mathematics

MDPI

Article

Enabling High-Quality Machine Learning Model Trading on Blockchain-Based Marketplace

Chunxiao Li [1], Haodi Wang [1], Yu Zhao [1], Yuxin Xi [1], Enliang Xu [2] and Shenling Wang [1,*]

[1] School of Artificial Intelligence, Beijing Normal University, Beijing 100875, China; lichunxiao@bnu.edu.cn (C.L.); whd@mail.bnu.edu.cn (H.W.); zhaoyu2022@mail.bnu.edu.cn (Y.Z.); xiyx@mail.bnu.edu.cn (Y.X.)
[2] School of Data Science and Artificial Intelligence, Dongbei University of Finance and Economics, Dalian 116025, China; enliangxu@dufe.edu.cn
* Correspondence: slwang@bnu.edu.cn

Abstract: Machine learning model sharing markets have emerged as a popular platform for individuals and companies to share and access machine learning models. These markets enable more people to benefit from the field of artificial intelligence and to leverage its advantages on a broader scale. However, these markets face challenges in designing effective incentives for model owners to share their models, and for model users to provide honest feedback on model quality. This paper proposes a novel game theoretic framework for machine learning model sharing markets that addresses these challenges. Our framework includes two main components: a mechanism for incentivizing model owners to share their models, and a mechanism for encouraging the honest evaluation of model quality by the model users. To evaluate the effectiveness of our framework, we conducted experiments and the results demonstrate that our mechanism for incentivizing model owners is effective at encouraging high-quality model sharing, and our reputation system encourages the honest evaluation of model quality.

Keywords: model sharing; MLaaS; incentive mechanism; smart contract

MSC: 91-05

Citation: Li, C.; Wang, H.; Zhao, Y.; Xi, Y.; Xu, E.; Wang, S. Enabling High-Quality Machine Learning Model Trading on Blockchain-Based Marketplace. *Mathematics* **2023**, *11*, 2636. https://doi.org/10.3390/math11122636

Academic Editor: Jakub Nalepa

Received: 11 May 2023
Revised: 29 May 2023
Accepted: 1 June 2023
Published: 9 June 2023

1. Introduction

Machine learning has revolutionized a multitude of industries, from face recognition [1] to automated vehicles [2] and disease diagnosis [3,4]. For instance, in the healthcare sector, machine learning algorithms have played a pivotal role in medical image analysis, enabling more accurate diagnoses and personalized treatment plans. Furthermore, the financial industry has embraced machine learning for fraud detection, where sophisticated models can identify fraudulent transactions in real-time, preventing substantial financial losses. However, training a high-quality machine learning model requires a large amount of data. For example, in the training process of an image recognition model, tens of thousands of images are needed, and these images need to be labeled. Additionally, during the training process, high-performance GPUs are also required to complete the training tasks over a shorter amount of time. This can limit the ability of smaller companies to benefit from machine learning technology. ML-as-a-service (MLaaS) [5] has emerged as a solution, led by industry giants such as Google, Amazon, and Microsoft, making machine learning accessible to a broader range of users. By easing the need for training models, users can now enjoy the benefits of machine learning services without the need for extensive infrastructure or data models.

Moreover, sharing machine learning models has the potential to encourage innovation and discovery by allowing others to build upon existing models and to develop new approaches. This can lead to further improvements in machine learning applications and

accelerate progress in various fields [6]. Despite the potential benefits of machine learning model sharing, challenges remain in incentivizing model owners to share high-quality models and to ensure the honest evaluation of model quality by users. A decentralized game theoretic framework based on blockchain [7,8] that includes mechanisms for incentivizing model owners and encouraging honest evaluation by users can address these challenges and improve the efficiency and effectiveness of model sharing markets [9–11].

In summary, machine learning has a tremendous potential to transform numerous industries and to make significant improvements to our lives. The emergence of MLaaS has made machine learning technology more accessible to a broader range of users, and sharing machine learning models can spur innovation and progress. However, challenges remain in designing effective mechanisms for incentivizing high-quality model sharing and honest evaluation, which can be addressed through the development of game theoretic frameworks.

Developing a machine learning model sharing platform can be a complex undertaking, and several challenges need to be addressed to ensure the platform's success. Firstly, ensuring the quality of models is critical. To achieve this, a robust vetting process is necessary to verify the accuracy and reliability of models uploaded to the platform. This will require a team of experts to review the models and to assess their quality, which can be time-consuming and costly. Secondly, developing an effective incentive mechanism is crucial to encourage model owners to upload their high-quality models to the platform. This can be accomplished by offering a revenue-sharing model that rewards model owners for high-quality models and penalizes them for low-quality ones. The revenue-sharing model can incentivize model owners to upload their best models and to promote a culture of quality sharing. Lastly, an efficient execution application is necessary to evaluate machine learning model quality and to allocate rewards automatically. This will require the development of an automated evaluation process that can assess the accuracy and reliability of models and allocate rewards based on the results. This process can be challenging, given the complexity of machine learning algorithms, but it is essential to ensure the platform's success. In conclusion, developing a machine learning model sharing platform requires addressing several challenges, including ensuring model quality, developing an effective incentive mechanism, and implementing an efficient execution application. Overcoming these challenges will be crucial to the success of the platform and will help promote a culture of quality sharing in the machine learning community.

Main Contributions

The main contributions of this paper are as follows: (1) We propose an honest evaluation incentive mechanism based on the EM algorithm and the sigmoid-prime function. Using the evaluation data provided by users, the EM algorithm is introduced to evaluate the model quality. We measure the honesty of evaluations using the deviation between the evaluation provided by users and the model quality evaluation result, and we propose an incentive function based on the sigmoid-prime function to encourage users to provide honest evaluations. (2) We propose a quality and similarity-driven dynamic incentive model for model sharing. Traditional incentives focus more on the quantity of the models shared, with no differentiation in terms of content similarity and quality, which can easily induce model owners to share low-quality, homogeneous models. Therefore, based on the sigmoid function, we propose an incentive function driven by model quality and similarity, and realize the dynamic adjustment of the incentive function by constructing an evolutionary game model, to encourage model owners to share high-quality, innovative models. (3) Our proposed solution is built upon blockchain technology, enabling a decentralized approach without the need for intermediaries. Every step of the process is conducted openly and transparently, with the added advantage of automated execution through smart contracts.

2. Related Work

MLaaS: MLaaS is an emerging technology that provides machine learning services over the internet, allowing users to utilize machine learning algorithms and models without the need for in-house development and maintenance. In recent years, MLaaS has gained significant attention and has become increasingly popular due to its potential to democratize machine learning and to make it more accessible to businesses and individuals alike. Freenome provides AI-powered cancer screening and diagnosis services, while SoundHound uses natural language processing and machine learning to provide voice-enabled AI assistants. Despite the benefits of MLaaS, there is one drawback to consider. When users upload their data to MLAAS for prediction, there is a risk that the platform may steal their data, leading to potential privacy breaches. The authors in [12] propose a decentralized model marketplace based on blockchain technology for Machine Learning as a Service (MLaaS) platforms. The proposed system aims to address the limitations of centralized MLaaS platforms, including data privacy and security concerns. However, evaluating model quality in this paper requires a benchmark dataset, which can be very difficult for a model sharing market where there are many models. Building a testing dataset for every model is a challenging task.

Model Evaluation: The value of a machine learning model depends on its quality, with a higher prediction accuracy indicating a higher quality and value. Currently, there are two methods for evaluating models. The first involves constructing a standardized test dataset with labeled data, and evaluating the model's quality based on its accuracy in predicting the test data [13–15]. This approach requires a large amount of data, which can be time-consuming and even impossible for a shared market with numerous types of models, and may raise privacy concerns. The second method for evaluating models involves using user feedback, which is often employed in crowdsourcing systems [16–18]. By allowing users to test the model and provide feedback, its quality can be assessed. However, current research lacks incentives for user feedback, leading to low user engagement and even malicious feedback. The authors in [19,20] utilize formal approaches to verify machine learning systems; however, these approaches do not directly evaluate the quality of the machine learning models.

3. Problem Formulation

3.1. System Overview

Figure 1 illustrates the system architecture of the model sharing platform, which comprises four entities: the model user, the model owner, blockchain, and IPFS. Model owners have large amounts of data and expertise to train their machine learning models and decide whether to share them in the market. Model owners can receive rewards from users who utilize their models and benefit from the models. Model users have data to predict, but they do not have their own machine learning models. They can search for models that satisfy their requirements from the market, and pay a deposit before downloading and evaluating the model. The blockchain serves as a mediator between model owners and model users. It collects models from model owners and queries requests from model users, and matches them. It should be noted that model owners can also purchase models that are shared by others and then become model users in this scenario. The InterPlanetary File System (IPFS) is a decentralized protocol and network that is designed for storing and sharing files in a distributed manner. Unlike traditional file systems that rely on centralized servers, IPFS operates on a peer-to-peer network, where files are addressed based on their content and stored across multiple nodes. Because blockchain is not suitable for storing large-capacity files, the model is stored on IPFS. Overall, the workflow of our design is shown in Figure 1 as follows:

- **Share Models**: Model owners query reward information and existing models in the market through the platform, and then decide whether to share their own models. If they choose to do so, they upload the model to IPFS and add a description of the model to the market.

- **Search Models**: When model users search for the desired model on the platform and find it, they will be required to pay a deposit to the platform before they can proceed to download the model.
- **Evaluate Models**: When a model user completes a prediction task using a model shared by others, they are required to evaluate the model.
- **Update Reputation**: When the smart contract receives a threshold number of evaluations from users, it calculates the actual quality of the computational model. Based on the deviation between the user evaluations and the actual quality, the smart contract calculates the user's reputation value.

For clarity of presentation, we omit the process of user registration and authentication.

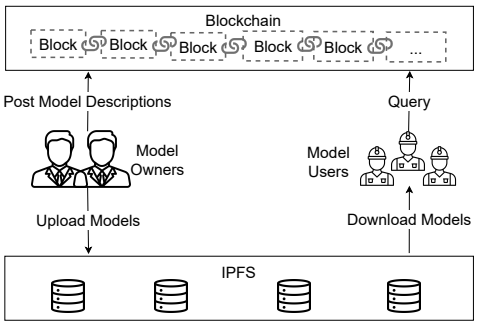


Figure 1. The overview of blockchain-based model sharing system.

3.2. Background

Gaussian Mixture Model: A Gaussian Mixture Model (GMM) is a probabilistic model used for clustering and density estimation tasks. It assumes that the data are generated from a mixture of several Gaussian distributions, each with its own mean and covariance matrix. The parameters of the model include the mixing coefficients, which represent the probabilities of each Gaussian distribution being responsible for generating a particular data point. The goal of fitting a Gaussian mixture model is to estimate the parameters of the model that maximize the likelihood of the observed data. This can be achieved using the expectation-maximization (EM) algorithm, which iteratively updates the model parameters until convergence. Once the Gaussian mixture model has been fitted to the data, it can be used for various tasks such as clustering and density estimation. The model can assign each data point to one of the Gaussian distributions in the mixture, allowing it to be used for clustering. It can also be used to estimate the probability density function of the data, which can be useful for anomaly detection or for generating new samples from the same distribution. In our proposed solution, the EM algorithm is utilized to classify honest evaluations and malicious evaluations.

Blockchain and smartcontract: Blockchain is a decentralized, distributed ledger that records transactions on a network [21–26]. Each block in the chain contains a set of transactions that have been verified by network participants, and once added to the chain, the transaction cannot be altered. This makes blockchains highly secure and transparent, as all transactions are visible to all participants on the network [27–30]. Smart contracts are self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code. These contracts are stored on a blockchain and can be automatically executed when certain conditions are met [31–34]. Smart contracts can be used for a wide range of applications, from financial transactions to supply chain management [35–37]. Together, blockchain and smart contracts can be used to create secure and transparent systems for a wide range of applications, from financial transactions to supply chain management, and beyond [38]. The decentralized nature of blockchain and the automated execution of smart contracts make these technologies highly efficient and cost-effective, with the potential to revolutionize many industries. In our proposed solution, smart contracts automatically collect user evaluations and calculate the true quality of the model.

Evolutionary Game: Evolutionary game theory is a branch of game theory that studies how populations of individuals, each following a particular strategy, interact over time, and how their strategies evolve based on their past successes and failures [39–42]. In an evolutionary game, individuals are represented by strategies that determine their behavior in a given situation. Each individual has a fitness value, which represents their success in terms of survival or reproduction, and this value is influenced by the strategies of other individuals in the population [43]. Over time, the strategies that result in higher fitness values become more prevalent in the population, while those that result in lower fitness values become less prevalent. This process of natural selection can lead to the emergence of stable equilibria, where the strategies of all individuals in the population are in balance and no individual can improve its fitness by changing its strategy. In our proposed solution, evolutionary game theory is used to analyze the cooperative and competitive relationships among model sharers.

4. System Design

In this section, we provide a detailed overview of the design of our blockchain-based system for sharing machine learning models. We first discuss our approach for incentivizing users to provide honest feedback on models, as opposed to maliciously leaving negative comments. Next, we propose solutions for motivating model owners to share high-quality models.

4.1. Honest Evaluation Incentive Model

The honest evaluation incentive stage can be divided into three steps. The first step is to collect user evaluations, and when the number of evaluations reaches a threshold, the system proceeds to the second step. In the second step, the collected evaluations are classified into honest and malicious evaluations, and the model quality is calculated using the honest evaluations. In the third step, the system calculates the user's reputation value based on the deviation between the model quality and the user's evaluation. In our proposed system, model users may either provide honest or malicious feedback. Due to the anonymity and lack of trust in blockchain-based systems, the possibility of collusion between model owners and model users is low, thereby reducing the likelihood of malicious praise. However, model users may also be potential model owners themselves, and therefore, there is a high probability of providing poor evaluations for other models in order to improve the competitiveness of their own models. Therefore, while considering the evaluation behaviors of malicious model users, we only take into account malicious negative comments and we do not consider malicious favorable comments for the time being. In general, the evaluation data are expected to follow a normal distribution [44]. For the model quality evaluation phase, we consider two possible states of the model users—honest or malicious—and treat the evaluation data as a combination of data that follow a normal distribution with different parameters. We use the Gaussian mixture model to fit the evaluation data. The problem of determining the model quality is then transformed into finding the expectation of the normal distribution that fits the honest evaluation data, which can be abstracted as a parameter estimation problem of the Gaussian mixture model. We use the EM algorithm to classify honest evaluations and malicious negative comments, and we use the mean of the honest evaluations as the model quality evaluation result. In the honest evaluation incentive phase, we use reputation as the incentive carrier and we measure the honesty of the evaluation behavior by calculating the deviation between the model user's evaluation and the model evaluation result. We design an evaluation incentive function based on the sigmoid-prime function to provide feedback and targeted incentives for the model user's evaluation behavior. The sigmoid function, also known as the logistic function, is a mathematical function that maps input values to a range of between 0 and 1. The sigmoid function has the property where it has the maximum slope at x = 0, meaning that the rate of change of the y-value is higher. As x increases or decreases, the rate of change of the y-value gradually decreases. Based on this characteristic, the

reputation value of users undergoes significant changes in the initial state, but as the system stabilizes, the rate of change of the reputation value decreases. The more honest the model user is, and the closer the evaluation is to the evaluation result, the more reputation value they can obtain.

The proposed system estimates model quality based on honest evaluations, but it needs to determine whether the user's evaluation is honest, since the system cannot identify it directly. Assuming that there are more honest model users than malicious users, the distribution with a higher mixing coefficient in the Gaussian mixture model can be considered as the distribution followed by the evaluation data of honest model users. Its parameters can represent the quality of models. The problem of estimating model quality based on comments can be transformed into a parameter problem of solving the Gaussian mixture model. The EM algorithm is a commonly used algorithm for solving the Gaussian mixture model. In the following section, we will describe how the EM algorithm is used to estimate the quality of models.

Let $\mathcal{M}$ be the set of models, m_i be the i-th model, and MC^i be the comments set of m_i. The MC^i can be divided into two groups; the first group of comments are from honest model users and the second group of comments are from malicious model users. We denote the first group of comments as MC_1^i; these comments fit the normal distribution N_1^i, whose mean value is μ_1^i, and the variance is σ_1^i. We denote the first group of comments as MC_2^i; these comments fit the normal distribution N_i^2, whose mean value is μ_2^i, and the variance is σ_2^i and $MC_1^i + MC_2^i = MC^i$. The length of MC^i equals the number of m_i users, that is, each user can only evaluate a mode once, and must evaluate once. We denote the coefficients of N_1^i and N_2^i in the Gaussian mixture model as α_1^i,α_2^i, representing the proportion of data from N_1^i and N_2^i in the total data; we can obtain $\alpha_1^i + \alpha_2^i = 1$. Generally, honest users are more common than malicious users, i.e., $0 \leq \alpha_2^i \leq 0.5 \leq \alpha_1^i \leq 1$. The probability distribution function of the Gaussian mixture model can be expressed as:

$$\mathrm{P}\left(MC^i|\theta\right) = \sum_{k=1}^{2} \alpha_k^i \phi_k^i \left(MC^i, \mu_k^i, {\sigma_k^i}^2\right) \tag{1}$$

In Equation (1), θ is a model parameter, and $\theta = \left(\alpha_k^i, \mu_k^i, {\theta_K^i}^2\right), k \in 1, 2$. ϕ_k^i is the k-th normal distribution $N_k^i\left(\mu_k^i, {\sigma_k^i}^2\right)$.

$$\phi_k^i(MC^i, \mu_k^i, {\sigma_k^i}^2) = \frac{1}{\sqrt{2\pi}\theta_k^i} e^{-\frac{(MC^i - \mu_k^i)^2}{2{\sigma_k^i}^2}} \tag{2}$$

We define hidden variables as variables that cannot be directly observed but can be derived from other variables. We do not know whether the user is honest; we can calculate the probability that the user belongs to each state. So, the state of the model user is a hidden variable. We define the state of a model user as $Z = \{z_{ik} | i \in [1, n], k \in [1, 2], \Sigma_{k=1}^2 Z_{ik} = 1\}$

$$\begin{aligned} \mathrm{P}\left(\mathrm{SC}^1, \mathrm{Z}; \theta\right) &= \prod_{\mathrm{i}=1}^{\mathrm{n}} \prod_{\mathrm{k}=1}^{2} \left[\alpha_\mathrm{k} \varphi_\mathrm{k}\left(\mathrm{sc}^1; \mu_\mathrm{k}, \sigma_\mathrm{k}^2\right)\right]^{\mathrm{z_{ik}}} \\ &= \prod_{\mathrm{k}=1}^{2} \alpha_\mathrm{k}^{\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{z_{ik}}} \prod_{\mathrm{i}=1}^{\mathrm{n}} \left[\frac{1}{\sqrt{2\pi}\sigma_\mathrm{k}} \mathrm{e}^{-\frac{\left(\mathrm{sc}^1 - \mu_\mathrm{k}\right)^2}{2\sigma_\mathrm{k}^2}}\right]^{\mathrm{z_{ik}}} \end{aligned} \tag{3}$$

In order to reduce the computational complexity and to eliminate irrelevant variables, take the logarithm of the left and right sides of Equation (3) at the same time and obtain the log-likelihood function of the complete data as follows:

$$\log \mathrm{P}\left(\mathrm{SC}^1, \mathrm{Z}; \theta\right) = \sum_{\mathrm{k}=1}^{2} \left\{ \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{z_{ik}} \log \alpha_\mathrm{k} + \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{z_{ik}} \left[\log\left(\frac{1}{\sqrt{2\pi}}\right) - \log \sigma_\mathrm{k} - \frac{\left(\mathrm{sc}_\mathrm{i}^1 - \mu_\mathrm{k}\right)^2}{2\sigma_\mathrm{k}^2}\right] \right\} \tag{4}$$

The EM algorithm is an iterative algorithm, and the input of each iteration is the output of the previous iteration. Assuming that we are at the $(m+1)$th iteration and need to solve for the parameter values, we can use θ^m to represent them. Based on the existing evaluation data SC^l and parameter estimation value θ^m, the conditional expectation of the complete data log-likelihood function $Q(\theta, \theta^m)$ can be constructed as follows:

$$\begin{aligned} Q(\theta, \theta^{m}) &= E\left(\log P(SC, Z; \theta) \mid SC^{l}, \theta^{m}\right) \\ &= \sum_{k=1}^{2}\sum_{i=1}^{n} E\left(z_{ik} \mid SC^{l}, \theta^{m}\right) \log \alpha_{k}^{m} + \sum_{k=1}^{2}\sum_{i=1}^{n} E\left(z_{ik} \mid SCC^{1}, \theta^{m}\right) \\ &\quad \left[\log\left(\frac{1}{\sqrt{2\pi}}\right) - \log \sigma_{k}^{m} - \frac{\left(sci_{i}^{l} - \mu_{k}^{m}\right)^{2}}{2\left(\sigma_{k}^{m}\right)^{2}}\right] \end{aligned} \tag{5}$$

E represents the expectation of the user's state under all data. The user's state can only be honest or malicious, and the expression for expectation can be expanded as Equation (6).

$$\begin{aligned} &E\left(z_{ik} \mid SC^{l}, \theta^{m}\right) \\ &= 1 * p\left(z_{ik} = 1 \mid SC^{l}, \theta^{m}\right) + 0 * p\left(z_{ik} = 0 \mid SC^{l}, \theta^{m}\right) \\ &= p\left(z_{ik} = 1 \mid SC^{l}, \theta^{m}\right) \end{aligned} \tag{6}$$

The expected value of the latent variable is equivalent to the conditional probability when the latent variable takes a value of 1, which is the probability of the user's state, given that the parameters of the Gaussian mixture model and the evaluation data are known, and belongs to the posterior probability. According to Bayes' theorem, the posterior probability can be expressed as Equation (7).

$$p\left(z_{ik} = 1 \mid SC^{1}, \theta^{m}\right) = \frac{\alpha_{k}^{m}\varphi_{k}\left(sc_{i}^{1}, \mu_{k}^{m}, \left(\sigma_{k}^{m}\right)^{2}\right)}{\sum_{u=1}^{2}\alpha_{u}^{m}\varphi_{u}\left(sc_{i}^{l}, \mu_{u}^{m}, \left(\sigma_{u}^{m}\right)^{2}\right)} \tag{7}$$

where $\theta^{m} = \left(\alpha_{k}^{m}, \mu_{k}^{m}, \left(\sigma_{k}^{2}\right)^{m}\right), k \in \{1,2\}$ represents the expected value of parameters after the m-th iteration. Substituting Equation (7) into Equation (5), the expected complete-data log-likelihood function Q can be obtained.

Next, we move on to the M-step of the EM algorithm, where we calculate the parameter values θ^m that corresponds to the maximum of the expected log-likelihood function. We calculate the partial derivatives of $Q(\theta, \theta^m)$ with respect to the parameters μ_k, θ_k^2, and α_k, set them to zero, and solve for each parameter. Letting $p\left(z_{ik} = 1 \mid SC^{l}, \theta^{m}\right) = \tau_{ik}$, the new parameter μ_{k}^{m+1}, $\left(\sigma_{k}^{2}\right)^{m+1}$, α_{k}^{m+1} can be expressed as follows:

$$\frac{\partial Q(\theta, \theta^{m})}{\partial \mu_{k}} = 0, \quad \mu_{k}^{m+1} = \frac{\sum_{i=1}^{n}\tau_{ik} sc_{i}^{l}}{\sum_{i=1}^{n}\tau_{ik}}$$

$$\frac{\partial Q(\theta, \theta^{m})}{\partial \sigma_{k}} = 0, \sigma_{k}^{m+1} = \frac{\sum_{i=1}^{n}\tau_{ik}\left(sc_{i}^{l} - \mu_{k}^{m+1}\right)^{2}}{\sum_{i=1}^{n}\tau_{ik}}$$

$$\frac{\partial Q(\theta, \theta^{m})}{\partial \alpha_{k}} = 0, \alpha_{k}^{m+1} = \frac{\sum_{i=1}^{n}\tau_{ik}}{n}$$

If the expectation of the log-likelihood function does not converge, we update the parameters $\theta^{m} = \left(\alpha_{k}^{m}, \mu_{k}^{m}, \left(\sigma_{k}^{2}\right)^{m}\right), k \in \{1,2\}$, and $Q(\theta, \theta^m)$. We iterate the E and M steps to update the expectation of the log-likelihood function $Q(\theta, \theta^m)$ until it converges or reaches the maximum number of iterations, and output the current parameter value θ'. Based on

the assumption that there are more honest learners than malicious ones, $\mu k'$ corresponding to $max\alpha'_k$ is the evaluation result of the model quality, denoted as sc^l.

If users cannot obtain any reward after evaluating the model, their willingness to evaluate will decrease, and malicious or arbitrary evaluation behaviors will emerge, which will increase the difficulty and error of quality evaluation, reduce the credibility of the system, and be unfavorable for the sustainable development of the sharing ecosystem. Therefore, designing reasonable and effective incentive functions is of great significance. As an attribute of users, reputation is related to the cost of purchasing models, and it can also be used as the weight of user evaluation data to measure the credibility of evaluation. Therefore, using reputation to provide targeted incentives to users, and designing an honest evaluation incentive function is necessary. The function should satisfy two conditions: motivate users to actively participate in evaluations, and motivate users to provide honest evaluations.

Firstly, we evaluate the honesty level of the user using the evaluation bias. The evaluation bias of $user_i$ is defined as the distance between the evaluation sci_i^l and the quality evaluation result sci_i^l, denoted as $Dev_i^l = |sc_i^l - sc^l|$. The smaller the evaluation bias, the more honest the user is, and the more rewards they will receive. Evaluations that are higher or lower than sc^l by the same amount are considered to have the same level of honesty. We assume that the user's evaluation of models is a real number within the range of $[0,\ 5]$, and that the evaluation bias is also a real number within the range of $[0,\ 5]$.

Secondly, the incentive function ΔR related to the honesty level is defined to measure the reputation value obtained by users participating in the evaluation in a quality evaluation process. The user's total reputation value is the cumulative result of the incentive received from a single evaluation. When reputation is used to exchange system models, the total reputation value will decrease correspondingly. To incentivize users to actively and honestly evaluate, the function ΔR should satisfy the following two properties.

Property 1. *For any participating* $user_i$, $\Delta R \geq 0$ *always holds. According to the law of large numbers, when a large number of users participates in the evaluation, the expected evaluation is closer to the model quality. However, evaluations incur costs for users. In order to encourage users to actively participate in the model evaluation process, the evaluation incentive function needs to satisfy the participation constraint. The participation constraint, also known as the individual rationality constraint, in the context of teaching quality evaluation, refers to the expected benefit of participating in the evaluation being not less than the expected benefit of not participating in the evaluation. When users do not participate in the evaluation, their reputation gain is 0, and so satisfying the participation constraint is equivalent to the evaluation incentive function* $\Delta R \geq 0$.

Property 2. *For any two users* $user_i$ *and* $user_j$, *who evaluate* sc^l, *if* $Dev_i^l \leq Dev_j^l$, *then* $\Delta R_i \geq \Delta R_j$. *The incentive function for evaluation needs to satisfy the incentive compatibility constraint, which means that the behaviors of individuals who pursue the maximization of their own interests should be consistent with the goal of maximizing the collective interests. Incentivizing users to provide honest evaluations is a goal that maximizes collective interests, as it helps to reduce the error of model quality evaluation, and improves the authenticity and accuracy of quality evaluation results. To satisfy the incentive compatibility principle, under the induction of the evaluation incentive function, honest evaluation should be the optimal strategy for users. For the same model, the more honest the user is, the smaller the evaluation deviation, and the more rewards they receive.*

Based on the above two properties, a sigmoid-prime function is introduced as the incentive function, with the evaluation deviation Dev as the independent variable. According to Skinner's reinforcement theory, the ratio of reputation value to evaluation deviation is not a fixed value, which is conducive to delaying the trend of the decay of the honest evaluation behaviors of users. The sigmoid-prime function is the derivative of the sigmoid function, and the gradient increases and then decreases. Using it as the evaluation incentive

function is conducive to long-term incentives for users to make honest evaluations. The evaluation incentive function ΔR can be expressed as:

$$\Delta R = \frac{be^{-kDev}}{(1+e^{-kDev})^2}$$

where k and b are parameters of the incentive function that affect the range and strength of incentives, as detailed in Section 5.2 and set by the educational institution deploying the EIC. The function passes through the point (0, b/4), and when the user's evaluation value is equal to the model quality assessment result, the user receives the maximum reputation value of b/4. The incentive function varies with changes in the user's evaluation and evaluation bias as follows.

Let R_i denote the reputation score of $user_i$, and let R_i^0 be the initial reputation score of the user. The change in the reputation score depends on the user's behavior. Actively participating in evaluation will increase the reputation score, while using reputation score and points to exchange for other models will decrease the reputation score. In addition, as a user's attribute, reputation can be used to indicate the credibility of the user's evaluation. The higher the reputation score, the more honest the user is in the history of evaluations, and the more likely the user will remain honest in future evaluations, indicating a higher evaluation credibility. Therefore, reputation can be used as the weight of user evaluations to calculate the weighted expectation of evaluations, assigned to the EM algorithm as the initial model parameters. In summary, a reputation-based honest evaluation incentive function is proposed based on the sigmoid-prime function as a prototype, and the user's evaluation bias as the independent variable. On the one hand, it provides feedback to users on their evaluations, and on the other hand, it encourages them to actively and honestly evaluate, constraining the evaluation behavior and promoting a good evaluation atmosphere.

4.2. Model Sharing Incentive Model

Model sharing can improve the utilization rate of models, reduce the repeated training of models, and benefit more companies and individuals. However, training a high-precision model requires a lot of data and consumes a lot of computing models. In addition, there may be competition among them. In order to protect their own interests, many model owners are unwilling to share their models. It is particularly important to establish a good incentive mechanism to encourage model owners to share their own models.

A model owner can improve the accuracy of his own model by sharing the model with others, so the model owners will benefit from each other, in addition to competition. A large number of repetitive and inferior models will increase the difficulty of obtaining information for model users, and increase the storage burden of the system.

Inspired by the above, we propose a model sharing incentive model based on evolutionary game. We first introduce the overall model, and then we design an incentive function based on the sigmoid function by integrating quality and similarity. Finally, we construct an evolutionary game model among model owners, and we analyze the impact of incentive function on model sharers' strategy selection.

First, the model owner views the incentive information of the shared model through the platform and decides whether to share his own model. If they choose to share the model and upload it to the platform, the platform will calculate the similarity between the model and the existing model based on the description information of the model. The model users can search, download, and evaluate the model. The platform will evaluate the quality of the model after receiving the evaluation. Finally, the platform calculates the results of the incentive function according to the model quality, similarity, and parameters of the incentive function, and rewards the model sharers.

4.3. Incentive Function Design

Improving the sense of value and identity of model owners and giving appropriate incentives is the key to creating a good sharing atmosphere. The use of an integral system

can encourage model owners to share models with high quality and low similarity, and realize the long-term development of the model sharing system. Model owners can use points to download the other models in the platform.

Model quality is an important factor affecting the development of the model sharing system. However, the time cost and opportunity cost of high-quality models are higher in the production process. If the quality factor is ignored and the same degree of incentive is given to users who share different quality models, model owners will be guided to share low-quality models. When the quality of most models in the system cannot satisfy the requirements of the demanders, the attractiveness of the model sharing platform to users will be greatly reduced, and sharing will lose its value.

Model repeatability is another important factor affecting the development of the model sharing system. There is a common problem of the repeated construction of models in the existing sharing system, which on the one hand increases the burden for model demanders to retrieve and browse models, and on the other hand, increases the storage pressure of the sharing system.

In order to solve the problems of poor model quality and high repeatability in the shared system, we design an incentive function I (Q, sim) related to model quality Q and similarity sim, which should satisfy the following properties. Among them, the model quality Q is calculated using the quality evaluation function in Section 4.1. The similarity sim is the ratio of the function of the repetitive part of the model to the total function of the model in the sharing system, which is calculated using the smart contract.

Theory 1. $\forall q, sim, \frac{\partial I(q,sim)}{\partial q} \geq 0, \frac{\partial I(q,sim)}{\partial sim} \leq 0$

Function I is a monotone increasing function about Q and a monotone decreasing function about sim. When the similarity is the same, the higher the quality of the teaching models is, the more incentives will be obtained. When the quality is the same, the higher the similarity of the model, the more repetitive parts of the model and existing models in the system. Even if it is shared into the system, the less benefits it brings to the system as a whole and to the model demanders, and so the less incentive it obtains.

Theory 2. $I(0, sim) = I(q, 1) = 0$

If all sharing behaviors can gain benefits, there will usually be malicious users who gain benefits without labor, such as sharing blank files, meaningless models, or copying existing models in the system. In order to prevent this phenomenon, users cannot obtain any incentive when the quality of the shared model is 0 or the similarity is 1.

Theory 3. *Let* $I(q, sim) = \phi(sim) * f(q)$; $\phi(sim)$ *denotes the impact of model similarity on incentives. We can obtain the following conclusions.*

$$\begin{cases} \varphi_{\text{sim}\to 0} \to 1 \\ \varphi_{\text{sim}\to 1} \to 0 \\ \varphi(0) = 1 \\ \varphi(1) = 0 \end{cases} \tag{8}$$

ϕ *means the proportion of the incentives actually received by the model owner in the deserved incentives. The revenue from sharing a certain model should be less than or equal to the revenue from sharing the original model with the same accuracy and function. In extreme cases, when users download models from the system and re-upload them to the system, no matter what the quality of the models are, users will not benefit from them. When users share completely original models, the benefits obtained are only related to the accuracy of model. So,* $0 < \phi(sim) < 1$, *and the closer sim is to 0, the closer* $\phi(sim)$ *is to 1.*

Theory 4. *With the increase in sim, the trend of $\phi(sim)$ is as follows:*

$$\begin{cases} \frac{d^2\varphi}{d\,sim^2} > 0, & sim < sim_{threshold} \\ \frac{d^2\varphi}{d\,sim^2} < 0, & sim > sim_{threshold} \end{cases} \tag{9}$$

It is assumed that the tolerance of model users to model similarity is nonlinear. When the similarity is small, the users are more tolerant of models, and the change of the incentive coefficient is small. When the similarity is large, the information gain brought by models to users is small, and the change of the incentive coefficient is small. With the increase in similarity, the incentive coefficient decreases, and the decreasing rate first increases and then decreases.

If functions satisfy the above properties, they can be used as incentive functions. We choose the sigmoid function, which is often used in machine learning as the excitation function. Because the sigmoid function realizes the nonlinear mapping of input data to (0, 1), the greater the absolute value of input data, the smaller the gradient and the lower the sensitivity. This change trend is shown in Figure 2.

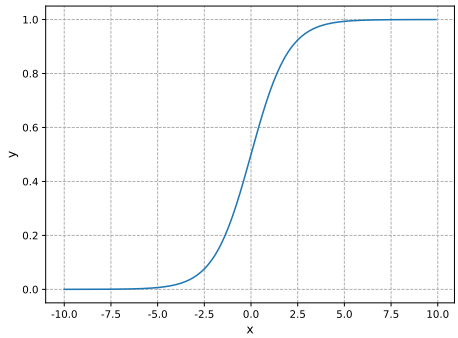


Figure 2. The Graph of the Sigmoid Function.

We can obtain the ϕ function by converting the sigmoid function.

$$\varphi = \begin{cases} 1, & sim = 0 \\ \frac{1}{1+e^{-a(1-\text{ sim-thr})}}, & 0 < sim < 1 \\ 0, & sim = 1 \end{cases} \tag{10}$$

The $thr(0 < thr < 1)$ is the threshold of similarity, and the tolerance of model demanders to model similarity is relatively sensitive near the threshold. A (a > 0) affects the decreasing speed of the excitation function. When the similarity is between 0 and 1, the trend of incentive coefficient changing with the similarity is shown in Figure 3:

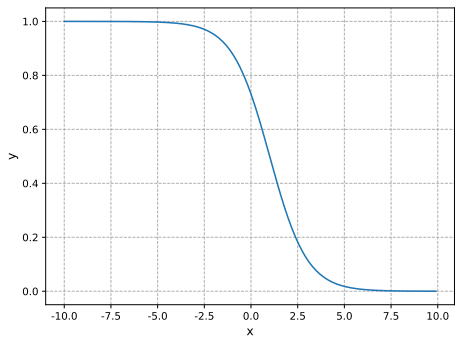


Figure 3. The Graph of the ϕ Function.

Then, we can obtain $I(q, sim)$ as follows:

$$\mathrm{I(q,sim)} = \begin{cases} \xi \mathrm{q}, & \mathrm{sim} = 0 \\ \frac{\xi q}{1+\mathrm{e}^{-\mathrm{a}(1-\mathrm{sim}-\mathrm{thr})}}, & 0 < \mathrm{sim} < 1 \\ 0, & \mathrm{sim} = 1 \end{cases} \tag{11}$$

$\xi(\xi > 0)$ indicates the excitation parameter. For specific excitation function I, when $0 < sim < 1$, $\partial I/\partial q = \xi/(1 + e^{-a(1-sim-thr)}) > 0$, and the incentive function is a monotonically increasing function of model quality; $\partial I/\partial sim = -a\xi Q e^{-a(1-sim-thr)}/(1 + e^{-a(1-sim-thr)})^2 < 0$, the incentive function is a monotonically decreasing function of model similarity. The function satisfies Property 1. $I(0, sim) = I(q, 0) = 0$; the function satisfies Theory 2. According to the figure, the function satisfies Theories 3 and 4. The function satisfies the above three theories and can be used to encourage model owners to share high-quality and low-homogeneity models.

4.4. Evolutionary Game Model

The model owner does not fully know other owners' private information (such as the quality and similarity of the model owned), the selection strategy, the utility function, and other information. The computing power to derive the optimal strategy based on the collected information is limited. Therefore, model owners belong to individuals with limited rationality. In the multiple rounds of the game, model owners can choose strategies to maximize their own interests, through learning and correcting mistakes. From the perspective of the evolutionary game, this paper analyzes the strategic preferences of model owners and the state of the group when it finally reaches dynamic equilibrium.

Assumption 1. *N is a game group composed of a large number of participants with models, which can be expressed as $N = \{1, 2, 3 \ldots\}$. Assuming that each model owner is bounded and independent, there is no conspiracy between model owners.*

Assumption 2. *S is the strategic space of the evolutionary game. S_i is the set of strategies, and s_i is the selected strategy in a certain round of the game. Each model owner has two strategies: participating in sharing models and not participating in sharing models, namely, $S_i = \{s_i | s_i = P(participate) or NP(not participate)\}$. Define $s_{-i} = \{s_1, \ldots, s_{i-1}, s_{i+1}, \ldots\}$ as the combination of strategies of other owner except the owner i. In each round of game, model owners interact with each other through repeated random matching. The evolution process conforms to natural laws, and the final group will reach a dynamic and stable state. If at a certain stage, the proportion of model owners who choose strategy si in the group is pi, the evolutionary game strategy can be expressed as (p_i, p_{-i}), and the state of the group can be expressed as $\sigma = p_i s_i + p_{-i} s_{-i}$, where $pi, p_{-i} \geq 0$ and $p_i + p_{-i} = 1$.*

Assumption 3. *U is a utility function that represents the mapping from the strategy combination to the reward; $u : s \to R$. Given a strategy combination such as $s = (s_I, s_{-i})$, the utility function $u_i(s_i, s_{-i})$ represents the benefit of model owner i when i selects strategy s_i and the strategy set selected by others other than i is s_{-i}.*

Assumption 4. *In the initial state, the model owner has a certain preference for strategy selection. Randomly select two model owners as game subjects, representing two groups. Assuming that the probabilities of both sides of the game choosing the sharing strategy are x and y, respectively, the probabilities of choosing not to participate in the sharing strategy are $1 - x$, $1 - y$, $0 \leq x, y \leq 1$, respectively. X is a function of time t; $x = x(t)$, $y = y(t)$. According to the assumption of inertia behavior, the proportions of people who choose to participate in the strategy in the two groups are x and y, respectively.*

Assumption 5. *Model accuracy and model quantity are the private information of the model owner. Before the collective makes a choice, this information is only known to the model owner. The guesses*

of other model owners about private information satisfy the common prior probability distribution. In addition, users have the same level of information about the model.

Assumption 6. *The knowledge production function is introduced to measure the value of the model. The machine learning model not only has the function of prediction, but also has the value of inspiration for the re-creation of the model. Model users can gain knowledge by accessing the shared model, and combine the newly learned knowledge with the original model to create a new model. This process is called knowledge spillover.*

There are similarities between knowledge spillover and the production process. Economics uses a production function to describe the relationship between the input factors and the output in the production process. Greitz introduced the production function into the research of knowledge sharing for the first time, and put forward the concept of the knowledge production function [45]. Since then, the knowledge production function has been widely used to study the relationship between the knowledge output and input factors. Among them, the Cobb-Douglas function is widely used to predict production and to analyze the development path because of its simple model and convenient calculation in parameter estimation. Therefore, the Griliches-Jaffe knowledge production function based on the Cobb Douglas function is introduced to measure the knowledge outputs of teaching models in the sharing process. The calculation method is $Q = AK^{\alpha}L\beta$, wherein, Q represents the benefits brought about by knowledge output, namely, the knowledge value. $A(A > 0)$ is a coefficient, which is related to technical level, teaching model type, field, productivity, and other factors. K is the model stock, expressed by the product of the model scale and quality. L is the number of personnel. α, β represents the elasticity coefficient of the knowledge stock and the number of inputs, and it represents the impact of the two factors on the knowledge output.

When no one else shares the model in the system, the model stock of model owner i $K_i = m_i q_i$, where m_i is the data volume of the model and q_i is the model quality. When other people share the model in the system, the model stock of model owner i $K_i = m_i q_i + m_j q_j$. m_j is the amount of data available in the shared model, that is, the amount of data complementary to the existing model, and q_j is the quality of the shared model. Let the number of personnel invested be 1, and the additional benefits obtained by model owner i from the system model can be expressed as $F_i = A_i[\{(m_i q_i + m_j q_j)^{\alpha} - (m_i q_i)^{\alpha}\}]$.

Other parameters used in the evolutionary game are shown in Table 1:

Table 1. Glossary.

Symbol	Meaning
F_i	The additional benefits that the model owner i obtains from the system model
I_i	Rewards obtained by model owner i from the sharing system
ω	Model owners' losses due to sharing models
c	Points paid to the system for accessing the model when the model owner does not share models
γ	Sharing income coefficient

Based on the above assumptions, the corresponding returns of game players 1 and 2 in different strategy choices are shown in Table 2:

Table 2. Payoff Matrix for Game between Model Owners.

1 \ 2	P	NP
P	$(\gamma F_1 + I_1 - \omega, \gamma F_2 + I_2 - \omega)$	$(I_1 - \omega, F_2 - c)$
NP	$(F_1 - c, I_2 - \omega)$	$(0, 0)$

Each subject has two strategic choices. There are four possible outcomes of the game, which can be summarized into the following three situations:

Case 1: When the game subjects choose not to share the strategy, neither party can access the other party's models, gain shared benefits, and pay no costs. At this time, the income is 0.

Case 2: When one party in the game entity shares models while the other party does not, it corresponds to (P, NP) or (NP, P) in the payment matrix. Most incentive mechanisms believe that "the party who chooses not to share cannot obtain the models of the sharing party" . However, realizing model sharing is the original intention of the construction of teaching the model sharing platform. Therefore, on the basis of the article, EduShare allows users to access models without sharing, but they need to pay a certain amount of points to the system. At this time, the model owners who choose to share cannot obtain the desired models from the other party, so they cannot obtain additional benefits, but they need to bear the losses caused by sharing. For the party who does not share the models, it is necessary to pay the system the credit c to access the models and to obtain additional benefits from the models. Therefore, when player 1 chooses P and player 2 chooses Non, the income of player 1 is $I_1 - \omega$, The income of entity 2 is $F_2 - c$. When player 1 chooses Non and player 2 chooses P, the income of player 1 is $F_1 - c$, and the income of player 2 is $I_2 - \omega$.

Case 3: When both sides of the game choose to share models, the model owner can not only visit the other side's models, but also understand the evaluations of others on their own models, gain more recognition, and provide direction and motivation for improving models. At this time, the benefits of model owners from shared models can be expressed as γF. Among them, $\gamma(\gamma > 1)$ is the sharing income coefficient. In addition, the model owner will also receive the sharing incentive issued by the system and bear the losses caused by sharing. Therefore, when all players choose to share, the benefit of player 1 is $\gamma F_1 + I_1 - \omega$. The income of entity 2 is $\gamma F_2 + I_2 - \omega$.

4.5. Game Process Analysis

This section analyzes the change trend of the model owner's choice preference by solving the equilibrium strategy of the evolutionary game model. The equilibrium strategy for solving the evolutionary game model can be transformed into solving the fixed point problem of the dynamic system. Since the evolution direction is related to the expected reward of the players in different strategy choices, the expected reward and the average expected reward when the players choose to share and not share are analyzed, and the dynamic system trajectory expression is constructed as follows.

When player 1 chooses the sharing strategy, the expected reward consists of two parts. If game player 2 chooses the shared strategy, then player 1 gains $\gamma F_1 + I_1 - \omega$. If player 2 chooses not to share the model, the income obtained by player 1 is $I_1 - \omega$. According to the assumptions, the probabilities of player 2 choosing the two strategies are y and $1 - y$, respectively. For player 1, the expected income E_{1P} corresponding to the strategy selected to participate in the sharing is:

$$\begin{aligned} E_{1P} &= (\gamma F_1 + I_1 - \omega) * y + (I_1 - \omega) * (1 - y) \\ &= \gamma F_1 y + I_1 - \omega \end{aligned} \tag{12}$$

In the same way, the expected income E_{1NP} of the non-shared strategy can be expressed as:

$$\mathrm{E_{1Non}} = (\mathrm{F_1} - \mathrm{c})\mathrm{y} \tag{13}$$

Then, the average expected income $\overline{E_1}$ can be expressed as:

$$\overline{E_1} = xE_{1P} + (1-x)E_{1NP} \tag{14}$$

For player 2, the strategies of sharing, not-sharing, and the average expected returns, E_{2P}, E_{2N}, and $\overline{E_2}$ are, respectively:

$$\begin{aligned} &\mathrm{E_{2P}} = (\gamma \mathrm{F_2} + \mathrm{I_2} - \omega) * \mathrm{x} + (\mathrm{I_2} - \omega) * (1 - \mathrm{x}) \\ &= \gamma \mathrm{F_2 x} + \mathrm{I_2} - \omega \\ &\mathrm{E_{2Non}} = (\mathrm{F_2} - \mathrm{c})\mathrm{x} \\ &\overline{E_2} = yE_{2P} + (1-y)E_{2N_0} \end{aligned} \tag{15}$$

Assume that in the game group 1, x' is the proportion of the individuals who choose the sharing strategy in the next round in the group. Use the finite difference form to express the dynamic change of the group strategy, then x' can be expressed as $x' = xE_{1p}/\overline{E_1}$. The change of the proportion of individuals who choose this strategy in the population per unit time can be expressed as:

$$\frac{x' - x}{\Delta t} = \frac{dx}{dt} = \frac{x\left(E_{1p} - \overline{E_1}\right)}{\overline{\mathrm{E_1}}} \tag{16}$$

The trajectory and fixed point of the differential equation are equivalent to the following equation, and we can build the replication dynamic equation of the evolutionary game.

$$\begin{aligned} \frac{dx}{dt} &= x\left(E_{1p} - \overline{E_1}\right) \\ &= x(1-x)\{[(\gamma - 1)F_1 + c]y + I_1 - \omega\} \end{aligned} \tag{17}$$

Similarly, the replication dynamic equation of game group 2 can be expressed as:

$$\frac{dy}{dt} = y(1-y)\{[(\gamma - 1)F_2 + c]x + I_2 - \omega\} \tag{18}$$

The replication dynamic equation reveals the evolution trend of the strategy of participation and sharing in the group. When the expected return of selecting shared models is greater than the average expected return, the probability of selecting this strategy in the next round of the game will increase, e.g., $dx/dy > 0$, $dy/dt > 0$. On the contrary, the probability of selecting the sharing strategy will be reduced, e.g., $dx/dt < 0$, $dy/dt < 0$. When R$dx/dt = 0$ and $dy/dt = 0$, the evolutionary game reaches the equilibrium point. Therefore, there are five equilibrium points in the model:

$$\begin{aligned} &\mathrm{E_1} = (0,0), \mathrm{E_2} = (0,1), \mathrm{E_3} = (1,0), \mathrm{E_4} = (1,1), \\ &\mathrm{E_5} = \left(\frac{\omega - \mathrm{I_2}}{(\gamma - 1)\mathrm{F_2} + \mathrm{c}}, \frac{\omega - \mathrm{I_1}}{(\gamma - 1)\mathrm{F_1} + \mathrm{c}}\right) \end{aligned} \tag{19}$$

Because the equilibrium of the evolutionary game is dynamic, the equilibrium point may not be stable. Next, we analyze the local stability of the five equilibrium points. According to evolutionary game theory, the linearization theorem is introduced to judge the stability of the fixed point of the dynamic system, which is mainly divided into the following three steps.

Step 1: According to the replication dynamic equation, the Jacobian matrix is constructed. Calculate the partial derivatives of x and y for dx/dt and dy/dt, respectively, to

obtain the Jacobian matrix as follows:

$$
\begin{aligned}
\mathrm{J} &= \begin{bmatrix} \frac{\partial(\mathrm{dx}/\mathrm{dt})}{\partial \mathrm{x}} & \frac{\partial(\mathrm{dx}/\mathrm{dt})}{\partial \mathrm{y}} \\ \frac{\partial(\mathrm{dy}/\mathrm{dt})}{\partial \mathrm{x}} & \frac{\partial(\mathrm{dy}/\mathrm{dt})}{\partial \mathrm{y}} \end{bmatrix} \\
\frac{\partial(\mathrm{dx}/\mathrm{dt})}{\partial \mathrm{x}} &= (1-2\mathrm{x})\{[(\gamma-1)\mathrm{F}_1+\mathrm{c}]\mathrm{y}+\mathrm{I}_1-\omega\} \\
\frac{\partial(\mathrm{dx}/\mathrm{dt})}{\partial \mathrm{y}} &= \mathrm{x}(1-\mathrm{x})[(\gamma-1)\mathrm{F}_1+\mathrm{c}] \\
\frac{\partial(\mathrm{dy}/\mathrm{dt})}{\partial \mathrm{x}} &= \mathrm{y}(1-\mathrm{y})[(\gamma-1)\mathrm{F}_2+\mathrm{c}] \\
\frac{\partial(\mathrm{dy}/\mathrm{dt})}{\partial \mathrm{y}} &= (1-2\mathrm{y})\{[(\gamma-1)\mathrm{F}_2+\mathrm{c}]\mathrm{x}+\mathrm{I}_2-\omega\}
\end{aligned} \tag{20}
$$

Step 2: Calculate the determinant value of the Jacobian matrix and the trace of the matrix. The determinant Det.J of the Jacobian matrix can be expressed as:

$$
\begin{aligned}
\mathrm{Det.} &= \frac{\partial(\mathrm{dx}/\mathrm{dt})}{\partial \mathrm{x}} \times \frac{\partial(\mathrm{dy}/\mathrm{dt})}{\partial \mathrm{y}} - \frac{\partial(\mathrm{dx}/\mathrm{dt})}{\partial \mathrm{y}} \times \frac{\partial(\mathrm{dy}/\mathrm{dt})}{\partial \mathrm{x}} \\
&= (1-2\mathrm{x})(1-2\mathrm{y})\{[(\gamma-1)\mathrm{F}_1+\mathrm{c}]\mathrm{y}+\mathrm{I}_1-\omega)* \\
&\{[(\gamma-1)\mathrm{F}_2+\mathrm{c}]\mathrm{x}+\mathrm{I}_2-\omega\}-\mathrm{xy}(1-\mathrm{x})(1-\mathrm{y})* \\
&[(\gamma-1)\mathrm{F}_1+\mathrm{c}][(\gamma-1)\mathrm{F}_2+\mathrm{c}]
\end{aligned} \tag{21}
$$

The trace $Tr.J$ of the Jacobian matrix can be expressed as:

$$
\begin{aligned}
\mathrm{Tr.}\ J &= \frac{\partial(\mathrm{dx}/\mathrm{dt})}{\partial \mathrm{x}} + \frac{\partial(\mathrm{dy}/\mathrm{dt})}{\partial \mathrm{y}} \\
&= (1-2\mathrm{x})\{[(\gamma-1)\mathrm{F}_1+c]\mathrm{y}+\mathrm{I}_1-\omega\} \\
&+ (1-2\mathrm{y})\{[(\gamma-1)\mathrm{F}_2+c]\mathrm{x}+\mathrm{I}_2-\omega\}
\end{aligned} \tag{22}
$$

Step 3: Judge the stability of the equilibrium point of the model according to the linearization theorem. According to the linearization theorem, the equilibrium point is stable if and only if the determinant of the Jacobian matrix is greater than 0 and the trace of the matrix is less than 0, otherwise the equilibrium point is not stable. As the number of games increases, the equilibrium point will eventually tend to evolve into a stable strategy. The determinant and trace of the Jacobi matrix corresponding to the five equilibrium points are as follows (Table 3):

Table 3. Equilibrium Analysis.

Equilibrium Point	Value of Determinant	Trace of Matrix
E_1	$(\omega - I_1)(\omega - I_2)$	$I_1 + I_2 - 2\omega$
E_2	$((\gamma-1)F_1 + I_1 + c - \omega)(\omega - I_2)$	$(\gamma-1)F_1 + I_1 + c - I_2$
E_3	$(\omega - I_1)((\gamma-1)F_2 + I_2 + c - \omega)$	$-I_1 + (\gamma-1)F_2 + I_2 + c$
E_4	$((\gamma-1)F_1 + I_1 + c - \omega)((\gamma-1)F_2 + I_2 + c - \omega)$	$-((\gamma-1)F_1 + I_1 + c - \omega) - ((\gamma-1)F_2 + I_2 + c - \omega)$
E_5	/	0

The status of the equilibrium point is discussed according to the parameter values, as follows:

Case 1: When $I_1 \geq \omega$ and $I_2 \geq \omega$, the status of the equilibrium point is shown in Table 4. At this point, (1,1) is an evolutionary stability strategy. Regardless of the initial willingness of model owners to share, the group will eventually tend to share teaching

models. At this time, the incentive function plays a strong incentive role, and the strategy selection of individual benefit maximization is consistent with the strategy selection of group benefit maximization, satisfying the incentive compatibility constraint.

Table 4. Stability Analysis of Equilibrium Point in Case 1.

Equilibrium Point	Sign of Determinant	Sign of Trace	Result
E_1	+	+	stable
E_2	−	unknown	saddle point
E_3	−	unknown	saddle point
E_4	+	−	stable
E_5	+	0	center

Case 2: When $I_1 \leq \omega - (\gamma - 1)F_1 - c$ and $I_2 \leq \omega - (\gamma - 1)F_2 - c$, the status of the equilibrium point is shown in Table 5. At this time (0,0) is an evolutionary stability strategy. Regardless of the initial willingness of model owners to share, the group will eventually tend to not share models. The incentive mechanism satisfying this condition is invalid.

Table 5. Stability Analysis of Equilibrium Point in Case 2.

Equilibrium Point	Sign of Determinant	Sign of Trace	Result
E_1	+	+	stable
E_2	−	unknown	saddle point
E_3	−	unknown	saddle point
E_4	+	+	unstable
E_5	+	0	center

Case 3: When $\omega - (\gamma - 1)F_1 - c < I_1 < \omega$ and $\omega - (\gamma - 1)F_2 - c < I_2 < \omega$, the equilibrium point status is shown in Table 6. At this time, (0,0) and (1,1) are evolutionary stability strategies, and the evolution direction is related to the equilibrium point. When the proportion of model owners selected to share in game subject 1 x>1, game subject 2 evolves towards sharing. When $x < 1$, game subject 2 evolves towards non-sharing. When the proportion of shared model owners selected by game player 2 is $y > (\omega - I_1)/((\gamma - 1)F1 + c)$, game subject 1 evolves towards sharing. When $y < c$, the game subject 1 evolves towards non-sharing.

Table 6. Stability Analysis of Equilibrium Point in Case 3.

Equilibrium Point	Sign of Determinant	Sign of Trace	Result
E_1	+	−	stable
E_2	+	+	unstable
E_3	+	+	unstable
E_4	+	−	stable
E_5	+	0	center

4.6. Dynamic Incentive for Model Sharing

Based on the above analysis, this model has different evolutionary outcomes when the incentive function satisfies different conditions. When the incentive function satisfies the conditions in Case 1, the model owner's preference for strategy selection is sharing. When the incentive function satisfies the condition in Case 2, the model owner's preference for

strategy selection is not sharing. When the incentive function satisfies the conditions in Case 3, the evolution direction of the population, and the current state and the equilibrium solution$((\omega - I_2)/((\gamma - 1)F_2 + c), (\omega - I_1)/((\gamma - 1)F_1 + c))$.

The system uses the integral incentive model owner to share the model. When the integral in the system reaches the saturation state, the attraction of the integral to the model owner decreases, and the increase in the share rate brought by the unit integral decreases, which is called the phenomenon of the diminishing marginal effect. If the incentive is set to a fixed value, if the setting is too low, the incentive effect will not be achieved. If the setting is too high, the incentive saturation speed will be too fast and the marginal effect will decrease. Therefore, the dynamic adjustment of the incentive function can achieve a long-term incentive for model owners.

Based on the results of evolutionary game analysis, the incentive function is dynamically adjusted. Realizing model sharing is the goal of maximizing social benefits. Therefore, evolutionary stability strategy (1, 1) is the expected result of the game. Since the evolutionary stability strategy corresponding to case 2 is (0, 0), which does not satisfy the incentive compatibility principle, only case 1 and case 3 are considered. Under the initial conditions, the proportion of the model owners selected to share in the system is small, and the incentive function satisfying the conditions in Case 1 is used for the incentive. After a period of time, the excitation function satisfying the conditions in Case 3 is used to continue the excitation.

In order to make the shared models in the system have high quality, the model access threshold is designed to realize the constraint on the model owner. Assume that the minimum quality of models that the model demander can accept in the sharing system is qmin, and that the maximum model similarity that can be tolerated is simmax. The threshold of model access is recorded as T, then T shall satisfy:

$$\mathrm{T} = \max(\mathrm{q_{min}}, \mathrm{q_{max}}\varphi(\mathrm{sim_{max}})) \tag{23}$$

This means that the system allows for high-quality model sharing with high similarity, and that it allows for low-quality but original model sharing. For example, the minimum acceptable model quality for model demanders is 2, and the maximum tolerable similarity corresponds to $\phi = 0.5$; the threshold value is max(2, 2.5) = 2.5. After the introduction of T, the incentive function parameters of game subject i are set as shown in Equation (24), where λ_j is the proportion of model owners who choose to share with another subject in the game.

$$\xi_{\mathrm{i}} = \begin{cases} \frac{\omega}{\mathrm{T}}, & \lambda_{\mathrm{j}} = 0 \\ \frac{\omega - [(\gamma-1)\mathrm{F_i} + \mathrm{c})]\lambda_{\mathrm{j}}}{\mathrm{T}}, & 0 < \lambda_{\mathrm{j}} < \frac{\omega}{(\gamma-1)\mathrm{F_i}+\mathrm{c}} \\ 0, & \lambda_{\mathrm{j}} > \frac{\omega}{(\gamma-1)\mathrm{F_i}+\mathrm{c}} \end{cases} \tag{24}$$

When the quality or similarity of models does not satisfy the system requirements, not sharing models is the best strategy choice for rational individuals. With the increase in the proportion of sharing strategies selected in the system, the dependence of the system on incentives gradually decreases to 0. After that, the system can still maintain a good sharing atmosphere without incentives.

Model owners can know the value of incentive parameters through smart contracts, and then decide whether to share. If the quality of models is high and the innovation is strong, and the expected return of model owners to share models is greater than the average expected return, they will choose to share models. If the model has low quality, high repeatability, and does not satisfy the system requirements, the model owner will gain less than the average income when choosing to share the model. At this time, model owners can choose to share models after improving the quality of models and reducing the repetition rate, or give up sharing models. In this way, we encourage the sharing of high-quality models, punish the sharing of inferior models, and realize the quality control of the models shared in the system.

When the model owner decides to share the model, the model will be uploaded to the local database of the educational institution, and the address summary, digital signature, detailed introduction, and parameter information of the current incentive function of the model will be stored in the blockchain together. The model quality is calculated using the quality evaluation function in Section 4.1. The incentive for model owners to share models can be calculated using Equation (11). By adjusting the parameters of the incentive function through Equation (24), the dynamic incentive for the model owner is realized, which not only reduces the burden of the shared system, but also maintains the good operation of the system and delays the incentive saturation and marginal effect decline caused by excessive incentive.

5. Experiments

In order to evaluate the effect of the proposed scheme, we have conducted two sets of numerical simulation experiments. Both groups of experiments are programmed in Python 3.7 and run on a 64-bit Win11 system with 64G RAM and Intel(R) Core(TM) i7-11700F CPU @ 2.50GHz.

5.1. Numerical Simulation Experiment of an Honest Evaluation Incentive Model

In this experiment, five models were selected for testing. Each model has a real quality $real_z$ and a set of evaluation data sc^z. We call the numpy library in Python to generate N pieces of evaluations, each of which include evaluation and reputation values. Among them, the evaluations of malicious users conform to the normal distribution, with a mean value of μ_1 and the standard deviation σ_1. The reputation value corresponding to each data point is a random number in the range of $[r_{1min}, r_{1max}]$. The evaluations of honest users conform to the normal distribution, with the mean value of μ_2 and the standard deviation σ_2. The reputation value corresponding to each data point is a random number in the range of $[r_{2min}, r_{2max}]$. We use n_1, indicating the number of malicious users. The parameters of the experiment are listed in Table 7.

Table 7. Parameters of the Quality Evaluation Experiment.

n_1	μ_1	σ_1	σ_2	$[r_{1min}, r_{1max}]$	$[r_{2min}, r_{2max}]$
100	0	0.5	1	[0.5, 1.4]	[1.2, 2.5]

In the experiment, the number of malicious users was constant, and the proportions of the two groups were changed by adjusting the number of honest users. The purpose of malicious users is to lower the quality evaluation of the model, and the malicious evaluation will be gathered around 0. Honest evaluation will be scattered around the real quality of the model. Therefore, the evaluation of malicious users is set to confirm to the normal distribution of $N(0, 0.5)$. The evaluation of honest users is set to confirm to the normal distribution with a standard deviation of 1. The reputation value of honest users is generally higher than that of malicious users, but there may be an overlap. Therefore, we set the reputation of the malicious users range to be from 0.5 to 1.4, and the reputation of honest users ranges from 1.2 to 2.5. The evaluation data are assumed to be real numbers ranging from 0 to 5; the data outside of the range need to be filtered out. The remaining data can be regarded as the simulation evaluation data of models, which are confirmed to be the truncated normal distribution $N(\mu, \sigma^2; x_l, x_r)$; the expected can be calculated as follows:

$$E[X] = \mu + \sigma \frac{\varphi(x_1) - \varphi(x_r)}{\phi(x_r) - \phi(x_1)} \tag{25}$$

Among them, φ and ϕ are the probability density function and distribution function of random variables that satisfy the standard normal distribution. x_l and x_r are two truncated positions. Since the evaluation score is between 0 and 5, the corresponding truncated points are $x_l = 0$ and $x_r = 5$.

The expectation of the truncated normal distribution obeyed via honest evaluation data is used to express the real quality of the models. The estimated quality of models is expressed using the result of the quality evaluation function based on the EM algorithm. We will analyze the deviation of the results of the quality evaluation function when the proportion of honest users is different. Referring to the method of the article, the root-mean-square-error (RMSE) is introduced to measure the comparison between the real quality of the models and the estimated quality. The calculation method is as follows (Table 8):

$$RMSE = \sqrt{\frac{\sum_{z=1}^{num}(q_{real} - q_{appro})}{num}} \tag{26}$$

Table 8. Parameters of Quality Evaluation Experiment.

μ_1	σ_1	μ_2	σ_2	rep_{1min}	rep_{1max}	rep_{2min}	rep_{2max}
0	0.5	3	1	0.5	1.5	1.2	3

When the parameters correspond to different values, the real qualities of models are calculated using Equation (25). The mean and the weighted mean, with reputation as the benchmark method, are compared with the quality evaluation method (EM) based on the EM algorithm, and the results are shown in Table 9. Because there are random errors in the generation of simulation data, in order to offset the impacts of random errors, each experimental result in the table is represented by the average of five repeated experiments with the same parameters.

Table 9. Results of Quality Assessment Methods under Different Parameters.

μ	q_{real}	α_2	*Mean*	*Weighted Mean*	*EM*
3	3.05	100	3.02	3	3.75
		90	2.79	2.86	2.47
		80	2.58	2.72	2.96
		70	2.42	2.63	3.41
		60	2.27	2.57	3.59
		50	2.07	2.42	3.86
3.5	3.64	100	3.02	3	3.75
		90	3.13	3.22	3.35
		80	2.88	3.07	4.3
		70	2.67	2.95	1.67
		60	2.46	2.83	1.48
		50	2.18	2.6	1.3
4	4.29	100	3.73	3.69	3.85
		90	3.39	3.49	4.16
		80	3.13	3.33	3.92
		70	2.88	3.2	3.94
		60	2.6	3.04	1.59
		50	2.25	2.75	3.16

The test sample with the same proportion of honest users is a group, and a group of data containing μ_2 are the five cases of 2, 2.5, 3, 3.5, and 4, and the root mean square errors of the three methods are calculated, respectively. In the three methods, the change of the root mean square error with the proportion of honest users is shown in Figure 4.

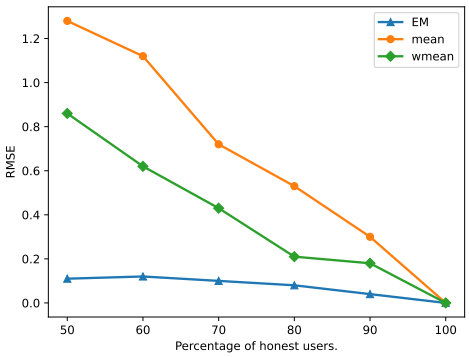


Figure 4. The relationship between RMSE and the percentage of honest users.

According to the experimental results, with an increase in the proportion of honest learners, the error of the average and weighted average methods gradually decreases. However, because the quality evaluation method based on the EM algorithm effectively filters the evaluation data of malicious learners, it is less greatly affected by the proportion of malicious learners. When there are more honest learners in the system, the result of the model quality evaluation based on the EM algorithm is closer to the real value.

5.2. Analysis of the Parameters of the Incentive Function

The experiment adopts the method of controlling variables to analyze the effects of the parameters k and b of the evaluation incentive function on the function.

Firstly, the impact of parameter k on the incentive function was analyzed. By keeping the parameter b constant and by setting $b = 4$, the value of parameter k was adjusted to 2, 4, 6, 8, and 10. Under different parameter values, the change of the evaluation incentive function ΔR with respect to the evaluation deviation Dev is shown in Figure 5. As the parameter k increases, the rate at which the function value decreases with the evaluation deviation gradually increases, and the incentive range gradually narrows. When k is 2, users whose evaluation deviation is within the range of [0, 2.9932] obtain a reputation value of greater than or equal to 0.01. When k is 10, only users whose evaluation deviation is within the range of [0, 0.5986] obtain a reputation value that is greater than or equal to 0.01.

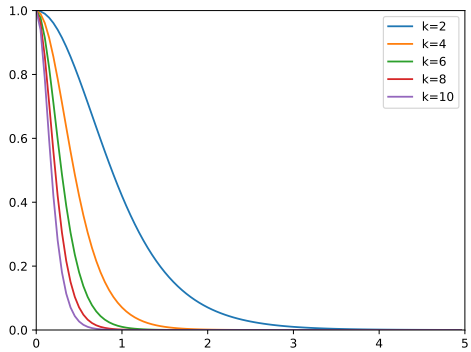


Figure 5. The relationship between the incentive function and the parameter k.

The experimental results indicate that parameter k affects the incentive range of the evaluation incentive function. If k is set too small, users can still benefit from large evaluation biases, and the incentive function has insufficient constraints on the user evaluation behavior. If k is set to be too large, only users with very accurate evaluations can benefit, which is not conducive to mobilizing users' enthusiasm for evaluation. Therefore, setting an appropriate value for k according to the distribution of the evaluation data is beneficial for incentivizing users to actively and honestly evaluate.

Secondly, we analyzed the impact of parameter b on the incentive function. Keeping parameter k constant at $k = 4$, we adjusted the value of parameter b to 1, 2, 3, 4, and 5,

respectively. The change of the evaluation incentive function ΔR with respect to the evaluation deviation *Dev* under different parameters is shown in Figure 6, with b=1 yielding the smallest extremum value and corresponding incentive value for a given evaluation deviation, while larger b values result in greater extremum values and corresponding incentive values for the same evaluation deviation. For instance, for a user with an evaluation deviation of 0.5, the expected reputation value is 0.105 when $b = 1$, while it is 0.42 when $b = 4$.

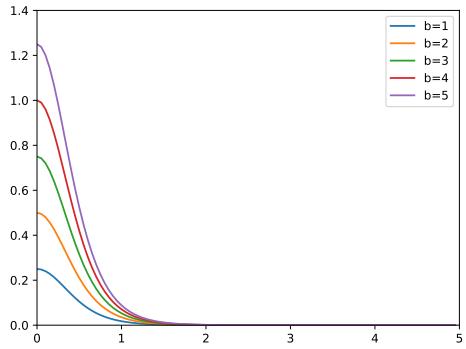


Figure 6. The relationship between the incentive function and the parameter b.

The experimental results show that parameter b affects the incentive strength of the evaluation incentive function. If b is set too small, the incentive strength is insufficient, which affects the users' participation in evaluation. If b is set too large and the incentive method is single, there will be a diminishing marginal effect. That is, with an increase in time, the incentive effect corresponding to a unit reputation value decreases until the reputation value reaches saturation and the incentive becomes ineffective. Therefore, according to the actual situation, setting an appropriate value for b can incentivize users to evaluate models in the long term and actively.

In summary, in the honest evaluation incentive function, parameter k affects the incentive range of the function, and parameter b affects the incentive strength of the function. The reasonable setting of parameter values is conducive to the long-term operation of the honest evaluation incentive function.

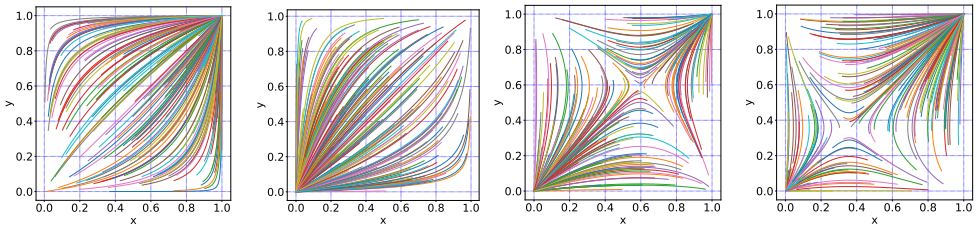
5.3. Model Numerical Simulation Experiment Based on an Evolutionary Game

To verify that the parameter value of ξ in the incentive function results in different evolutionary outcomes under different conditions, a numerical simulation experiment was designed as follows. The parameter settings used in the experiment are shown in Table 10.

Table 10. Parameters for evolutionary game experiments.

k_1	k_2	q_1	q_2	sim_1	sim_2	γ	c	A	α	β	ω	a	thr
2	2	3	4	0.2	0.1	1.1	1	1	1	1	3	10	0.5

The evolution paths and results of the game for both players under three different conditions specified in Section 4.5 are shown in Figure 7 by adjusting the parameter ξ of the incentive function. In Figure 7a–d, the x and y coordinates represent the proportions of model owners who choose to participate in the game for players 1 and 2, respectively.



(**a**) (**b**) (**c**) (**d**)

Figure 7. The relationship between game outcome and parameter ξ. (**a**) $\xi = 2$; (**b**) $\xi = 0.2$; (**c**) $\xi = 0.8$; (**d**) $\xi = 0.9$.

Set $\xi = 2.0$; the parameters satisfy the conditions in Case 1. As can be seen from Figure 7a, regardless of the initial state of both sides of the game, they will eventually converge to the equilibrium point $(1, 1)$. The larger the initial probability, the faster the convergence speed.

Set $\xi = 0.2$, the parameters satisfy the conditions in Case 2. As can be seen from Figure 7b, regardless of the initial state of both sides of the game, they will eventually converge to the equilibrium point $(0, 0)$. The larger the initial probability, the faster the convergence speed.

Set $\xi = 0.8$, the parameter satisfies the condition in Case 3, and the equilibrium solution $E5 = (0.6146, 0.6173)$. At this point, the evolutionary state is related to the initial state. At the initial time, $x > 0.6146$ and $y > 0.6173$, and the game converges to the equilibrium point $(1, 1)$. At the initial time, $x < 0.6146$ and $y < 0.6173$, the game converges to the equilibrium point $(0, 0)$, as shown in Figure 7c; At the initial time, $x < 0.6146$ and $y > 0.6173$, then game entity 2 evolves in the direction of non-sharing, and game entity 1 evolves in the direction of sharing. Figure 7d shows the changes in the critical point when $\xi = 0.9$.

In summary, the incentive model for model sharing can dynamically motivate model owners to share high-quality models, and the incentive function is feasible and effective.

6. Conclusions

This paper introduces a novel incentive model that addresses the critical issues of quality control and sharing incentives in the existing model sharing system. By encouraging model users to provide active and honest feedback, and encouraging owners to share high-quality models, the proposed model aims to improve the overall quality of the models shared within the ecosystem. Furthermore, the integration of blockchain technology enhances the security of the system, while the use of smart contracts automates the transaction process, leading to reduced operating costs. To evaluate the feasibility and effectiveness of the proposed model, extensive simulation experiments were conducted. The results highlight the potential of this model to establish a model sharing market that supports the high-quality, long-term development of the model sharing ecosystem. Overall, the proposed model has significant implications for improving the quality of models shared within the ecosystem and promoting a sustainable model sharing market. The findings of this study could inform the development of similar incentive models for other collaborative systems where quality control and sharing incentives are critical.

In the proposed approach, it is required that a certain number of user evaluations are gathered before calculating the model quality, which may result in delayed rewards for model owners. In our future work, we plan to address this issue and to make improvements. Additionally, we will explore how to establish more reasonable pricing mechanisms for the models.

Author Contributions: Conceptualization, C.L.; Methodology, C.L. and S.W.; Investigation, H.W. and Y.X.; Writing—original draft, C.L. and Y.Z.; Writing—review and editing, E.X. and S.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the National Natural Science Foundation of China (No. 61772044, 62077044, and 62293555), the Major Program of Science and Technology Innovation 2030 of China (No. 2022ZD0117105), and the Major Program of Natural Science Research Foundation of the Anhui Provincial Education Department (2022AH040148).

Data Availability Statement: No data were used to support this study.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ahonen, T.; Hadid, A.; Pietikainen, M. Face Description with Local Binary Patterns: Application to Face Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 2037–2041. [CrossRef] [PubMed]
2. Ohn-Bar, E.; Trivedi, M.M. Looking at Humans in the Age of Self-Driving and Highly Automated Vehicles. *IEEE Trans. Intell. Veh.* **2016**, *1*, 90–104. [CrossRef]
3. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
4. Danaee, P.; Ghaeini, R.; Hendrix, D.A. A Deep Learning Approach for Cancer Detection and Relevant Gene Identification. *Biocomputing* **2017**, *2017*, 219–229. [CrossRef]
5. Ribeiro, M.; Grolinger, K.; Capretz, M.A. MLaaS: Machine Learning as a Service. In Proceedings of the 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 9–11 December 2015. [CrossRef]
6. Weng, J.; Weng, J.; Huang, H.; Cai, C.; Wang, C. Fedserving: A federated prediction serving framework based on incentive mechanism. In Proceedings of the IEEE INFOCOM 2021-IEEE Conference on Computer Communications, Vancouver, BC, Canada, 10–13 May 2021; pp. 1–10.
7. Boumaiza, A.; Sanfilippo, A. AI for Energy: A Blockchain-based Trading Market. In Proceedings of the IECON 2022—48th Annual Conference of the IEEE Industrial Electronics Society, Brussels, Belgium, 17–20 October 2022; pp. 1–6. [CrossRef]
8. Reshmy, A.; Paulraj, D. Data mining of unstructured big data in cloud computing. *Int. J. Bus. Intell. Data Min.* **2018**, *13*, 147–162. [CrossRef]
9. Li, K.; Wang, S.; Cheng, X.; Hu, Q. A misreport-and collusion-proof crowdsourcing mechanism without quality verification. *IEEE Trans. Mob. Comput.* **2021**, *21*, 3084–3095. [CrossRef]
10. Hu, Q.; Wang, S.; Cheng, X.; Ma, L.; Bie, R. Solving the crowdsourcing dilemma using the zero-determinant strategies. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 1778–1789. [CrossRef]
11. Wang, S.; Sun, W.; Ma, L.; Lv, W.; Cheng, X. Quantum game analysis on extrinsic incentive mechanisms for P2P services. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *31*, 159–170. [CrossRef]
12. Weng, J.; Weng, J.; Cai, C.; Huang, H.; Wang, C. Golden Grain: Building a Secure and Decentralized Model Marketplace for MLaaS. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 3149–3167. [CrossRef]
13. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
14. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
15. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
16. Lim, J.E.; Lee, J.; Kim, D. The effects of feedback and goal on the quality of crowdsourcing tasks. *Int. J. Hum.–Comput. Interact.* **2021**, *37*, 1207–1219. [CrossRef]
17. Chan, K.W.; Li, S.Y.; Ni, J.; Zhu, J.J. What feedback matters? The role of experience in motivating crowdsourcing innovation. *Prod. Oper. Manag.* **2021**, *30*, 103–126. [CrossRef]
18. Stumpf, S.; Rajaram, V.; Li, L.; Burnett, M.; Dietterich, T.; Sullivan, E.; Drummond, R.; Herlocker, J. Toward Harnessing User Feedback for Machine Learning. In Proceedings of the 12th International Conference on Intelligent User Interfaces, IUI '07, New York, NY, USA, 28–31 January 2007; pp. 82–91. [CrossRef]
19. Raman, R.; Gupta, N.; Jeppu, Y. Framework for Formal Verification of Machine Learning Based Complex System-of-Systems. *Insight* **2023**, *26*, 91–102. [CrossRef]
20. Krichen, M.; Mihoub, A.; Alzahrani, M.Y.; Adoni, W.Y.H.; Nahhal, T. Are Formal Methods Applicable To Machine Learning And Artificial Intelligence? In Proceedings of the 2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH), Riyadh, Saudi Arabia, 9–11 May 2022; pp. 48–53. [CrossRef]
21. Guo, Y.; Zhang, C.; Wang, C.; Jia, X. Towards public verifiable and forward-privacy encrypted search by using blockchain. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 2111–2126. [CrossRef]

22. Guo, Y.; Zhang, C.; Jia, X. Verifiable and forward-secure encrypted search using blockchain techniques. In Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–7.
23. Wang, M.; Guo, Y.; Zhang, C.; Wang, C.; Huang, H.; Jia, X. MedShare: A Privacy-Preserving Medical Data Sharing System by Using Blockchain. *IEEE Trans. Serv. Comput.* **2023**, *16*, 438–451. [CrossRef]
24. Liu, X.; Li, Y.; Wang, M. A Lightweight Authentication Protocol Based on Confidential Computing for Federated Learning Nodes. *Proc. Netinfo Secur.* **2022**, *22*, 37–45. [CrossRef]
25. Wang, Y.; Chen, L.; Zhong, M. Progress in Blockchain Solutions Based on Zero-Knowledge Proof. *Proc. Netinfo Secur.* **2022**, *22*, 47–56. [CrossRef]
26. Ren, T.; Jin, R.; Luo, Y. Network Intrusion Detection Algorithm Integrating Blockchain and Federated Learning. *Proc. Netinfo Secur.* **2021**, *21*, 27–34. [CrossRef]
27. Guo, Y.; Xie, H.; Miao, Y.; Wang, C.; Jia, X. Fedcrowd: A federated and privacy-preserving crowdsourcing platform on blockchain. *IEEE Trans. Serv. Comput.* **2020**, *15*, 2060–2073. [CrossRef]
28. Li, C.; Qu, X.; Guo, Y. TFCrowd: A blockchain-based crowdsourcing framework with enhanced trustworthiness and fairness. *EURASIP J. Wirel. Commun. Netw.* **2021**, *2021*, 168. [CrossRef]
29. Li, M.; Weng, J.; Yang, A.; Lu, W.; Zhang, Y.; Hou, L.; Liu, J.N.; Xiang, Y.; Deng, R.H. CrowdBC: A blockchain-based decentralized framework for crowdsourcing. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *30*, 1251–1266. [CrossRef]
30. Ma, Y.; Sun, Y.; Lei, Y.; Qin, N.; Lu, J. A survey of blockchain technology on security, privacy, and trust in crowdsourcing services. *World Wide Web* **2020**, *23*, 393–419. [CrossRef]
31. Buterin, V. A next-generation smart contract and decentralized application platform. *White Pap.* **2014**, *3*, 2-1.
32. Wang, S.; Ouyang, L.; Yuan, Y.; Ni, X.; Han, X.; Wang, F.Y. Blockchain-enabled smart contracts: Architecture, applications, and future trends. *IEEE Trans. Syst. Man, Cybern. Syst.* **2019**, *49*, 2266–2277. [CrossRef]
33. Atzei, N.; Bartoletti, M.; Cimoli, T. A survey of attacks on ethereum smart contracts (sok). In Proceedings of the Principles of Security and Trust: 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of ETAPS 2017, Uppsala, Sweden, 22–29 April 2017; Proceedings 6. Springer: Berlin/Heidelberg, Germany, 2017; pp. 164–186.
34. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
35. Zhang, C.; Guo, Y.; Jia, X.; Wang, C.; Du, H. Enabling Proxy-Free Privacy-Preserving and Federated Crowdsourcing by Using Blockchain. *IEEE Internet Things J.* **2021**, *8*, 6624–6636. [CrossRef]
36. Alsalami, N.; Zhang, B. SoK: A Systematic Study of Anonymity in Cryptocurrencies. In Proceedings of the 2019 IEEE Conference on Dependable and Secure Computing (DSC), Hangzhou, China, 18–20 November 2019; pp. 1–9. [CrossRef]
37. Zhang, C.; Guo, Y.; Du, H.; Jia, X. PFcrowd: Privacy-Preserving and Federated Crowdsourcing Framework by Using Blockchain. In Proceedings of the 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS), Hangzhou, China, 15–17 June 2020; pp. 1–10. [CrossRef]
38. Wang, S.; Zhang, Y.; Guo, Y. A Blockchain-Empowered Arbitrable Multimedia Data Auditing Scheme in IoT Cloud Computing. *Mathematics* **2022**, *10*, 1005. [CrossRef]
39. Smith, J.M. *Evolution and the Theory of Games*; Cambridge University Press: Cambridge, UK, 1982.
40. Hofbauer, J.; Sigmund, K. *Evolutionary Games and Population Dynamics*; Cambridge University Press: Cambridge, UK, 1998.
41. Maynard Smith, J. Game theory and the evolution of cooperation. *Evol. Mol. Men* **1983**, 445–456.
42. Taylor, P.D.; Jonker, L.B. Evolutionary stable strategies and game dynamics. *Math. Biosci.* **1978**, *40*, 145–156. [CrossRef]
43. Chen, Y.; Guo, Y.; Wang, Y.; Bie, R. Toward Prevention of Parasite Chain Attack in IOTA Blockchain Networks by using Evolutionary Game Model. *Mathematics* **2022**, *10*, 1108. [CrossRef]
44. Luo, Y.; Li, J.; Xie, Z.; Zhou, G.; Xiao, X. MOOC Course Evaluation Based on Big Data Analysis. In Proceedings of the 2018 International Conference on Computer Science, Electronics and Communication Engineering (CSECE 2018), Wuhan, China, 7–8 February 2018; Atlantis Press: Amsterdam, The Netherlands, 2018; pp. 349–352.
45. Griliches, Z. Issues in assessing the contribution of research and development to productivity growth. *Bell J. Econ.* **1979**, *10*, 92–116. [CrossRef]