# How AI Revolutionizes Ethereum, Another View on “AI+Blockchain”

Mirror Tang, Salus & Yixin Ren, Sequoia Capital China

Over the past year, as generative AI has repeatedly surpassed public expectations, the productivity revolution led by AI has swept through the cryptocurrency world like a tidal wave. We have witnessed numerous AI concept projects create wave after wave of wealth myths in the secondary market, while an increasing number of developers have plunged into developing their own “AI+Crypto” projects.

However, upon closer inspection, it’s not hard to find that there is a widespread issue of homogeneity among these projects, and the efforts of most projects remain at the level of improving “production relations,” such as organizing computing power through decentralized networks or creating a “decentralized Hugging Face.” Very few projects genuinely integrate and innovate at the fundamental technological level. We believe this phenomenon is due to a “domain bias” between the fields of AI and blockchain. Although there is a broad intersection between these two fields, very few people deeply understand them. For example, for AI developers, delving into the technical implementation and historical infrastructure of Ethereum poses a significant challenge, let alone proposing in-depth optimization solutions.

Take machine learning, a cornerstone of the AI field, as an example. It allows machines to make decisions based on data learning without explicit programming instructions, showing tremendous potential in data analysis and pattern recognition and becoming well-known in the web2 era. However, due to the limitations of the early era, even at the forefront of blockchain technology innovation like Ethereum, its architecture, network, and governance mechanisms have yet to employ machine learning as an effective tool for solving complex problems.

We firmly believe that “great innovations often arise from the collision of cross-disciplinary fields.” The purpose of writing this article is to help AI developers gain a deeper understanding of the mysteries of blockchain, while also offering new perspectives to developers in the Ethereum community. We first explored the technical foundations of Ethereum and then proposed applying machine learning algorithms within the Ethereum network to enhance its security, efficiency, and scalability. We hope that this case will serve as an opportunity to provide a new perspective to the market, inspiring more innovation and cross-fertilization in the “AI+Blockchain” direction within the developer community.

## Technical Implementation of Ethereum

1. Basic Data Structure

The essence of blockchain is a series of interconnected blocks forming a chain, with the key distinction between different blockchains lying in their chain configuration. This is an indispensable part of the genesis block, which constitutes the initial stage of a blockchain. In Ethereum, the chain configuration not only differentiates between various chains but also marks significant upgrade protocols and key events. For example, the DAOForkBlock signifies the block height of the hard fork following the DAO attack, while the ConstantinopleBlock indicates the specific block height of the Constantinople upgrade. Major upgrades, including numerous improvement proposals, have specific fields set to denote the respective block heights. Additionally, Ethereum’s various testnets and the mainnet are uniquely identified by their ChainID.

As the starting point of the blockchain, the genesis block is the zeroth block that all other blocks directly or indirectly reference. Therefore, it’s crucial for nodes to load the correct genesis block information at startup, and this information cannot be arbitrarily modified. The configuration of the genesis block includes the aforementioned chain configuration as well as important fields such as mining rewards, timestamps, difficulty, and gas limits. It’s worth noting that Ethereum’s consensus mechanism has transitioned from Proof of Work (PoW) to Proof of Stake (PoS).

Ethereum accounts are divided into two types: external accounts, controlled exclusively by a private key, and contract accounts, which have no private key control and can only be operated through calls made by external accounts. Both have unique addresses. The world state of Ethereum is a tree made up of all Ethereum accounts, with each account corresponding to a leaf node that stores the account’s state information, including various account data and code information.

Transactions are the core of Ethereum, a decentralized platform aimed at facilitating transactions and executing contracts. An Ethereum block contains a series of packaged transactions and additional information, specifically divided into the block header and block body. The block header contains evidence that links all blocks into a chain, such as the hash of the previous block, and represents the global state of Ethereum with the state root, transaction root, receipt root, among other additional data like difficulty and nonce. The block body stores the list of transactions and the list of uncle block headers (with the transition to PoS, uncle block references no longer exist).

Transaction receipts provide the outcome and additional information post-transaction execution, offering details that cannot be directly obtained from the transaction itself. This includes consensus content, transaction details, and block information, such as whether the transaction was successful, transaction logs, and gas consumption. Analyzing receipt information facilitates debugging of smart contract code and optimization of gas consumption, serving as a confirmation that the transaction has been processed by the network, and its results and impact can be reviewed.

In Ethereum, gas fees essentially act as a transaction fee for sending tokens, executing contracts, transferring Ether, or performing other blockchain operations. These operations require gas fees because Ethereum consumes computational resources while processing transactions. Therefore, gas fees must be paid to execute these operations. Ultimately, gas fees are paid to miners as a service fee, with the cost formula being Fee = Gas Used * Gas Price, which is the actual amount of gas consumed multiplied by the gas price, set by the transaction initiator, typically influencing the transaction processing speed. If set too low, the transaction may not be executed; additionally, a gas limit must be set to prevent unexpected high gas consumption due to contract errors.

1. Transaction Pool

In Ethereum's decentralized ecosystem, despite the vast number of transactions, the transaction processing rate per second pales in comparison to centralized systems. To manage the sheer volume of transactions, nodes maintain a transaction pool for proper handling. Transactions are broadcast through the P2P network, where a node broadcasts executable transactions to its neighboring nodes, which in turn broadcast the transaction further. This process enables a transaction to rapidly spread across the entire Ethereum network within six seconds.

Transactions in the pool are categorized into executable and non-executable states, with executable transactions given higher priority to be executed and packed into blocks. Initially, all transactions entering the pool are non-executable and may later become executable. Executable and non-executable transactions are recorded in the "pending" and "queue" containers, respectively.

The transaction pool also maintains a special list of local transactions, which enjoy several advantages, such as higher processing priority, no limitations by transaction volume, and the ability to be immediately reloaded into the pool upon node restart. Local transaction persistence is achieved through a journal, ensuring unfinished local transactions can be recovered after a node restart and periodically updated.

Before entering the queue, transactions undergo legality checks, including various types of inspections such as preventing DOS attacks, ensuring transactions are non-negative, and adhering to transaction gas limits. The transaction pool is fundamentally composed of the "queue" and "pending" sections, together comprising all transactions. After passing the legality checks, further evaluations are conducted, including assessing whether the transaction queue has reached its limit, and determining if remote transactions (non-local transactions) have the lowest gas fees in the pool to replace the lowest-priced transactions. For executable transactions, by default, only those with a 10% higher gas fee are allowed to replace transactions waiting to be executed, which are then stored as non-executable. Additionally, during the maintenance of the transaction pool, invalid or over-limit transactions are removed, and transactions meeting specific criteria are replaced.

1. Consensus Algorithm

In Ethereum's early days, the consensus mechanism primarily relied on calculating the difficulty value hash. That is, by computing the hash value of a block and ensuring it meets a specific difficulty condition to validate the block's legitimacy. As Ethereum's consensus algorithm shifted from PoW to PoS, theories related to mining are no longer the focus of discussion. Here, we primarily introduce the PoS algorithm.

In September 2022, Ethereum completed the merger of the Beacon Chain and implemented the PoS algorithm. Under the PoS algorithm, the block generation time in the Ethereum network is stabilized at approximately 12 seconds. Users gain the opportunity to become validators by staking Ether, with the system randomly selecting a group of staking participants as validators. Each epoch consists of 32 slots, and in each slot, one validator is chosen as the proposer to generate a new block, while the rest of the validators form a committee responsible for verifying the legitimacy of the proposed block and reviewing the validity of blocks from the previous epoch. The PoS algorithm not only stabilizes and accelerates the speed of block generation but also significantly reduces the waste of computational resources.

1. Signature Algorithm

Ethereum inherited the signature algorithm standard from Bitcoin, adopting the same secp256k1 elliptical curve and using the Elliptic Curve Digital Signature Algorithm (ECDSA) for signing. Simply put, a complete signature contains three parts: R, S, and V. Each time a signature is generated, a random number is introduced, where R and S constitute the original output of the ECDSA. V, as the recovery field, represents the number of attempts needed to successfully recover the public key from the message content and signature. This is because, based on the R value, multiple points on the elliptical curve that satisfy the conditions may be found.

The signing process can be summarized as follows: the transaction data and relevant signature information are encoded using RLP and hashed, then the signature is completed using the private key and the ECDSA algorithm, specifically using the secp256k1 elliptical curve. Integrating the signature result with the transaction data forms a signed transaction that can be broadcasted.

In terms of data structure, Ethereum has integrated the Merkle Patricia Tree (MPT), which can efficiently store and verify a large volume of data. The MPT combines the cryptographic hash properties of the Merkle tree with the key path compression ability of the Patricia tree, offering a solution that both ensures data integrity and supports efficient querying.

1. MPT

In the Ethereum system, the Merkle Patricia Tree (MPT) plays a crucial role in storing all states and transaction data, ensuring that any change in data is directly reflected in the variation of the tree's root hash. This mechanism means that by simply verifying the root hash, we can prove the integrity and authenticity of the data without having to inspect the entire database one by one. MPT, through the interaction of four different types of nodes—leaf nodes, extension nodes, branch nodes, and null nodes—forms a tree structure that can flexibly respond to dynamic changes in data. Whenever data is updated, MPT reflects these changes by adding, deleting, or modifying nodes, and accordingly updates the tree's root hash value. Thanks to the encryption characteristics of hash functions, any minor change in data will lead to a significant variation in the root hash, thereby ensuring the high security and consistency of the data. Moreover, MPT's unique design also supports "light client" verification, enabling nodes to verify the existence or state of specific information solely based on the tree's root hash and the necessary path nodes, significantly reducing the demands on data storage and processing.

With the help of MPT, Ethereum not only improves the efficiency of data management and access speed but also strengthens network security and decentralization features, providing a solid foundation for the stable operation and continuous development of the entire Ethereum network.

1. State Machine

Ethereum's core architecture ingeniously integrates the concept of state machines, with one of its cores being the Ethereum Virtual Machine (EVM), a virtual environment for executing all smart contract code. This makes Ethereum fundamentally a global state transition system. Each block's processing is essentially a transition from one globally shared state to another. This unique design not only ensures the unity and decentralization of the Ethereum network but also ensures that the outcomes of smart contracts are predictable and tamper-proof.

In the Ethereum network, "state" refers to the record of current information for all accounts, including account balances, stored data, and smart contract code. The execution of transactions triggers the EVM to calculate the new state based on the content of the transactions and records this state transition process in a highly efficient and secure manner through the MPT. Each change in state not only updates the data of the accounts but also triggers an update in the MPT structure, reflected in changes to the root hash value.

The collaboration between the EVM and MPT is crucial, providing solid guarantees of data integrity for state transitions within the Ethereum network. When the EVM processes transactions and updates account states, the related MPT nodes are updated synchronously to reflect these changes. The hash links of each node in the MPT ensure that any change in state leads to an update of the root hash, which is then included in the new block, thereby ensuring the overall consistency and security of the Ethereum network state. Let's delve deeper into the EVM.

1. EVM

EVM is the foundational cornerstone for the execution of smart contracts and state transitions within Ethereum, enabling Ethereum to be recognized as a global computer. With its Turing completeness, the EVM endows Ethereum smart contracts with the ability to perform computations of any complexity. Furthermore, the introduction of the gas mechanism by the EVM effectively prevents infinite loops during contract execution, thereby ensuring the network's high stability and security.

On a deeper technical level, the EVM is a stack-based virtual machine that executes smart contracts through Ethereum-specific bytecode. Developers use high-level programming languages, such as Solidity, to write smart contracts and then compile them into bytecode that the EVM can recognize, enabling the deployment and execution of smart contracts. The existence of the EVM is a core driver of innovation within the Ethereum, not only ensuring the smooth operation of smart contracts but also laying a solid technical foundation for the development of decentralized applications (DApps). With the EVM, Ethereum is dedicated to building a decentralized, secure, and open digital future.

## Historical Review

[

1281×940 229 KB

](https://ethresear.ch/uploads/default/original/2X/d/d533b678b318bc138191769817edc956d1320d7c.jpeg)

Figure 1: Ethereum Historical Review

# Challenges Faced by Ethereum

## Security

Smart contracts, as computer programs on the Ethereum blockchain, have enabled developers to create a diverse array of applications, including lending platforms, decentralized exchanges, insurance, crowdfunding, social media, and NFTs, among many other areas. The security of smart contracts is particularly critical for these applications, as they directly involve the handling and management of cryptocurrencies. Thus, any vulnerabilities in smart contracts or exposure to malicious attacks could directly endanger financial security and lead to significant economic losses. For example, on February 26, 2024, the DeFi lending platform Blueberry Protocol suffered an attack due to a logical error in its smart contract, resulting in a loss of approximately 1.4 million dollars.

Smart contract vulnerabilities are varied and include issues such as flaws in business logic, improper access control, insufficient data validation, reentrancy attacks, and Denial of Service (DOS) attacks. These vulnerabilities can lead to erroneous contract execution, impacting the normal operation of smart contracts. For instance, in DOS attacks, attackers send a large volume of transactions to deplete network resources, causing transactions from regular users to not be processed in a timely manner. This not only affects user experience but can also lead to an increase in transaction gas fees. In situations of scarce resources, users may be forced to pay higher fees to have their transactions prioritized.

Additionally, Ethereum users face investment risks, with financial security under threat. For example, the term “shitcoin” is used to describe cryptocurrencies considered to have little to no value or lacking long-term growth potential. Shitcoins are often used as tools for scams or price manipulation, posing high investment risks and leading to significant financial losses. Due to their low price and market cap, shitcoins are easily manipulated, with common schemes including pump-and-dump and “honey pot” scams, where investors are lured by fraudulent projects and then robbed of their funds.

Another common risk is the “Rug Pull,” where project creators suddenly remove all liquidity, causing the token value to plummet. These scams are usually promoted through false partnerships and endorsements, and once the token’s value rises, scammers sell off their tokens, take the profits, and disappear, leaving investors with worthless tokens. Investing in shitcoins also diverts attention and resources away from legitimate cryptocurrencies with real-world applications and growth potential.

Besides shitcoins, “air coins” and “ponzi coins” also represent high-risk investments for quick profits. For users lacking sufficient professional knowledge and experience, distinguishing them from legitimate cryptocurrencies is particularly challenging.

## Efficiency

Two key indicators of Ethereum’s efficiency are transaction speed and gas fees. Transaction speed indicates the number of transactions the Ethereum network can process in a given unit of time, serving as a direct measure of the network’s processing capacity, where faster speeds denote higher efficiency. Additionally, every transaction requires the payment of a certain amount of gas fees as compensation to miners for transaction verification. Lower gas fees thus imply greater network efficiency.

A decrease in transaction speed often leads to an increase in gas fees. Typically, as the efficiency of transaction processing drops and the competition for limited block space intensifies, the number of transactions vying for entry into the next block surges. To ensure their transactions are processed preferentially, users may find themselves compelled to increase the gas fees they offer. Miners tend to prioritize transactions with higher gas fees for confirmation. As a result, an increase in gas fees can deteriorate the user experience.

Transactions are just one of the foundational activities within the Ethereum ecosystem. Users can engage in a variety of activities such as lending, staking, investing, and insurance through specific DApps. However, the vast array of DApps, combined with a lack of personalized recommendation services, often leaves users confused about choosing the applications and products best suited to their needs. This not only reduces user satisfaction but also impacts the overall efficiency of the Ethereum ecosystem.

Take DeFi lending platforms as an example. To ensure platform security and stability, they often employ an over-collateralization mechanism, requiring borrowers to pledge assets exceeding the loan amount as collateral. This restricts borrowers from freely using these assets during the loan period, thereby reducing the efficiency of capital utilization and market liquidity.

# Application of Machine Learning in Ethereum

In the Ethereum ecosystem, numerous machine learning models, including the RFM (Recency, Frequency, Monetary value) model, Generative Adversarial Networks (GANs), decision trees, the K-Nearest Neighbors (KNN) algorithm, and the Density-Based Spatial Clustering of Applications with Noise algorithm (DBSCAN), play a crucial role. The integration of these models not only provides a powerful boost to optimizing transaction processing efficiency but also significantly enhances the security level of smart contracts. Furthermore, by enabling refined user segmentation, they facilitate the provision of customized services and solidify the stable operation of the network.

## Algorithm Introduction

Machine learning algorithms comprise a series of instructions or rules designed to parse data, learn patterns within it, and make predictions or decisions based on these learnings. What sets these algorithms apart is their ability to automatically learn and improve from the provided data without relying on explicitly programmed instructions by humans. Within the Ethereum ecosystem, machine learning models like the RFM model, GANs, decision trees, KNN algorithm, and DBSCAN algorithm play a pivotal role. The application of these models enhances the efficiency of transaction processing, elevates the security level of smart contracts, enables precise user segmentation to lay a solid foundation for providing more personalized services, and contributes to the stability of the network's operation.

1. Bayesian Classifier

The Bayesian classifier is a type of statistical classification method aimed at minimizing the probability of classification errors or minimizing the average risk within a specific cost framework. The design philosophy of this classifier is deeply rooted in Bayes' theorem, enabling it to calculate the probability that an object belongs to a certain category based on known characteristics and make classification decisions accordingly. Specifically, the Bayesian classifier first considers the prior probability of the object and then uses Bayes' theorem to consider observed data comprehensively, thereby updating its judgment on the object's category. Among all possible classifications, the Bayesian classifier selects the category with the highest posterior probability for the object.

As shown in Figure 2, in supervised learning scenarios, the Bayesian classifier makes classification decisions using data and a probability model based on Bayes' theorem. It calculates the posterior probability of each class for the data points, based on the likelihood of the data points, categories as well as the prior probabilities of features, and then assigns the data points to the category with the highest posterior probability. In the scatter plot of Figure 2, the classifier aims to find a boundary line that best separates points of different colors, thereby minimizing classification errors.

[

944×776 273 KB

](https://ethresear.ch/uploads/default/original/2X/9/9c081e78cd99d22c71680954bbf516e59f806ae7.png)

[

1000×352 81.1 KB

](https://ethresear.ch/uploads/default/original/2X/4/4cc8830f060078828121aa009948eb5630f7e54e.png)

Figure 2: Bayesian Classifier

1. Decision Tree Algorithm

The decision tree algorithm is a widely used machine learning method for classification and regression tasks. It operates on a hierarchical decision-making logic, branching out by selecting features with higher information gain rates to gradually build a decision tree model. In essence, the algorithm automatically learns decision rules from data to determine the values of variables. In its concrete implementation, a decision tree decomposes a complex decision-making process into several simpler sub-decision processes, each based on the decision criteria of its predecessor, thus forming a tree-like structure.

As illustrated in Figure 3, each node in the decision tree represents a decision criterion for a certain attribute, while the branches indicate the outcomes of that decision. Each leaf node corresponds to the final predicted result or category. In terms of structure, the decision tree model is intuitive and easy to understand, offering strong interpretability.

[

1400×713 122 KB

](https://ethresear.ch/uploads/default/original/2X/f/ff5b91d982ed5ff0b8f823ef66fedb6b1ead643f.jpeg)

Figure 3: Decision Tree

1. DBSCAN Algorithm

The DBSCAN algorithm is a density-based spatial clustering method for datasets with noise, particularly suited for discontinuous datasets. It can identify clusters of various shapes without the need to predefine the number of clusters. Moreover, the DBSCAN algorithm is capable of effectively identifying outliers, specifically those located in low-density regions, enhancing its applicability in complex data environments, as shown in Figure 4.

[

740×721 207 KB

](https://ethresear.ch/uploads/default/original/2X/a/a0734bd262bb3c089dae2e065b62d48438f5efe6.png)

Figure 4: The DBSCAN Algorithm Identifies Noise

1. KNN Algorithm

The KNN algorithm is suitable for both classification and regression tasks. In classification, it uses a voting mechanism to determine the category of the target instance; for regression, it predicts by calculating the average or weighted average of the closest k neighbors.

As illustrated in Figure 5, the core logic of the KNN algorithm in classification tasks is to identify the nearest K neighbors to a new data point and predict the category of the new data point based on the categories of these neighbors. When K=1, the new data point is simply assigned to the same category as its closest neighbor. If K>1, the category of the new data point is usually determined by a majority vote, meaning it is assigned to the category most common among its neighbors. When applied to regression tasks, the KNN algorithm follows the same principle of proximity, but the prediction is made based on the average output values of the K closest samples.

[

1096×832 168 KB

](https://ethresear.ch/uploads/default/original/2X/9/90d50bd706586c91db9a88751d5b40f012d1b160.png)

Figure 5: KNN Algorithm for Classification

1. Generative Artificial Intelligence

Generative Artificial Intelligence is an advanced AI technology capable of producing new content, such as text, images, and music, based on input requirements. The development of this technology has been propelled by continuous advancements in machine learning and deep learning, particularly in applications within natural language processing and image recognition. By analyzing large datasets, generative AI models learn patterns and associations, allowing them to generate entirely new outputs based on this acquired knowledge. The effectiveness of model training is crucial in generative AI, necessitating high-quality data for learning and training purposes. Throughout this process, models deeply analyze and understand the structure, patterns, and relationships within the dataset, progressively enhancing their capability to create new content.

1. Transformer

As a cornerstone in the field of generative artificial intelligence, the Transformer model has revolutionized information processing by introducing the attention mechanism, which enables both focused precision and a holistic overview, particularly demonstrating exceptional performance in text generation. With advanced natural language processing models, such as GPT, Transformers can accurately interpret user requirements expressed in natural language and convert them into executable code, significantly reducing development complexity and enhancing efficiency.

As shown in Figure 6, the performance of Transformer in the generation task of natural language processing has been significantly improved by integrating multi-head attention mechanism and self-attention mechanism, combining residual connection and fully connected neural network, and utilizing word embedding technology.

[

407×600 79 KB

](https://ethresear.ch/uploads/default/original/2X/1/19cd90c6286fa65d63c7c9acf76ccf64c127f24a.png)

Figure 6: Transformer Model

1. RFM Model

The RFM model is an analytical tool that utilizes data on customer transaction behavior to identify groups of users with varying levels of value. This model segments customers based on three key metrics: the Recency of the customer's last purchase, the Frequency of purchases, and the Monetary value of those purchases.

As shown in Figure 7, these three metrics form the foundation of the RFM model. By quantitatively scoring customer performance across these dimensions, the model ranks customers, thereby identifying those who are of the highest value. Furthermore, the RFM model facilitates precise customer segmentation, providing businesses with a solid basis for developing personalized marketing strategies and optimizing customer service.

[

780×780 11.3 KB

](https://ethresear.ch/uploads/default/original/2X/2/2af1c8a69e245482e088550c7f1289a0f0a3c683.png)

Figure 7: RFM Model

## Possible Applications

In exploring machine learning techniques to address Ethereum's security challenges, we focus on research in four key areas:

1. Identifying and Filtering Malicious Transactions with Bayesian Classifiers

By deploying Bayesian classifiers, we can effectively identify and filter potential malicious transactions, especially those frequent, small-scale transactions that could lead to DOS attacks. This approach relies on a detailed analysis of transaction characteristics, such as gas prices and transaction frequency, ensuring the health and stability of the Ethereum network.

1. Generating Smart Contract Codes that Meet Security Standards and Specific Requirements

GANs and Transformer-based generative models can both produce smart contract codes that meet specific requirements and have high security. These two methods have different focuses in the data types they rely on for training: the former primarily learns from unsafe code samples, while the latter tends to focus on safe samples.

Training GANs to learn existing safe contract patterns and self-generate potential unsafe code samples allows the model to further learn to identify these potential risks, achieving the goal of automatically generating high-quality smart contract codes with higher security levels. Using Transformer-based generative models, by analyzing and learning from a large number of safe contract examples, it is possible to generate contract codes that are safe and meet specific needs, such as optimized gas consumption, significantly improving the efficiency and safety of smart contract development.

1. Risk Analysis of Smart Contracts with Decision Trees

Utilizing decision tree models to analyze key features of smart contracts, such as the frequency of function calls, transaction values, and the complexity of the source code, can effectively identify potential risks in contracts. This method enables us to predict potential vulnerabilities and risk points in contracts, providing developers and users with accurate safety assessments and greatly enhancing the overall security of smart contracts in the Ethereum ecosystem.

1. Building Cryptocurrency Evaluation Models to Reduce Investment Risks

By integrating analysis of cryptocurrency transaction records, social media activities, and market performance using machine learning algorithms, we can construct an evaluation model that predicts the risk of cryptocurrencies. This model can help investors avoid potential investment risks, thereby promoting the healthy development of the cryptocurrency market.

Furthermore, the application of machine learning also has the potential to further enhance the efficiency of Ethereum, which can be explored from three dimensions:

1. Application of Decision Trees in Optimizing the Transaction Pool Queuing Mechanism

Through decision tree models, we can effectively optimize the queuing strategy of the Ethereum transaction pool. By analyzing transaction characteristics such as gas prices and transaction sizes, decision trees help optimize the selection and ordering of transactions, significantly increasing transaction processing speed, reducing network congestion, and lowering wait times for users.

1. Segmenting Users and Providing Customized Services

As a widely used analysis tool in customer relationship management, the RFM model effectively segments users by evaluating their most recent transaction time, transaction frequency, and the monetary value of transaction. Applying the RFM model to the Ethereum platform can help identify high-value user groups, optimize resource allocation, and provide more customized services, thereby enhancing user satisfaction and platform efficiency.

The DBSCAN algorithm assists in identifying different user groups on Ethereum by analyzing user transaction behavior, offering them more customized financial services. This strategy of user segmentation helps optimize marketing effectiveness, improve customer satisfaction, and service efficiency.

1. Credit Scoring with KNN

The KNN algorithm performs credit scores on users by analyzing their transaction history and behavior patterns, which plays a key role in activities such as financial lending. Credit scores provide financial institutions and lending platforms with an effective tool to assess borrowers' repayment abilities and credit risks, making loan decisions more accurate. This not only helps avoid excessive borrowing but also enhances market liquidity.

## Future Directions

From the perspective of macro financial allocation, considering Ethereum as the world's largest distributed computing platform, investment in its infrastructure is undoubtedly crucial. Thus, attracting developers with diverse backgrounds to jointly participate in construction has become an urgent need. This article, through a detailed analysis of Ethereum's technical construction and the challenges it faces, discusses a series of machine learning application scenarios and looks forward with anticipation to AI developers transforming these ideas into tangible value.

As the computational power on the blockchain continues to increase, we anticipate the development of more complex models to aid in network management, transaction monitoring, and security audits, thereby enhancing the efficiency and security of the Ethereum network.

Moreover, governance mechanisms driven by artificial intelligence/smart agents are expected to become an innovative highlight within the Ethereum ecosystem. Such governance mechanisms will lead to a more efficient, transparent, and automated decision-making process, providing the Ethereum platform with a more flexible and reliable governance structure. These potential developments not only may propel continuous innovation in Ethereum technology but also offer users an improved on-chain experience.

If you are an expert in artificial intelligence or Ethereum technology and are interested in further discussion or collaborative research on this topic, please contact us.

- The current AI Chains all scam
- AI + blockchain is a scam
- AI + blockchain change the world

0

voters