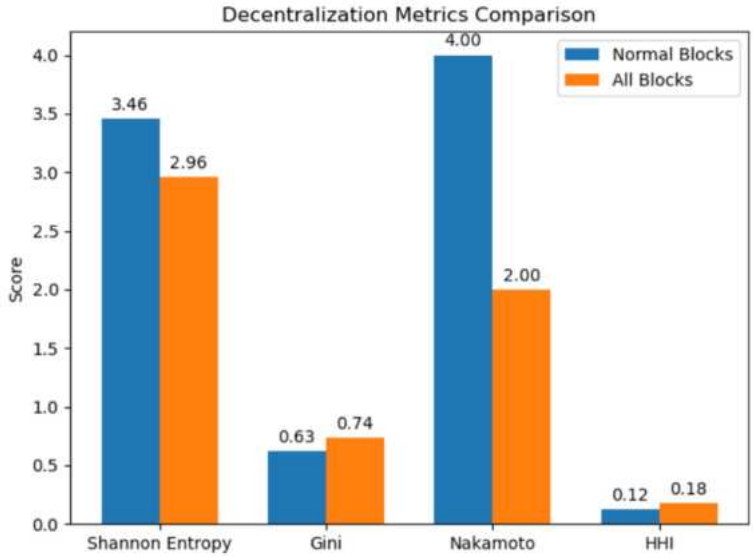


Figure 4.23 four decentralization metrics for 2024

The rules once again improved all four metrics, validating their continued relevance under evolving network conditions.

These results demonstrate that the decision tree rules—derived from historical decentralization peak data (2017)—can be **effectively applied to future datasets**, thereby reinforcing their practical utility for guiding protocol-level constraints in blockchain governance.

With the interpretable rules validated and proven effective at enhancing decentralization, we now move to the next phase: **integrating these rules into our blockchain network and simulating their impact** through controlled experiments.

4.6 Rule Integration and Simulation Setup

In this final stage of implementation, we integrate the decision rules derived from the Isolation Forest model (using the 0.3 anomaly threshold) into our custom-built blockchain network. These rules are designed to restrict miner behaviors that contribute to centralization, such as excessive block production or consecutive block wins. By enforcing the rules both during block validation and mining, we create a controlled simulation environment where their real-time impact on decentralization can be observed. The simulation enables us to compare network behavior before and after rule enforcement, providing practical evidence of their effectiveness.

4.6.1 Rule Integration into Consensus Logic

The decision rules extracted in Section 4.5.4 were embedded into our blockchain protocol in two key areas:

1. **Chain Validation:** Each node validates any incoming chain by checking that all blocks comply with the decentralization rules. If any block violates the rules (e.g., exceeding daily mining limits or producing consecutive blocks), the entire chain is rejected.
2. **Mining Enforcement:** A miner is only allowed to publish a new block if their activity does not violate the defined constraints. This proactive rule check ensures that dominant behavior is restricted at the source, not just during propagation.

By enforcing the rules at both validation and production stages, we ensure consistent network-wide compliance with decentralization goals.

4.6.2 Simulation Design and Parameters

To evaluate the effectiveness of the integrated rules, we simulate the mining of **1,000 blocks**—a quantity equivalent to approximately one week of block generation on the real Bitcoin network. The simulation involves **10 nodes**, each configured with varying computational power to reflect different mining capabilities.

The mining power is distributed as follows:

- Node 1 → 0.3
- Node 2 → 0.2
- Node 3 → 0.1
- Node 4 → 0.1
- Node 5 → 0.1
- Node 6 → 0.5
- Node 7 → 0.5
- Node 8 → 0.5
- Node 9 → 0.5
- Node 10 → 0.5

For simulation purposes, every **144 blocks** mined will represent a simulated **"day"**, allowing the AI rules (which include daily limits) to be applied realistically over a simulated one-week period.

4.6.3 Evaluation Metrics

To assess the impact of the rule-based intervention, we will compare network behavior **before and after rule integration** using the same four decentralization metrics introduced in **Section 3.6**:

- **Gini Coefficient**
- **Nakamoto Coefficient**
- **Herfindahl-Hirschman Index (HHI)**
- **Shannon Entropy**

These metrics provide quantitative insights into miner concentration, diversity, and influence, allowing for a rigorous evaluation of decentralization improvements.

4.6.4 Comparative Analysis of Simulation Results

In this subsection, we present and compare the results of the two simulation phases: one without rule enforcement (baseline) and one with AI-generated rules integrated into the consensus logic (enhanced). By analyzing the metrics across both scenarios, we aim to determine whether the applied rules effectively reduce mining concentration and improve decentralization without compromising network performance.

The following tables summarize the simulation outcomes.

Table 4.9 Summarize the simulation outcomes

Metric	Network without rules	Network with rules	Improvement
Nakamoto coefficient	1	3	**↑ +200%**
Shannon Entropy	1.83	2.87	**↑ +56.8%**
Gini Coefficient	0.716	0.423	**↓ –41.0%**
Herfindahl-Hirschman Index	0.387	0.159	**↓ –58.9%**
Total Simulation Time	39:27 min	47:06 min	**↑ ~7.6 min network delay**

Based on the previous comparison, the **Nakamoto Coefficient** increased from 1 to 3 after applying the rules, indicating a substantial improvement in network resilience. In practical terms, the number of independent entities required to compromise the network rose from just one to three, reflecting a stronger decentralization barrier. Similarly, the **Gini Coefficient** dropped from 0.716 to 0.423,

signifying a significant reduction in mining inequality. This means block production became more evenly distributed across miners, rather than being dominated by a few large participants.

The **Shannon Entropy** rose from 1.83 to 2.87, pointing to a higher level of miner diversity and less predictability in block production, which is a desirable trait in decentralized systems.

The **Herfindahl-Hirschman Index (HHI)** decreased from 0.387 to 0.159, further confirming that mining power was spread more evenly and not concentrated within a small number of entities.

Lastly, while the **total simulation time** increased slightly from 39 to 47 minutes, this trade-off is justified. The longer duration reflects stricter rule enforcement and greater miner competition, which contribute to a more secure and fair blockchain network.

4.7 conclusion

In this chapter, we successfully implemented and validated the full pipeline of our proposed system. From building the blockchain infrastructure and analyzing decentralization metrics to deploying AI-driven decision rules, each stage contributed to measurable improvements in network fairness and miner distribution. The final simulation confirmed that enforcing behavior-aware rules can mitigate centralization risks. These findings serve as a foundation for broader discussion in the concluding chapter.

Chapter 5
Conclusion

5.1 introduction

This chapter concludes our study by summarizing the key results and contributions of the proposed system. Throughout the project, we addressed the growing issue of centralization in the Bitcoin network by combining blockchain engineering with artificial intelligence. By developing a custom blockchain platform, analyzing decentralization trends, and implementing AI-driven rule extraction, we created a full pipeline capable of both detecting and mitigating centralizing behaviors. In this chapter, we reflect on the main findings, highlight the project's contributions, acknowledge limitations, and outline future research directions.

5.2 Summary of Results

The primary goal of this project was to investigate and address decentralization issues in Bitcoin through a hybrid approach combining data analysis, machine learning, and blockchain protocol design. We achieved this goal through the following stages:

- **Blockchain Network built from scratch:** We built a fully functional, Bitcoin-like blockchain network, enabling complete control over consensus logic and miner behavior simulation.
- **Bitcoin Analysis:** We used four established decentralization metrics (Gini Coefficient, Nakamoto Coefficient, Shannon Entropy, and HHI) to confirm that Bitcoin has become increasingly centralized, particularly after 2017.
- **AI Model Development:** An unsupervised anomaly detection model (Isolation Forest) was trained on Bitcoin's 2017 data to detect miner behaviors that deviate from decentralization norms. Clustering was applied to isolate harmful anomalies from benign ones.
- **Rule Extraction and Simulation:** Interpretable rules were extracted via decision tree learning and integrated into the blockchain's consensus layer. Simulation showed measurable decentralization improvements, such as increasing the Nakamoto Coefficient from 1 to 3 and reducing the Gini Coefficient from 0.716 to 0.423.

These results validate the effectiveness of AI-guided rule enforcement in promoting fairer miner participation and reducing the risk of network domination by a small group of entities.

5.3 Project Contributions

This project produced several significant contributions that can support both academic research and practical blockchain development:

- **Comprehensive Bitcoin System Analysis:** A detailed breakdown of Bitcoin's architecture, components, and workflows, which can serve as a reference model for educational and research purposes.
- **Custom Blockchain Simulation Framework:** A modular, Bitcoin-inspired blockchain network built from scratch. This framework enables experimentation with consensus logic and AI-driven protocol modifications in a fully controlled environment.
- **Empirical Centralization Study:** A decade-long (2014–2024) decentralization analysis using four established metrics (Gini Coefficient, Nakamoto Coefficient, Shannon Entropy, and HHI), evaluated at both the mining pool and unique address levels. This methodology forms a reusable framework for assessing decentralization in other blockchain networks.
- **AI Framework for Interpretable Unsupervised Learning:** The integration of unsupervised learning (anomaly detection and clustering) with interpretable model outputs provides a transparent and adaptable AI framework that can be extended to other blockchain governance or security applications.
- **AI-Powered Rule Generation:** A novel pipeline combining Isolation Forest for unsupervised anomaly detection with Decision Tree models for interpretable rule extraction, tailored to identify and address centralization risks in blockchain networks.
- **Decentralization Control Parameter:** The use of the anomaly detection threshold as a tunable parameter, enabling developers to control the desired level of decentralization and adjust consensus strictness accordingly.

Together, these contributions establish a strong, research-ready foundation for exploring AI-enhanced blockchain governance and designing more decentralized, transparent networks.

5.4 Limitations

Despite the promising results, this project has several limitations that should be acknowledged:

- **Simplified Simulation Environment:** While our custom-built blockchain provides a controlled and flexible platform for testing, it does not fully capture the complexity of real-world blockchain networks. Important factors such as network latency, asynchronous communication, adversarial behavior, and dynamic node participation (churn) were not modeled.
- **Miner Identity Evasion:** In real-world conditions, miners can easily change their public addresses to bypass rule enforcement based on miner identity. Although this was not addressed in our current implementation, a solution has been proposed in [12], where a framework is introduced to restrict miners from frequently changing their identities without detection.
- **Potential Network Bottlenecks:** There is a theoretical risk that the rules could become so strict that no miner qualifies to add a block, stalling the network. However, this limitation is mitigated by the fact

that lower-power miners—who are often sidelined in PoW environments—would now be incentivized to participate and claim rewards, thereby attracting more diverse mining activity.

These limitations provide opportunities for future improvement and represent areas where real-world deployment would require further robustness and testing.

5.6 Final Remarks

This study demonstrates the feasibility and effectiveness of combining artificial intelligence with blockchain consensus design to promote decentralization, a core principle of distributed systems. By identifying and acting upon centralization risks in a data-driven, interpretable, and enforceable manner, the system introduced in this project lays the groundwork for next-generation blockchain governance models. It also highlights the potential of AI not only as a tool for optimization but also as a guardian of fairness, transparency, and sustainability in decentralized ecosystems.

Chapter 6
References

[1] *Antonopoulos, A. M. (2017).* Mastering Bitcoin: Programming the open blockchain*. O'Reilly Media, Inc.*

[2] *Ashfaq, T., Khalid, R., Yahaya, A. S., Aslam, S., Azar, A. T., Alsafari, S., & Hameed, I. A. (2022). A machine learning and blockchain-based efficient fraud detection mechanism.* Sensors*, 22(19), 7162.*

[3] *BABUR, Sohail Mahmood, et al. Preventing 51% Attack by Using Consecutive Block Limits in Bitcoin.* IEEE Access*, 2024.*

[4] *Bashir, I. (2017).* Mastering blockchain*. Packt Publishing Ltd.*

[5] *BEIKVERDI, Alireza; SONG, JooSeok. Trend of centralization in Bitcoin's distributed network. In:* 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE, 2015. pp. 1–6.*

[6] *Buterin, V. (2013).* Ethereum white paper*. GitHub repository, 1, 22–23.*

[7] *DENG, Yuhao, et al. Outlier Summarization via Human Interpretable Rules.* Proceedings of the VLDB Endowment*, 2024, 17(7): 1591–1604.*

[8] *Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. (2008). Isolation forest. In:* 2008 Eighth IEEE International Conference on Data Mining*. IEEE, 413–42.*

[9] *Github: https://github.com/Ahmadshkeha/AI-Enhanced-Blockchain-network.git*

[10] *GRANDJEAN, Dominic; HEIMBACH, Lioba; WATTENHOFER, Roger. Ethereum proof-of-stake consensus layer: Participation and decentralization. In:* International Conference on Financial Cryptography and Data Security*. Cham: Springer Nature Switzerland, 2024. pp. 253–280.*

[11] *LAKSHMI, B. N., & Raghunandhan, G. H. (2011, February). A conceptual overview of data mining. In* 2011 National Conference on Innovations in Emerging Technology *(pp. 27–32). IEEE.*

[12] *LEE, Suhyeon; KIM, Seungjoo. Countering block withholding attack efficiently. In:* IEEE INFOCOM 2019–IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019. pp. 330–335.*

[13] *LIN, Qinwei, et al. Measuring decentralization in bitcoin and ethereum using multiple metrics and granularities. In:* 2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW)*. IEEE, 2021. pp. 80–87.*

[14] *Loh, Wei-Yin. "Classification and regression trees."* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1.1 (2011): 14–23.*

[15] *LONG, Sishan; BASU, Soumya; SIRER, Emin Gün. Measuring miner decentralization in proof-of-work blockchains.* arXiv preprint arXiv:2203.16058*, 2022.*

[16] *Mehrotra, D. (2019).* Basics of artificial intelligence & machine learning*. Notion Press.*

[17] *Nakamoto, S. (2008).* Bitcoin: A peer-to-peer electronic cash system*.*

[18] *Puthal, D., Malik, N., Mohanty, S. P., Kougianos, E., & Yang, C. (2018). The blockchain as a decentralized security framework [future directions].* IEEE Consumer Electronics Magazine*, 7(2), 18–21.*

[19] *Raj, P., Saini, K., & Surianarayanan, C. (Eds.). (2020).* Blockchain technology and applications. *CRC Press.*

[20] *RAMAKRISHNAN, Akshay Bhuvaneswari; SRIJANANI, S. Unsupervised Learning. In:* Generative AI Foundations, Developments, and Applications. *IGI Global Scientific Publishing, 2025. pp. 303–314.*

[21] *ROMITI, Matteo, et al. A deep dive into bitcoin mining pools: An empirical analysis of mining shares.* arXiv preprint arXiv:1905.05999*, 2019.*

[22] *Shayegan, M. J., Sabor, H. R., Uddin, M., & Chen, C. L. (2022). A collective anomaly detection technique to detect crypto wallet frauds on bitcoin network.* Symmetry*, 14(2), 328.*

[23] *Walfish, Steven. "A review of statistical outlier methods."* Pharmaceutical Technology*, 30.11 (2006): 82.*

[24] *XU, Rui; WUNSCH, Donald. Survey of clustering algorithms.* IEEE Transactions on Neural Networks*, 2005, 16(3): 645–678.*

[25] *YANG, Kun; KPOTUFE, Samory; FEAMSTER, Nick. An efficient one-class SVM for anomaly detection in the Internet of Things.* arXiv preprint arXiv:2104.11146*, 2021.*

[26] *ZOLOTAVKIN, Yevhen; GARCIA, Julian; LIU, Joseph K. Time-dependent decision-making and decentralization in proof-of-work cryptocurrencies. In:* 2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*. IEEE, 2019. pp. 108–10813.*